Study of PI based stochastic optimal control and Implementation of MPPI for Aggressive Driving

Rajnish Tiwari, Vishnu R., Ameya Wagh
rtiwari@wpi.edu, vrudrasamudram@wpi.edu, aywagh@wpi.edu
Department of Robotics Engineering, Worcester Polytechnic Institute,
Worcester, MA, USA

***Abstract*— Usually optimal control theory deals with the systems with deterministic dynamics which is a huge abstraction. We could never correctly predict the outcome when we give a set of inputs to a system. Considering the stochasticity in the model to some extent helps to understand and reduce the uncertainties. In this report, we present a study on path integral approach to solve stochastic optimal control problems. Dynamic programming approach or Entropy - Free Energy based methods gives a partial differential equation which is known as Hamilton -Jacobi-Bellman (HJB). These Hamilton-Jacobi-Bellman equations are computationally insolvable for higher dimensions. One way to solve this problem is by linearizing Hamilton-Jacobi-Bellman equations and representing them as path integral, this method yields optimal solution and is computationally efficient. We implemented path integral algorithm in model predictive setting to perform swing-up action on cart-pole to study the effect of parameters like temperature and variance. And, we used Autorally platform provided by Gerogia Tech to study the effects of changing the cost functions in the context of aggressive driving.**

Keywords: Stochastic Optimal Control, Path Integral, swing-up action, Aggressive driving.

I. INTRODUCTION

Control theory gives formal description of how a system can move from its current state to a desired state. Classical linear and non linear control methods achieve this objective, but never considers the effort that need to be put in. On the other hand optimal control drives the system to the desired state at minimal cost, where cost can mean time spent, or energy spent or any other quantity. Usually these theories deal with the systems with deterministic dynamics which are huge abstractions of actual systems. In many of real life situations one faces the extra difficulty of uncertainty due to model imperfections and unpredictable external influences. Stochastic Control or Stochastic Optimal Control studies the systems with the existence of uncertainty either in observations or in the noise that drives the evolution of the system.

A popular method for stochastic optimal control is to compute an optimal feedback map from the system state to the control via dynamic programming. For systems modeled by stochastic differential equations (SDEs), dynamic programing leads to a partial differential equation that is known as the Hamilton-Jacobi-Bellman (HJB) Equation. The complexity and memory requirements of grid-based partial differential equation (PDE) solvers increase exponentially as the dimension of the system increases[11]. This makes grid-based methods impractical for problems of large dimension. Additionally, Optimal control computation usually involves estimation of the value function (or optimal cost-to-go) which is hard to perform exactly for non-linear system with quadratic rewards and Gaussian noise. In these cases, we either have to rely on approximations of the system dynamics, e.g. by linearization or the value function. However, such approximations can significantly degenerate the quality of the estimated controls and hinder the application for complex, non-linear tasks [11]. These difficulties motivate the researchers interest to expand nonlinear stochastic optimal control in terms of theoretical generalizations and algorithms. These methods provide the ability to solve stochastic optimal control problems with forward sampling of stochastic differential equations (SDEs). The resulting stochastic optimal control frameworks are known under the names of Path Integral (PI) control for continuous time, Kullback Leibler (KL) control for discrete time, or more generally Linearly Solvable Optimal Control[5].

The path integral optimal control framework is derived for continuous time stochastic systems affine in controls and noise and for finite horizon optimal control problems[9]. This method provides a mathematically sound methodology for developing optimal control algorithms based on stochastic sampling of trajectories. The key idea in this framework is that the value function for the optimal control problem is transformed using the FeynmanKac lemma into an expectation over all possible trajectories, which is known as a path integral. This transformation allows stochastic optimal control problems to be solved with a Monte Carlo approximation using forward sampling of Stochastic Differential Equations (SDEs)[5].

In general the HJB is impossible to solve analytically, and numerical solutions are intractable due to the curse of dimensionality. One way to proceed is to consider the class of control problems in which the HJB can be linearized, and, consequently, expressed as a path integral [2]. Using this approach, researchers have developed computationally efficient methods that have been successfully applied to control, for example, cart-pole, multi agent systems like swarm of quadrotors, and RC Cars [10].

II. RELATED WORK

"The path integral formulation of quantum mechanics is a description of quantum theory that generalizes the action

principle of classical mechanics. It replaces the classical notion of a single, unique classical trajectory for a system with a sum, or functional integral, over an infinity of quantum-mechanically possible trajectories to compute a quantum amplitude.[8]" Researchers found a way to apply this path integral formulation to stochastic control.

Path integral control is a theory of stochastic optimal control that connects optimal control to key ideas in physics. This can be found in early works of Prof. H J Kappen in [2].

[1] gave interesting insights into symmetry breaking phenomena while stating conditions under which the nonlinear and second order HJB could be transformed into a linear PDE similar to the backward chapman Kolmogorov PDE. In [7] the path integral stochastic optimal control was extended to the case of multi-agent dynamics. All the theory was developed for dynamical systems of low dimentionality with control transition and diffusion matrices constant. Even though linear PDEs are easier to be solved with the use of Feynamn - Kac lemmas this connection was not initially made in [2]. In [4] the authors have generalized the path integral control framework. This could be applied to stochastic dynamics with state dependent control transition and diffusion matrices and they have used the Feynman Kac lemma to approximate solution of the resulting linear PDE.

There has been various algorithms developed using the path integral control method. Several works have been published where the path integral approach is used in reinforcement learning, for instance [11][3]. In [5], a straightforward application of path integral control is implemented as the iterative feedback control law in its open-loop formulation. This requires that sampling takes place only from the initial state of the optimal control problem. A more effective approach is to use the path integral control framework to find the parameters of a feedback control policy. This can be found in [3].

The path integral approach can be applied in a model predictive control setting. Basically, in the work of H. Kappen, the problem problem is treated in continuous-time. In [9] and [10], approximations to the solutions are made using numerical methods to express the problem in discrete-time. The optimal control law obtained from the theory in [5] is an iterative updated law which can be applied in model predictive setting [9]. This is suitable for implementation of the algorithm in GPU by parallelly sampling thousands of trajectories at a time.

III. Methodology

In this section, we give a brief introduction to the theory of path integral control and how it is derived, discuss the same in model predictive control setting, and finally explain the algorithm.

A. Path Integral Control

In Path Integral Control approach, we start with Hamilton-Jacobi-Bellman PDE for control and noise affine system. Then, with exponential transformation of value function with noise assumption, we linearize the PDE, which enables the Feynman-Kac theorem to be applied. This theorem relates the linear PDE to path integral representation of the value function.

Let $x_t = x(t) \in \mathbb{R}^n$ denote the state of a dynamical system at time t, $u(x_t, t) in \mathbb{R}^n$ denotes a control input for the system, $\tau : [t_0, T] \rightarrow \mathbb{R}^n$ represent a trajectory of the system, and $dw \in \mathbb{R}^p$ is a Brownian disturbance. Suppose that the dynamics take the following form:

$$dx = (f(x_t, t) + G(x_t, t)u(x_t, t))dt + B(x_t, t)dw \quad (1)$$

That is, the dynamics are affine in control and subject to an affine Brwonian disturbance. We also assume that G,B,f, and x are partitioned as follows:

$$G(x_t, t) = \begin{pmatrix} 0_{l\times m} \\ G_c(x_t, t) \end{pmatrix}, \; B(x_t, t) = \begin{pmatrix} 0_{l\times p} \\ B_c(x_t, t) \end{pmatrix}$$

$$f(x_t, t) = \begin{pmatrix} f_a(x_t, t) \\ f_c(x_t, t) \end{pmatrix}, \; x_t = \begin{pmatrix} x_t^{(a)} \\ x_t^{(c)} \end{pmatrix} \quad (2)$$

where

$$G_c(x_t, t) : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^{(n-l)\times m}$$
$$B_c(x_t, t) : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^{(n-l)\times p}$$
$$f_a(x_t, t) : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^{l\times 1}$$
$$f_c(x_t, t) : \mathbb{R}^n \times \mathbb{R}^+ \rightarrow \mathbb{R}^{(n-l)\times 1}$$

and

$$x_t^{(a)} \in \mathbb{R}^l \; and \; x_t^{(c)} \in \mathbb{R}^{n-l}$$

The state-space formulation given by equations (1) and (2) can represent a large class of dynamical systems. gather* In the classical stochastic optimal control setting we define optimal control sequence $u(\cdot)$ such that:

$$\mathbf{u}^*(\cdot) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}} \; \mathbb{E}_{\mathbb{Q}} \left[\phi(\mathrm{x}_T, T) + \int_{t_0}^{T} \mathcal{L}(\mathrm{x}_t, \; \mathbf{u}_t, \; t)\mathrm{d}t \right] \quad (3)$$

where the expectation is taken with respect to the dynamics defined in equations (1) and (2). We consider the costs composed of an arbitrary state-dependent term and a quadratic control cost:

$$\mathcal{L}(\mathrm{x}_t, \; \mathbf{u}_t, \; t) = q(\mathrm{x}_t, \; t) + \frac{1}{2}\mathbf{u}_t^{\mathbf{T}}\mathbf{R}(\mathrm{x}_t, \; t)\mathbf{u}_t \quad (4)$$

and dynamics are affine in control.

The relative entropy between the probability distribution induced by uncontrolled dynamics $\mathbb{P}$ and controlled dynamics $\mathbb{Q}(u)$ can be computed by applying the Girsanov's theorem as,

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

Where $\Sigma(\mathrm{x}_t,\ t) = \mathbf{B}(\mathrm{x}_t,\ t)\mathbf{B}(\mathrm{x}_t,\ t)^{\mathrm{T}}$. Then we make an assumption that the control cost matrix takes the form:

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

So, RHS of equation (3) will become

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

From the information theoretic concepts of free energy and relative entropy we get the expression for the path integral as the following equation[5]:

$$-\lambda \mathcal{F}(S(\tau)) = \inf_{\mathbb{Q}} \left[\mathrm{E}_{\mathbb{Q}}[S(\tau)] + \lambda \mathbb{D}_{KL}(\mathbb{Q}\|\mathbb{P})\right] \tag{8}$$

In equation (8)

- $\lambda \in \mathbb{R}^+$
- $S(\tau)$ - State dependent cost-to-go term
- $\mathcal{F}(S(\tau))$ - Free energy distribution over $S(\tau)$ which is given by $\mathcal{F}(S(\tau)) = \log(\mathbb{E}_{\mathbb{Q}}[\exp(-\frac{1}{\lambda}S(\tau))])$
- $\mathbb{P}$ - Probability measure over the space of trajectories $\mathbb{Q}$ - Any Probability measure such that $\mathbb{Q}$ is absolutely continuous with $\mathbb{P}$
- $\mathbb{D}_{KL}(\mathbb{Q}\|\mathbb{P})$ = $\mathbb{E}_{\mathbb{Q}}\left[\log \frac{d\mathbb{Q}}{d\mathbb{P}}\right]$. - KL Divergence

Infimum of the equation (8) attained at $\mathbb{Q}^*$ which is the optimal probability measure in terms of the Radon-Nikodym derivative with respect to uncontrolled dynamics

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} = \frac{\exp(-\frac{1}{\lambda}S(\tau))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda}S(\tau))]} \tag{9}$$

Derivation can be found in [6].

In stochastic optimal control, generally, we try to solve stochastic HJB equation. Here, we could solve the minimization problem, instead of solving difficult PDE, defined by equation (8) by reducing the distance between the distribution induced by the controller, $\mathbb{Q}(u)$ and the optimal probability measure, $\mathbb{Q}*$, defined by the Radon-Nikodym derivative $\frac{\mathbb{Q}*}{\mathbb{P}}$. A minimization problem can be defined using the relative entropy between $\mathbb{Q}*$ and $\mathbb{Q}(u)$ [9].

$$\mathbf{u}^*(\cdot) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}} \quad \mathbb{D}_{KL}(\mathbb{Q}^*\|\mathbb{Q}(\mathbf{u})) \tag{10}$$

According to the definition of relative entropy,

$$\mathbb{D}_{KL}(\mathbb{Q}^*\|\mathbb{Q}(\mathbf{u})) = \mathbb{E}_{\mathbb{Q}} * \left[\log\left(\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})}\right)\right] \tag{11}$$

In order to find the KL-Divergence between optimal control distribution $\mathbb{Q}^*$ and distribution induced by the controller $\mathbb{Q}(u)$, we need to find an expression for the Radon-Nikodym derivative $\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})}$.

Using chain rule, we can write,

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})} = \frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{P}} \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\mathbf{u})} \tag{12}$$

According to the Girsanov's theorem,

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}(\mathbf{u})} = \exp(\mathcal{D}(\tau,\ \mathbf{u}(\cdot))) \tag{13}$$

.

where,

$$\mathcal{D}(\tau,\ \mathrm{u}(\cdot)) = -\int_0^T \mathbf{u}_t^{\mathrm{T}} \mathrm{G}(\mathrm{x}_t,\ t)^{\mathrm{T}} \Sigma(\mathrm{x}_t,\ t)^{-1} \mathbf{B}(\mathrm{x}_t,\ t) dw^{(0)} + \frac{1}{2}\int_0^T \mathbf{u}_t^{\mathrm{T}} \mathbf{G}(\mathrm{x}_t,\ t)^{\mathrm{T}} \Sigma(\mathrm{x}_t,\ t)^{-1} \mathbf{G}(\mathrm{x}_t,\ t)\mathbf{u}_t \mathrm{d}t \tag{14}$$

Where, $\mathrm{d}wx^{(0)}$ is a Brownian motion with respect to $\mathbb{P}$(i.e. $\mathbb{E}_{\mathbb{P}}\left[\int_0^t \mathrm{d}wx^{(0)}\right] = 0, \forall t$).

Substituting equations (9) and (13) in equation (12) gives

$$\frac{\mathrm{d}\mathbb{Q}^*}{\mathrm{d}\mathbb{Q}(\mathbf{u})} = \frac{\exp(-\frac{1}{\lambda}S(\tau))\exp(\mathcal{D}(\tau, \mathbf{u}(\cdot)))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda}S(\tau))]} \tag{15}$$

Now, by substituting equation (15) in (11) we get the following expression for

$$\mathbb{D}_{KL}(\mathbb{Q}^*\|\mathbb{Q}(\mathbf{u})) = \mathbb{E}_{\mathbb{Q}^*}\left[\log\left(\frac{\exp(-\frac{1}{\lambda}S(\tau))\exp(\mathcal{D}(\tau, \mathbf{u}(\cdot)))}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda}S(\tau))]}\right)\right] \tag{16}$$

Since $S(\tau)$ does not depend on the controls $u\,(\cdot)$, we can exclude the terms involving $S(\tau)$ in equation (16). This results in the following equation,

$$\underset{\mathbf{u}(\cdot)}{\operatorname{argmin}}\ \mathbb{D}_{KL}(\mathbb{Q}^*\|\mathbb{Q}(\mathbf{u})) = \underset{\mathbf{u}(\cdot)}{\operatorname{argmin}}\ \mathrm{E}_{\mathbb{Q}^*}[\mathcal{D}(\tau,\ \mathbf{u}(\cdot))] \tag{17}$$

*1) **Discretization**:* In discrete time the dynamics of the system will become,

$$\mathrm{dx}_{t_j} = (\mathbf{f}(\mathbf{x}_{t_j},\ t_j) + \mathbf{G}(\mathbf{x}_{t_j},\ t_j)\mathbf{u}_j)\Delta t + \mathbf{B}(\mathrm{x}_{t_j},\ t_j)\epsilon_j\sqrt{\Delta t} \tag{18}$$

Consider the class of step function for control in discrete time,

$$\mathbf{u}_t = \begin{cases} \vdots \\ \mathbf{u}_j \quad \text{if } j\Delta t \le t \le (j+1)\Delta t \\ \vdots \end{cases} \tag{19}$$

With $j = 0, 1, ..., N$. Parameterizing $\mathcal{D}(\tau,\ \mathbf{u}(\cdot))$ using the above equation yields:

$$-\sum_{j=0}^{N} \mathbf{u}_j^{\mathrm{T}} \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathrm{x}_t,\ t) dw^{(0)} + \frac{1}{2} \mathbf{u}_j^{\mathrm{T}} \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathrm{x}_t,\ t) \mathrm{d}t\ \mathbf{u}_j \tag{20}$$

where,

$$\boldsymbol{\mathcal{G}}(\mathrm{x},\ t) = \mathbf{G}(\mathrm{x},\ t)^{\mathrm{T}} \Sigma(\mathrm{x},\ t)^{-1} \mathbf{B}(\mathrm{x}, t)$$

$$\boldsymbol{\mathcal{H}}(\mathrm{x},\ t) = \mathbf{G}(\mathrm{x},\ t)^{\mathrm{T}} \Sigma(\mathrm{x},\ t)^{-1} \mathbf{G}(\mathrm{x},\ t)$$

$$N = T/\Delta t$$

Taking the expectation of $\mathcal{D}(\tau,\ \mathbf{u}(\cdot))$ with respect to $\mathbb{Q}^*$ gives,

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}^*}[\mathcal{D}(\tau,\ \mathbf{u}(\cdot))] = &-\sum_{j=0}^{N} \mathbf{u}_j^{\mathrm{T}} \mathbb{E}_{\mathbb{Q}^*} \left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathrm{x}_t,\ t) dw^{(0)} \right] \\ &+ \sum_{j=0}^{N} \frac{1}{2} \mathbf{u}_j^{\mathrm{T}} \mathbb{E}_{\mathbb{Q}^*} \left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathrm{x}_t,\ t) \mathrm{d}t \right] \mathbf{u}_j \end{aligned} \tag{21}$$

It is clear that the equation (21) is convex with respect to each u_j. So, optimal u_j^* can be found by taking the gradient of equation (21) with respect to u_j. This yields,

$$\mathbf{u}_j^* = \mathbb{E}_{\mathbb{Q}^*} \left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathrm{x}_t,\ t) \mathrm{d}t \right]^{-1} \mathbb{E}_{\mathbb{Q}^*} \left[\int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathrm{x}_t,\ t) \mathrm{dw}^{(0)} \right] \tag{22}$$

For small Δt,

$$\begin{aligned} \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{H}}(\mathrm{x}_t,\ t) \mathrm{d}t &\approx \boldsymbol{\mathcal{H}}(\mathrm{x}_{t_j},\ t_j) \Delta t \\ \int_{t_j}^{t_{j+1}} \boldsymbol{\mathcal{G}}(\mathrm{x}_t,\ t) \mathrm{dw}^{(0)} &\approx \boldsymbol{\mathcal{G}}(\mathrm{x}_{tj},\ t_j) \epsilon_j \sqrt{\Delta t} \end{aligned} \tag{23}$$

This approximation changes the equation (22) to,

$$\mathbf{u}_j^* = \frac{1}{\Delta t} \mathbb{E}_{\mathbb{Q}^*}[\boldsymbol{\mathcal{H}}(\mathrm{x}_{t_j},\ t_j)]^{-1} \mathbb{E}_{\mathbb{Q}^*} \left[\boldsymbol{\mathcal{G}}(\mathrm{x}_{t_j},\ t_j) \epsilon_j \sqrt{\Delta t} \right] \tag{24}$$

To sample from the distribution over uncontrolled dynamics $\mathbb{P}$, we need change the expectation from expectation with respec to $\mathbb{Q}^*$ to expectation with respect to $\mathbb{P}$. This enables to directly sample trajectories from $\mathbb{P}$ to approximate the controls. This can be achieved by using Radon-Nikodym derivative:

$$\begin{aligned}\mathbf{u}_j^* = \frac{1}{\Delta t} \mathrm{E}_{\mathbb{P}} &\left[\frac{\exp(-\frac{1}{\lambda} S(\tau)) \boldsymbol{\mathcal{H}}(\mathrm{x}_{t_j}, t_j)}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda} S(\tau))]} \right]^{-1} \\ &\mathbb{E}_{\mathbb{P}} \left[\frac{\exp(-\frac{1}{\lambda} S(\tau)) \boldsymbol{\mathcal{G}}(\mathrm{x}_{t_j}, t_j) \epsilon_j \sqrt{\Delta t}}{\mathrm{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda} S(\tau))]} \right] \end{aligned} \tag{25}$$

*2) **Importance Sampling**:* In practice, sampling from the distribution from uncontrolled dynamics $\mathbb{P}$ is an inefficient method as it would be like turning on the machine and waiting for the natural noise in the system dynamics to produce the desired response. A novel method, importance sampling is derived in [10] by changing the initial control input and variance of the sampling distribution.

Using importance sampling, instead of sampling from $\mathbb{P}$, we sample from a new probability distribution $q_{\mathbf{u}}^{\nu}$ that is corresponding to the distribution over the following system dynamics.

$$\mathrm{dx}_{t_j} = \mathbf{f}(\mathbf{x}_{t_j},\ t_j) \Delta t + \mathbf{G}(\mathbf{x}_{t_j},\ t_j) \mathbf{u}_{t_j} \Delta t + \mathbf{B}_E(\mathbf{x}_{t_j},\ t_j) \epsilon_j \sqrt{\Delta t} \tag{26}$$

Here, the new diffusion matrix $\mathbf{B}_E$ is defined as,

$$\mathbf{B}_E(\mathbf{x}_t) = \begin{pmatrix} (\mathbf{B}_a(\mathbf{x}_t) & 0 \\ 0 & \nu \mathbf{B}_c \end{pmatrix} \tag{27}$$

Using the derivation of importance sampling from [10], we change the running cost from $q(x_t, t)$ to:

$$\begin{aligned}\tilde{q}(\mathbf{x}_t,\ \mathbf{u}_t,\ \epsilon_t,\ t) = q(\mathbf{x}_t,\ t) + \frac{1}{2} \mathbf{u}_t^{\mathrm{T}} \mathbf{R} \mathbf{u}_t + \lambda \mathbf{u}^{\mathrm{T}} \boldsymbol{\mathcal{G}} \frac{\epsilon}{\sqrt{\Delta t}} + \\ \frac{1}{2} \lambda (1 - \nu^{-1}) \frac{\epsilon^{\mathrm{T}}}{\sqrt{\Delta t}} \mathbf{B}_c^{\mathrm{T}} (\mathbf{B}_c \mathbf{B}_c^{\mathrm{T}})^{-1} \mathbf{B}_c \frac{\epsilon}{\sqrt{\Delta t}} \end{aligned} \tag{28}$$

Because we shifted the random variable from a zero mean term $\mathrm{B}_c \epsilon_j \sqrt{\triangle t}$ to the non-zero mean term $\mathbf{G}_c \mathbf{u} \Delta t + \mathbf{B}_E \epsilon_j \sqrt{\Delta t}$, the update rule (25) also changes to

$$\begin{aligned}\mathbf{u}_j^* = \mathbf{u}_j + \mathrm{E}_{\mathbb{P}} &\left[\frac{\exp(-\frac{1}{\lambda} \tilde{S}(\tau)) \boldsymbol{\mathcal{H}}(\mathrm{x}_{t_j}, t_j)}{\mathbb{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda} \tilde{S}(\tau))]} \right]^{-1} \\ &\mathbb{E}_{\mathbb{P}} \left[\frac{\exp(-\frac{1}{\lambda} \tilde{S}(\tau)) \boldsymbol{\mathcal{G}}(\mathrm{x}_{t_j}, t_j) \frac{\epsilon_j}{\sqrt{\Delta t}}}{\mathrm{E}_{\mathbb{P}}[\exp(-\frac{1}{\lambda} \tilde{S}(\tau))]} \right] \end{aligned} \tag{29}$$

where,

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

*3) **State Independent Control Matrix**:* Let the control matrix and diffusion matrix have the form,

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

Which means, there is no correlation between the noises in actuated state and non-actuated state. Then, the covariance matrix becomes,

$$\Sigma(\mathrm{x}) = \begin{pmatrix} \mathrm{B}_a(\mathrm{x}) \mathrm{B}_a(\mathrm{x})^{\mathrm{T}} & 0 \\ 0 & \mathbf{B}_c \mathbf{B}_c^{\mathrm{T}} \end{pmatrix} \tag{31}$$

Now, expressions given for the terms $\boldsymbol{\mathcal{H}}(\mathbf{x}_{t_j})$, $\boldsymbol{\mathcal{G}}(\mathbf{x}_t)$ in equation (20) become,

$$\boldsymbol{\mathcal{H}} = \mathbf{G}_C^{\mathrm{T}}(\mathbf{B}_c\mathbf{B}_c^{\mathrm{T}})^{-1}\mathbf{G}_c \quad \boldsymbol{\mathcal{G}} = \mathbf{G}_c^{\mathrm{T}}(\mathbf{B}_C\mathbf{B}_C^{\mathrm{T}})^{-1}\mathbf{B}_C \tag{32}$$

Since, these matrices are state independent, equation (29) becomes,

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

The terms inside the parenthesis is approximated as:

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

B. Model Predictive Path Integral

The path integral control update law is applied in a model predictive control setting. We roll out a certain number of trajectories that are sampled from the importance sampling distribution along a finite time horizon and estimate the optimal control input. But, we use only the first input from the generated control sequence.

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;
$\Delta t, \mathbf{x}_{t_0}, \mathbf{f}, \mathbf{G}, \mathbf{B}, \nu$: 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**
      $\mathbf{x}_{i+1} = \mathbf{x}_i + (\mathbf{f} + \mathbf{G}(\mathbf{u}_i + \delta\mathbf{u}_{i,k}))\Delta t$;
      $\tilde{S}(\tau_{i+1,k}) = \tilde{S}(\tau_{i,k}) + \tilde{q}$;
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{u}_i \leftarrow \mathbf{u}_i + \left[\sum_{k=1}^{K}\left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)\delta\mathbf{u}_{i,k}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_{i,k})\right)}\right)\right]$;
  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: Model Predictive Path Integral algorithm

The algorithm requires an initial input sequence. This is done either by initializing input buffer with zeros or by using a secondary controller such as PD feed-forward and its inputs as initial input sequence. U_{init} is used to initialize new control input, this is usually a constant or zero. The MPPI loop can be executed as a task based control loop or an infinite control loop. The task completed can be checked by setting a desired state or can be time limited and break the loop once it is reached.

The first loop computes K trajectories for N finite horizon with brownian motion. For each trajectory generated, a cost is computed and stored in memory. As this process is mutually exclusive, it can be parallelly run using multiple threads or on a GPU. The second loop computes an optimal input sequence using least cost of the trajectories for N finite horizons. The top of the stack value is given to actuator and the whole input control sequence is left shifted by 1. To maintain the length of buffer U_{init} is appended to the input control sequence. The states are then updated from the state estimator.

IV. IMPLEMENTATIONS

For our experiment, we considered the dynamics of the form,

$$\mathrm{d}\mathbf{x}_t = f(x_t, t)\Delta t + G(x_t, t)\left(\mathbf{u}(x_t, t) + \epsilon\sqrt{\Delta t}\right) \tag{35}$$

$\frac{\epsilon}{\sqrt{\Delta t}}$ can be considered as random change in the control input and denoted as $\delta\mathbf{u}$.

i.e.,

$$\delta\mathbf{u} = \frac{\epsilon}{\sqrt{\Delta t}}$$

Then, we have $\mathbf{B}_c(x_t, t) = \mathbf{G}_c(x_t, t)$. From this result, the values of $\mathcal{H}$ and $\mathcal{G}$ in equation (32) become 1. With these values, our update law for optimal control input becomes,

$$\mathbf{u}_j^* = \mathbf{u}_j + \sum_{k=1}^{K}\left(\frac{\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)\frac{\epsilon_{j,k}}{\sqrt{\Delta t}}}{\sum_{k=1}^{K}\exp\left(-\frac{1}{\lambda}\tilde{S}(\tau_k)\right)}\right) \tag{36}$$

A. Cart-pole inverted pendulum

1) Dynamic model:

$$\ddot{x} = \frac{1}{m_c + m_p sin^2(\theta)}\left(u + m_p sin\theta\left(l\dot{\theta}^2 + gcos\theta\right)\right) \tag{37}$$

$$\ddot{\theta} = \frac{1}{l\left(m_c + m_p sin^2(\theta)\right)}\left(-ucos\theta - m_p l\dot{\theta}^2 cos\theta sin\theta\right) \tag{38}$$

Where m_c and m_p are masses of the cart and pole, l is the length of the pole and θ being the angle made by the pole with negative y-axis.

The cart-pole inverted pendulum model was implemented in MATLAB and the focus of this experiment was to achieve optimization parameters for the controller and thus not much effort was put in to decrease the execution time. I have made a Python implementation for the same code.

2) Cost function:

- Running cost

$$S = 6\times p^2 + 12\times(1+\cos\theta)^2 + 0.1\times\dot{\theta}^2 + 0.1\times\dot{p}^2 \tag{39}$$

A go to goal controller was designed for an inverted pendulum to reach a position $x = 0$ and $\theta = \pi$.

To increase speed the variables were initialized as place-holders similar to a strictly typed code.The dynamics and cost functions were implemented as pluggable modules for faster code debugging and re-usability.

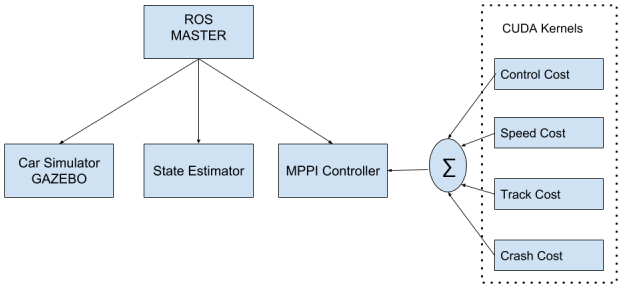

Fig. 2: AutoRally ROS framework

B. AutoRally

AutoRally is an RC car platform developed and maintained by Georgia Tech for advanced perception and control research. It is currently used extensively for research in aggressive driving for autonomous cars. The ROS framework is mainly divided into 3 parts.

1) Car Simulator: An RC car model is designed and simulated in Gazebo which resembles the real world physics of a racing car. The environment is an elliptical path where the car runs trying to maintain a desired speed. This simulator is later replaced by an actual RC car which is to be tested in a real racing environment.

2) State Estimator: The state estimator nodes take care of predicting the position orientation and velocity of the car using state of the art inertial measurement units, laser scanners and cameras. It is critical for the state estimator to be accurate within centimeters and work precisely at a high sampling rate.

3) Controller: There multiple controllers implemented in this package such as waypoint follower and MPPI but we will limit out discussion only to MPPI controllers. As the dynamics of the car are not affine, the dynamics of the car is a data driven model where multiple discrete basis functions are used. A better and more accurate dynamic model is generated by using neural networks as a regression model. To maintain a high sampling rate, cost functions are executed parallely on graphics processing unit. This is done by implementing cost functions as CUDA kernels. In our experiments we tweaked these kernels to study the effect on the cars behavior.

4) Cost Functions:

- Control cost
$$W \times U \times U^T$$
- Speed cost
$$(v - v_{desired})^2$$
- Track cost
$$100 \times |(\frac{x}{13})^2 + (\frac{y}{6})^2 - 1|$$
- Crash cost
$$\begin{cases} 0 & heading \leq \frac{\pi}{2} \\ constant & otherwise \end{cases}$$

We tried to weigh each cost before summation and observed different behaviors.

$$TotalCost = W_1 \times ControlCost + W_2 \times SpeedCost + W_3 \times TrackCost + W_4 \times CrashCost.$$

It can be seen that the control cost and speed cost are implemented in a generic form where control cost is a scaled value of squares of input, and speed cost penalizes the algorithm to keep up with the desired speed. The track cost in this case is designed keeping in mind the elliptical track. As these cost computations are executed in parallel, the time of execution does not depend on the number of cost functions.

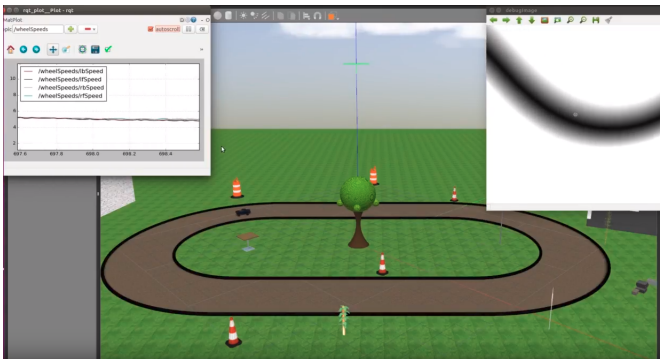

Fig. 3: AutoRally Gazebo simulation

V. RESULTS & DISCUSSIONS

We implemented model predictive path integral for cart-pole inverted pendulum. From our simulation experiments we understood that selection of the parameters decides the behavior of the system. These parameters could be cost function, time horizon, or exploration variance of MPPI algorithm. The state dependent cost functions for the cart-pole inverted pendulum consists four components: quadratic costs to reach desired location (x), reach desired angle (θ), minimize change in position and angle.

We observed that MPPI is sensitive to the selection of cost function, which drives the system to a state where the expectation over all trajectories will be too less to obtain an optimal input update. We have tested the cost function given in [10] as well as expression for cost function given in [9] which lead us to the situation described above. To resolve this issue, we normalized the cost functions over all samples. The normalization method also failed to give a proper solution. Finally, tuning the cost functions gave us the expected results shown in the figure 4.

The time horizon also place an important role in the performance of the algorithm. For longer time horizons, it is difficult to optimize because of more number of variable to be considered. Having long time horizons also results in

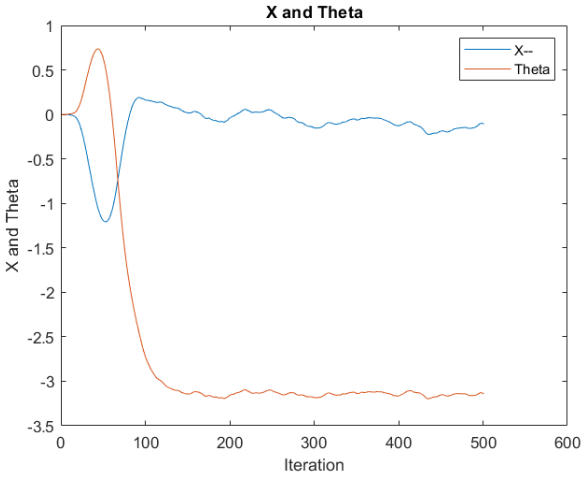

Fig. 4: Swing-up action with tuned cost function

higher cost values for the trajectories, which leads to the aforementioned problem. The effect of time horizon can be seen in figure

Exploration variance also has major impact on the performance of the algorithm. With lesser variance, the algorithm does not produce reasonable output while higher variance requires to define suitable value for $mathbf{}R$. For this simulation the temperature coefficient (λ) is $\frac{1}{\mathbf{R}}$. Figure 5 shows the effect of different λ values for a constant variance and time steps. The change in cost function can be observed in figure 6.

The simulation of exploration and state evolution can be seen in this video, and swing-up action of the inverted pendulum with two different cost functions can be seen in these videos [1],[2]

Source code (MATLAB and Python implementations) for this project can be accessed at `https://github.com/vvrs/MPPIController.git`

VI. Conclusion and Future Work

This paper briefly elaborates the math behind the Model Predictive Path Integral algorithm. We gradually build the mathematical foundation and key concepts such as discretization and importance sampling required to understand the algorithm. We then discussed the implementation of MPPI on cart-pole inverted pendulum and its results. To understand MPPI for aggressive driving, AutoRally framework was studied and modified to understand the effect of cost function on the controllers behavior.

Research in using MPPI for obstacle avoidance can be propelled by assigning infinite costs to obstacles, thus the optimal U^* generated tends to have trajectories far away from the obstacles, essentially avoiding them. A more in depth analysis of avoiding local-minimum can help solve optimization issues. A new direction to this research can be given by investigating Monte Carlo methods.

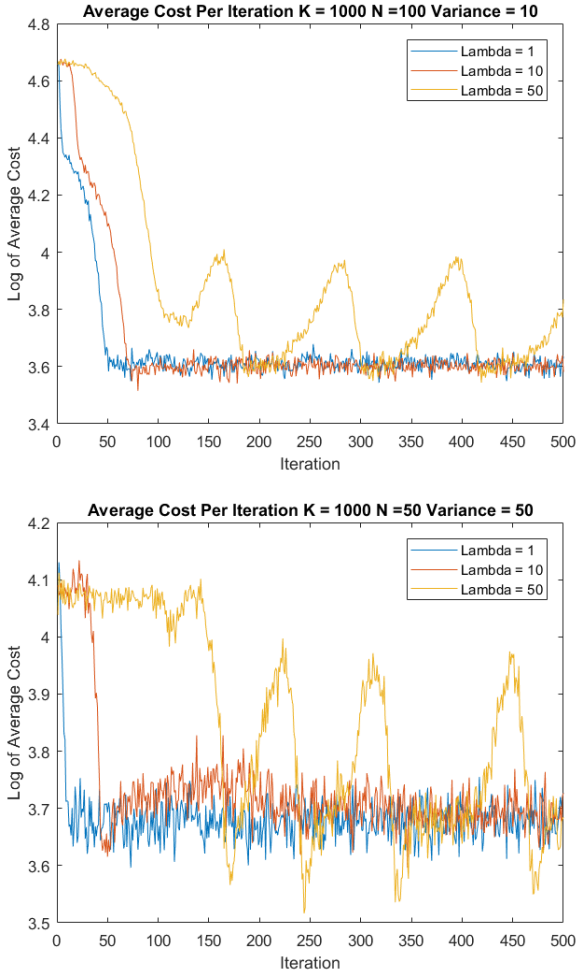

Fig. 5: The effect of λ

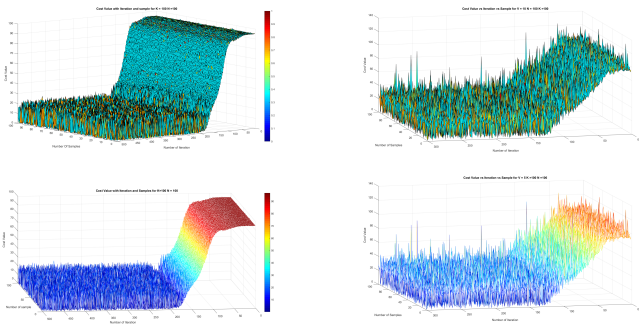

Fig. 6: Effect of variance on average cost over iterations

VII. Acknowledgements

We would like to thank Prof. Jie Fu for her help and invaluable guidance throughout the project.

References

[1] H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. 2005.

[2] Hilbert J. Kappen. Linear theory for control of nonlinear stochastic systems. *Phys. Rev. Lett.*, 95:200201, Nov 2005.

[3] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11:3137–3181, December 2010.

[4] Evangelos Theodorou, Freek Stulp, Jonas Buchli, and Stefan Schaal. An iterative path integral stochastic optimal control approach for learning robotic tasks. *IFAC Proceedings Volumes*, 44(1):11594 – 11601, 2011. 18th IFAC World Congress.

[5] Evangelos Theodorou and Emanuel Todorov. Relative entropy and free energy dualities: Connections to path integral and kl control. In *CDC*, pages 1466–1473. IEEE, 2012.

[6] Evangelos A. Theodorou. Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations. *Entropy*, 17(5):3352–3375, 2015.

[7] Wim Wiegerinck, Bart van den Broek, and Hilbert Kappen. Stochastic optimal control in continuous space-time multi-agent systems, 2012.

[8] Wikipedia. Path integral formulation, year =.

[9] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, May 2016.

[10] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, Jan 2017.

[11] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information theoretic model predictive control: Theory and applications to autonomous driving, 2017.