Model Predictive Path Integral Control

Vaggelis Dorovatas
NeuroFuzzy Control Project
ECE, NTUA

1 Problem Formulation

The issue that the described algorithm aims to tackle involves control and path optimization. It addresses this challenge by constructing a mathematical framework based in both stochastic optimal control theory and information theory. The algorithm was initially developed for a specific application, namely, the control of a small (1/5 scaled) autonomous racing vehicle referred to as AutoRally ([3]). The primary purpose for the vehicle was to navigate a dirt track as fast as possible while staying within the track boundaries[14]. The authors argue that by considering this application they could test their idea in a real world setting, experimentally proving their algorithm, while in parallel identifying practical issues related to robustness, performance and computational feasibility, which would be difficult to detect in less capable systems. Moreover, they support that the specific application is not constraining, as both the setup and the algorithm can be easily adapted to full-size autonomous driving systems. Additionally, they extend the scope to include experiments on various other navigating systems, again reinforcing their claim about the generality of the proposed algorithm.

Key Contribution of MPPI The authors posit that although existing control methodologies have demonstrated effectiveness in standard vehicle tasks like lane keeping, turning, and parking, there exists a critical frontier in control, particularly at the limits of vehicle performance, that prior research has not comprehensively tackled. This is crucial, as executing appropriate aggressive maneuvers can play a pivotal role in avoiding or mitigating collisions. MPPI is specifically designed to investigate and address this challenging issue.

Why Stochastic Optimal Control? Aggressive autonomous driving can be naturally formulated as a stochastic optimal control problem, where a cost function on the state and control input is specified and the goal is to minimize the total cost (expected cumulative cost) subject to the constraint imposed by the stochastic dynamics of the vehicle. The advantage of stochastic optimal control is that it integrates planning and control (execution) into a unified step by simultaneously accounting for both the noise characteristics and the vehicle dynamics during optimization [12]. This makes it a suitable choice when operating at the limits of the vehicle dynamics, where the external environment (planning part) and vehicle dynamics (control part) must be considered together. Earlier attempts to solve the problem did not use stochastic optimal control. Instead, they relied on a hierarchical approach [10], where the problem is split into a path planning sub-task (performed by a path planner) and a control sub-task (performed by a controller). First, the path planning algorithm calculates a path based on driving-related criteria and then the controller receives the path and calculates the controls (steering and throttle) based on the vehicle dynamics. As the authors in [12] argue, although this approach is simpler, it exhibits some limitations, the main being that the path planner has no knowledge of the underlying vehicle dynamics (in some cases it has access only to kinematics constraints). Consequently, when planning for aggressive maneuvers, it may compute an unfeasible path or prune a path that appears aggressive (outside of some safe limits) but is, in fact, feasible.

From the previous discussion it is apparent that using stochastic optimal control can be beneficial but in practice, it presents some challenges. The state space in autonomous driving may be too high dimensional for global methods like solving the Hamilton-Jacobi-Bellman equation to apply, and the main issue is that it involves *non-linear dynamics and non-convex objectives*. Variations of MPC that consider stochastic dynamics have been developed, but most of them rely on tools from constrained optimization, which means that convexification (such as with quadratic approximation) of

the cost function and first or second order approximations of the dynamics are required. The power of MPPI is that is a sample-based (MPC-variant) algorithm which can optimize for general non-convex cost criteria [11]. An illustration of the advantages of incorporating discontinuous functions can be observed in the realm of autonomous racing. In that context, one can utilize large weighted indicator functions as components of the running cost to penalize instances of deviating from the track (hitting the boundaries).

2 Mathematical Background

We proceed by presenting the main mathematical concepts behind the MPPI algorithm. This involves incorporating ideas from Stochastic Optimal Control and leveraging results from Path Integral Theory by assuming control-affine dynamics of the system. In addition, the problem can be independently formulated from an information-theoretic standpoint. The derivation of the mathematical expressions used in the final algorithm can be approached from either field, each offering distinct interpretations and assurances.

2.1 Stochastic Optimal Control

2.1.1 Stochastic Differential Equations

We begin by introducing a simple SDE, called a Wiener process or a Brownian Motion:

$$dX_t = X_{t+dt} - X_t = dW_t \tag{1}$$

with dW_t an infinitesimal mean zero Gaussian variable: $E[dW_t] = 0$, $Var[dWt] = vdt$. Then with initial conditions x_1 and t_1:

$$X_t = x_1 + \int_{t_1}^{t} dW_s \tag{2}$$

Since the increments are independent, X_t is Gaussian distributed and thus:

$$p(x_2, t_2|x_1, t_1) = \frac{1}{\sqrt{2\pi v(t_2 - t_1)}} e^{-\frac{(x_2-x_1)^2}{2v(t_2-t_1)}} \tag{3}$$

Now, we are considering a more realistic system described by the SDE below:

$$dX_t = f(X_t, t)dt + dW_t, \tag{4}$$

whre dW_t is a Wiener process.

In this case $p(x_2, t_2|x_1, t_1)$ may be very complex and is generally not known. We define two special cases of this general distribution that will help us .... We do so, by fixing the initial or final conditions:

First, by fixing the initial conditions we define $\rho(x, t) = p(x, t|x_0, 0)$. Then (Fokker-Planck forward equation):

$$\partial_t \rho(x, t) = -\nabla(f(x, t)\rho(x, t)) + \frac{1}{2} v \nabla^2 \rho(x, t) \tag{5}$$

By fixing the final conditions, we define $\psi(x, t) = p(z, T|x, t)$. Then (Kolmogorov backward equation):

$$-\partial_t \psi(x, t) = f(x, t)\nabla\psi(x, t) + \frac{1}{2} v \nabla^2 \psi(x, t) \tag{6}$$

2.1.2 Adding Control to the system

So far we saw simple SDEs modelling an uncontrolled stochastic dynamical system. We now introduce a general SDE that can describe an arbitrary stochastic system that we wish to control. An SDE in its general form can be decomposed into a deterministic part, called drift, and a stochastic part, called diffusion, and thus written in the form:

$$dx = f(x_t, u_t, t)dt + B(x_t, u_t, t)dw, \tag{7}$$

A fundamental equation for solving these stochastic optimal control problems is the Hamilton-Jacobi-Bellman (HJB) equation. A closed form solution for the HJB equation of the general SDE described above can be derived (see [7]). In our case, our focus lies in the control-affine dynamics regime. Consequently, we will proceed directly to formulate the Hamilton-Jacobi-Bellman (HJB) equation for this specific setting. The conducted analysis draws significant inspiration from [14] and [5]. Proofs not presented here, can be found in these references.

2.2 Path Integral Control Theory

Control-affine Dynamics As we mentioned, we are interested in systems which can be described by a special case of the general SDE. In particular:

$$dx = [f(x_t, t)dt + G(x_t)u(x_t, t)]dt + B(x_t, t)dw. \tag{8}$$

The property of these systems is that the controls affect the system via a **linear** transformation. More precisely, $x_t = x(t) \in \mathbb{R}^N$ denote the state of a dynamical system at time t, $u(x_t, t) \in \mathbb{R}^m$ denotes a control input for the system, $\tau : [t_0, T] \to \mathbb{R}^n$ represents a trajectory of the system and $dw \in \mathbb{R}^p$ is a **brownian** disturbance.

We now proceed to find the HJB equation. The value function $V(x_t, t)$ (cost-to-go) for this optimal control problem is:

$$V(x_t, t) = \min_u \mathbb{E}_{\mathbb{Q}} \left[\phi(x_T, T) + \int_t^T \left(q(x_t, t) + \frac{1}{2} u(x_t, t)^T R(x_t, t) u(x_t, t) \right) dt \right]. \tag{9}$$

The stochastic HJB and boundary condition is then (using Bellman's optimality principle):

$$-\partial_t V = \min_u \left[(f + Gu)^T \nabla_x V + \frac{1}{2} tr(BB^T \nabla_{xx} V) + q + \frac{1}{2} u^T R u \right] \tag{10}$$

$$V(x_T, T) = \phi(x_T). \tag{11}$$

The equation above is quadratic with respect to u. So we can solve for the optimal control input by taking the gradient of the expression. We result in $u*$ being:

$$u^* = -R^{-1} G^T \nabla_x V. \tag{12}$$

Then, we perform 3 steps. First we plug u^* back to the equation. Second we define a transformation ψ of the value function and finally we impose a relation between B, R and G. By doing so, our PDE becomes linearizable. More specific, the three steps described are:

`STEP 1`

$$-\partial_t V = q + f^T \nabla_x V - \frac{1}{2} \nabla_x V^T G R^{-1} G^T \nabla_x V + \frac{1}{2} tr(BB^T \nabla_{xx} V) \tag{13}$$

$$V(x_T, T) = \phi(x_T). \tag{14}$$

STEP 2

$$V(x,t) = -\lambda log(\Psi(x,t)) \tag{15}$$

STEP 3
For all $x \in \mathbb{R}^n$ and all $t \in [0,T]$:

$$B(x,t)B(x,t)^T = \lambda G(x,t)R(x,t)^{-1}G(x,t)^T \tag{16}$$

An interpretation of the result of Step 3 is that *if a state suffers from high variance it is necessary that the state can be directly actuated, and that the cost of actuation is low*.

By following these steps and calculating the derivatives of V in terms of Ψ, the stochastic HJB equation becomes linear (in terms of Ψ):

$$\partial_t \Psi(x_t,t) = \frac{\Psi(x_t,t)}{\lambda} q(x_t,t) - f(x_t,t)^T \Psi_x - \frac{1}{2} tr(B(x_t,t)B(x_t,t)^T \Psi_{xx}). \tag{17}$$

The strength of the resulting transformation in the previous session, lies in the capability to leverage Path Integral Theory, which relates **linear** PDEs to expectations. In particular, we can apply the Feynman-Kac formula which gives the solution to the linear PDE for the initial condition x_0 and time $t = 0$ as:

$$\Psi(x_{t_0},0) = \mathbb{E}_{\mathbb{P}}\left[exp\left(-\frac{1}{\lambda} \int_0^T q(x,t)dt \right) \Psi(x_T,T) \right]. \tag{18}$$

Here $\mathbb{P}$ is the probability of trajectories with respect to the uncontrolled dynamics (eq. 7). By taking into account that $\Psi(x_T) = e^{-\frac{1}{\lambda}\phi(x_T)}$, we can re-write the equation as:

$$\Psi(x_{t_0},0) = \mathbb{E}_{\mathbb{P}}\left[exp\left(-\frac{1}{\lambda} S(\tau) \right) \right], \tag{19}$$

$$S(\tau) = \phi(x_T) + \int_{t_0}^T q(x_t,t)dt. \tag{20}$$

Notice that $S(\tau)$ is the cost-to-go (state-dependent) of the trajectory. The final equation yields two interesting results. First, it is a forward process in time (whereas HJB in the beginning was a backward process). Second, the expectation is with respect to the uncontrolled dynamics and the control costs have vanished. Although this seems like computing the optimal control without considering the control dynamics or costs, the controls are linked to the optimization process through the uncontrolled dynamics because of the assumption between controls and noise (eq. 16). So G (the control matrix) is being taken into consideration implicitly through B and λ. Moreover, we can derive the optimal control $u*$ from equations 12, 15 and the resulting Ψ. The computations are not trivial but yield (for a full derivation see [8], [4]).

$$u^* = \frac{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(\tau))v_k]}{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(\tau))]}, \tag{21}$$

$$u^* \approx \frac{1}{I}\sum_{i=0}^{I-1} \frac{exp(-\frac{1}{\lambda}S(\tau_i))v_{k,i}}{\frac{1}{I}\sum_{n=0}^{I-1} exp(-\frac{1}{\lambda}S(\tau_n))} = \sum_{i=0}^{I-1} w_i v_{k,i}. \tag{22}$$

We note that a discretization has been performed to use Monte-Carlo Sampling. We will derive in more detail the result later (in the Information Theoretic Framework). We show this equation just to introduce the notion of trajectory weights as shown in eq. 22.

The fact that the expectation is taken over the uncontrolled distribution $\mathbb{P}$ makes sampling inefficient. If we perform a change of measure given by the Radon-Nikodym derivative, we can write the expectation with respect to a controlled distribution:

$$\Psi(x_{t_0}, 0) = \mathbb{E}_{\mathbb{Q}}\left[\frac{d\mathbb{Q}}{d\mathbb{P}} exp\left(-\frac{1}{\lambda}S(\tau)\right)\right], \tag{23}$$

$$\frac{d\mathbb{Q}}{d\mathbb{P}} = exp\left(-\frac{1}{2}\int_0^T u_t^T G^T \Sigma^{-1} G u_t dt - \int_0^T u_t^T G^T \Sigma^{-1} B dw\right) \tag{24}$$

Then Ψ becomes:

$$\Psi(x_{t_0}, 0) = \mathbb{E}_{\mathbb{Q}}\left[exp\left(-\frac{1}{\lambda}\tilde{S}(\tau)\right)\right], \tag{25}$$

$$\tilde{S}(\tau) = \phi(x_T) + \int_{t_0}^T q(x_t, t) + \frac{\lambda}{2} u_t^T G^T \Sigma^{-1} G u_t dt + \lambda \int_0^T u_t^T G^T \Sigma^{-1} B dw. \tag{26}$$

What we have achieved is introducing explicitly the controlled distribution in the expectation. What we would like to ultimately achieve is efficient sampling. Introducing a controlled distribution is a first step, since if this controlled distribution generates trajectories with low cost ("good trajectories") we expect the samples to be better. The problem though is that this formulation does not give any information on how to build a control scheme (control inputs) in order to achieve better sampling. What it gives is a theoretical guarantee that if you take the expectation given from eq. 23 or eq. 25 you will achieve the optimal control. There are various works in the literature (starting from [9]) which investigate what a good importance sampler in this setting is. The intuition behind the idea is that a Monte Carlo approximation of the optimal control solution is a weighted average, where the weight depends on the path cost. If the variance of the weights is high, then a lot of samples are required to obtain a good estimate. So what we want is choose a u that minimizes the variance of the weights. Kappen et al. [1], use the Cross-Entropy method [2], build a parametric controller $u(x_t, t|\theta)$ and update θ in the direction where the KL-Divergence between the controlled distribution (definded by u) and the optimal distribution is minimized. The optimal distribution is defined in a similar way that is defined here in the Information Theoretic Framework in the next section.

2.3 Information Theoretic Framework

We next frame the problem from an information theoretic standpoint. The idea is to make a formulation based on free-energy and KL-Divergence and define more explicitly an optimal distribution.

We will first define the two quantities mentioned above. In the definitions below, S is the cost-to-go of a trajectory, V denotes the trajectory (explicitly or implicitly) and λ is a positive scalar (with the same meaning as before). Then, the free-energy can be defined as:

$$\mathcal{F}(S, \mathbb{P}, x_0, \lambda) = -\lambda log\left(\mathbb{E}_{\mathbb{P}}\left[exp\left(-\frac{1}{\lambda}S(V)\right)\right]\right). \tag{27}$$

As it can be seen, the free-energy is a measure of the expected cost of a trajectory generated under a probability distribution $\mathbb{P}$ and it is a positive number bounded aboeve by 1. This distribution can be interpreted as the distribution generating trajectories under the uncontrolled stochastic dynamics. Also, notice the similarity between free-energy and the value function in the linearizable stochastic HJB case.

KL-Divergence is defined below. This quantity can be thought of as measure of distance between two probability distributions (technically is not a distance because it is not symmetric). We denote by $\mathbb{P}$ and $\mathbb{Q}$ two arbitrary probability distributions for which the Random-Nikodym derivative $\frac{d\mathbb{Q}}{d\mathbb{P}}$ exists [6]. Then the KL-Divergence of these two distributions is:

$$\mathbb{KL}(\mathbb{Q}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}}\left[log\left(\frac{d\mathbb{Q}}{d\mathbb{P}}\right)\right]. \tag{28}$$

By utilizing these quantities we can derive an upper bound for the free-energy of a system. The complete proof can be found in [14]. The result is shown below (again with some assumptions about the existence of the Random-Nikodym derivatives):

$$\mathcal{F}(S, \mathbb{P}, x_0, \lambda) \leq \mathbb{E}_{\mathbb{Q}}[S(V)] + \lambda \mathbb{KL}(\mathbb{Q}||\mathbb{P}) \tag{29}$$

The interpretation of this result is that the free-energy of a system is minimized when the probability distribution $\mathbb{Q}$ defined by a control law minimizes the cost-to-go for a trajectory while staying as close as possible to the uncontrolled probability distribution $\mathbb{P}$ (*minimal intervention principle*).

Naturally, the question that arises is if there is an optimal control distribution $\mathbb{Q}^*$ in the sense that it achieves the lower bound. The answer is positive and the resulting optimal distribution is presented in [14] in the form of the Radon-Nikodym derivative:

$$\frac{d\mathbb{Q}^*}{d\mathbb{P}} = \frac{exp(-\frac{1}{\lambda}S(V))}{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(V))]} \tag{30}$$

We will shortly use the bound in free energy as a lower bound and the form of the optimal distribution for the minimization problem.

So far $\mathbb{P}$, $\mathbb{Q}$ and S are arbitrary. We proceed to apply the above results in the continuous stochastic control-affine system regime and establish a connection with Path Integral Control. The involved probability distributions are:

$$\mathbb{P} : dx = f(x_t, t)dt + B(x_t, t)dw. \tag{31}$$
$$\mathbb{Q}_{U,\Sigma} : dx = [f(x_t, t)dt + G(x_t, t)u_t]dt + B(x_t, t)dw. \tag{32}$$

In the above equations, U is a mapping from time to control inputs and $\Sigma = B(x_t, t)B(x_t, t)^T$ is the covariance matrix. We define the cost-to-go of a trajectory τ as:

$$S(\tau) = \phi(x_T) + \int_0^T q(x_t, t)dt. \tag{33}$$

The control cost is omitted here as it will appear naturally through the KL-Divergence. By doing the computation (KL-Divergence and expectation) the free-energy inequality becomes:

$$\mathcal{F}(S, \mathbb{P}, x_0, \lambda) \leq \min_u \mathbb{E}_{\mathbb{Q}}\left[S(V) + \frac{\lambda}{2}\int_0^T u_t^T G(x_t, T)^T \Sigma(x_t, t)^{-1} G^T u_t dt\right]. \tag{34}$$

The result on the right-hand side is exactly the cost function used in path integral control. What is remarkable is that the assumption made in the Path Integral Framework (between noise and control, B, R and G) now appears naturally on its own. Now, this right-hand term can be minimized using the free-energy as a lower bound and utilizing the definition of the optimal distribution (eq. 30). The problem is formulated as: minimize u so the controlled distribution $\mathbb{Q}_{U,\Sigma}$ is as close as possible to the optimal distribution $\mathbb{Q}^*$. Mathematically:

$$U^* = \underset{U}{\operatorname{argmin}}\, \mathbb{KL}(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}) \tag{35}$$

Again the full derivation can be found in [14]. As it is proved in this work, the result is the same as the Path Integral formulation but valid for all time steps (optimal control **sequence**). It is not presented here since it's not directly used in the algorithm we are interested in. Instead we will focus

on the solution in the case of discrete systems, although, before we proceed with that, it is worth mentioning some interesting insights of what that solution represents.

Remark The optimal controlled distribution solution in the information theoretic framework is found by minimizing the KL-Divergence between the controlled and the optimal distribution. This means that samples drawn from the controlled distribution $\mathbb{Q}$ will have the same mean with the samples drawn from $\mathbb{Q}^*$. This is important since in the case of a uni-modal optimal distribution, the samples drawn from $\mathbb{Q}$ will look just like samples from $\mathbb{Q}^*$, which achieve lower cost than samples drawn from any other distribution [14]. Moreover, in the case of a multi-modal distribution we can count on the Path Integral guarantees (since the Path Integral formulation yields the same solution but for u_0^*) which reassure as that there will be **symmetry breaking**. The notion of symmetry breaking is important and it will be further discussed in the next section. So by combining both frameworks we conclude that what we need to do to compute the optimal control is to be close to the optimal distribution (maintain an estimate) and at each time step execute the mean of it.

2.3.1 Application to Discrete Time Systems

The information theoretic framework can be applied to discrete time systems **without even assuming control affine dynamics**.

A general stochastic discrete system can be described as:

$$x_{t+1} = F(x_t, v_t) \tag{36}$$

We assume that a control input u_t is applied to the system via noise addition:

$$v_t \sim \mathcal{N}(u_t, \Sigma) \tag{37}$$

This noise assumption is reasonable for realistic robot systems where the control input has to pass through a lower level controller. Especially this is the case in steering and throttle for a car which is the application of interest.

The control sequence input and the noisy actual input sequence to the system over T timesteps are defined as:

$$U = (u_0, u_1, .., u_{T-1}), \tag{38}$$
$$V = (\nu_0, \nu_1, .., \nu_{T-1}). \tag{39}$$

The advantage we have in the discrete case is that we can explicitly define probability density functions. So, for the uncontrolled distribution $\mathbb{P}$ and the controlled $\mathbb{Q}_{\mathbb{U},\mathcal{K}}$ the pdfs are:

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{\frac{1}{2}}} exp\left(-\frac{1}{2}\nu_t^T \Sigma^{-1} \nu_t\right). \tag{40}$$

$$q(V|U,\Sigma) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m |\Sigma|)^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\nu_t^T - u_t)\Sigma^{-1}(\nu_t - u_t)\right). \tag{41}$$

The cost-to-go state dependent function for a trajectory in the discrete case is defined as:

$$C(x_1, x_2, ..., x_T) = \phi(x_T) + \sum_{t=1}^{T-1} q(x_t). \tag{42}$$

Next, we need the KL Divergence between the controlled and uncontrolled distribution. In discrete time this is much easier and we can simply compute:

$$\mathbb{KL}(\mathbb{Q}_{U,\Sigma}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}\left[log\left(\frac{q(v|U,\Sigma)}{V}\right)\right] = \frac{1}{2}\sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t. \tag{43}$$

Then, the free-energy lower bound takes the form:

$$\mathcal{F}(S,\mathbb{P},x_0,\lambda) \leq \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[S(V,x_0)] + \frac{\lambda}{2}\sum_{t=0}^{T-1} u_t^T \Sigma^{-1} u_t. \tag{44}$$

We perform the same analysis as before: the optimal distribution acheives the lower bound and we want to push the controlled distribution as close as possible to the optimal one. The KL divergence between the two is defined as:

$$\mathbb{KL}(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}) = \int_{\Omega_V} q^*(V) log\left(\frac{q^*(V)}{q(V|U,\Sigma)}\right) dV, \tag{45}$$

$$= \int_{\Omega_V} q^*(V) log\left(\frac{q^*(V)}{p(V)}\frac{p(V)}{q(V|U,\Sigma)}\right) dV, \tag{46}$$

$$= \int_{\Omega_V} q^*(V) log\left(\frac{q^*(V)}{p(V)}\right) - q^*(V) log\left(\frac{q(V|U,\Sigma)}{p(V)}\right) dV. \tag{47}$$

The first term is independent of U so for the optimization we consider only the second term. This leaves us with:

$$U^* = \underset{U}{\operatorname{argmax}} \int_{\Omega_V} q^*(V) log\left(\frac{q(V|U,\Sigma)}{p(V)}\right) dV. \tag{48}$$

The fraction inside the logarithm can be written as:

$$\frac{q(V|U,\Sigma)}{p(V)} = exp\left(\sum_{t=0}^{T-1} -\frac{1}{2}u_t^T\Sigma^{-1}u_t + u_t^T\Sigma^{-1}\nu_t\right). \tag{49}$$

Plugging that into the previous equation yields:

$$U^* = \underset{U}{\operatorname{argmax}} \int_{\Omega_V} q^*(V)\left(\sum_{t=0}^{T-1} -\frac{1}{2}u_t^T\Sigma^{-1}u_t + u_t^T\Sigma^{-1}\nu_t\right) dV. \tag{50}$$

The probability can be integrated out in the first term (since its constant with respect to the integral), so we get:

$$U^* = \underset{U}{\operatorname{argmax}} \left[\sum_{t=0}^{T-1}\left(-\frac{1}{2}u_t^T\Sigma^{-1}u_t + u_t^T\int_{\Omega_V} q^*(V)\Sigma^{-1}\nu_t\right)\right] dV. \tag{51}$$

The resulting equation is concave with respect to u so we can find the maximum by setting the gradient to zero. This results in:

$$u_t^* = \int q^*(V)\nu_t dV \tag{52}$$

This states (as before) that **the optimal control, in the sense of minimizing the KL-Divergence, is the mean of optimal distribution**. In this case, there is not a direct analogy between the Information

Theoretic standpoint and Path Integral Theory. This means that the guarantee of symmetry collapse does not hold here. However, in practice the authors of [12] showed that the result cam be successfully applied to a general stochastic discrete system (even with highly multi-modal cost landscapes).

We note that a connection to Path Integral formulation can be made with the assumption that the discrete non-linear dynamical system can be approximated by a control-affine system. This idea is developed in detail in [14].

3 The Algorithm: MPPI

The two different approaches used resulted in the same solution: computing a cost-weighted average over trajectories. In the Path Integral Theory, this average is interpreted as the optimal control input for the current state and time. In the Information Theoretic Framework, the average is the best approximation of the optimal distribution parameterized by an open loop control sequence. Utilizing both interpretations gives us a strong theoretical background and helps understand and clarify some emergent phenomena.

We will use the example shown in figure 3. The goal is to avoid the obstacle (shown in orange) and there is an apparent symmetry (over the trajectories that successfully establish the goal). From the Information Theory perspective, the optimal control sequence will result in a trajectory that goes through the obstacle. That is because the optimal distribution is multi-modal and thus sampling around a control sequence that goes directly into the obstacle will result in half of the trajectories avoiding the obstacle by moving right and half by moving left. But taking the mean, as this approach dictates, does not solve the problem correctly (again, due to the multi-modal nature of the problem). On the other hand, Path Integral Theory will compute an optimal control input (for the current state and time) which again directs the car to move directly towards the obstacle. The idea here is that since there exists noise in the system the optimal action is not to waste energy now (make a decision) but just move straight to the obstacle and wait for the noise to break the symmetry. This symmetry collapse is a feature of path integral control. Moreover, the more the noise variance in the system, the more the delay in the choice (*the delayed choice* [5]). So, by combining both interpretations we can plan a control input **sequence** by taking the mean of the optimal distribution at time $t = 0$ even if there exists symmetry. Path Integral guarantees that this symmetry will eventually break. As a result the combination of the two frameworks guarantees, reassures us that the control input (cost-weighted average) will be successful both in the case of uni-modal and multi-modal distributions.

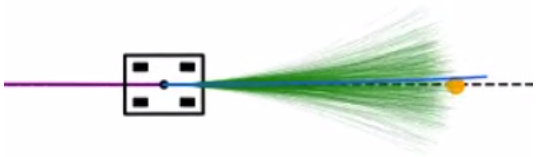


Figure 1: Symmetry Breaking

For the development of the algorithm we will assume that the system takes the general form:

$$x_{t+1} = x_t + F(x_t, v_t)\Delta t, \tag{53}$$
$$v_t \sim \mathcal{N}(u_t, \Sigma) \tag{54}$$

The above formulation is general enough for our purposes (even continuous systems need to be approximated with discrete systems for Monte-Carlo sampling). There is no control-affine dynamics assumption, so we proceed with the guarantees of the Information Theoretic framework.

We continue from the last equation derived in the previous section, which computes the optimal control sequence (eq. 52). Since we don't have knowledge of the optimal distribution, we can use $q^*(V) = q(V)\frac{q^*(V)}{p(V)}\frac{p(V)}{q(V)}$ as an importance sample technique to re-write the equation in the form:

$$u_t^* = \int q(V)\frac{q^*(V)}{p(V)}\frac{p(V)}{q(V)}\nu_t dV. \tag{55}$$

Then, the optimal control can be written as an expectation with respect to the controlled distribution:

$$u_t^* = \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[w(V)\nu_t], \tag{56}$$

$$w(V) = \frac{q^*(V)}{p(V)}exp\left(\sum_{t=0}^{T-1} -\frac{1}{2}\nu_t^T\Sigma^{-1}u_t + u_t^T\Sigma^{-1}u_t\right), \tag{57}$$

$$= \frac{1}{\eta}exp\left(-\frac{1}{\lambda}S(V) + \sum_{t=0}^{T-1} -\frac{1}{2}\nu_t^T\Sigma^{-1}u_t + u_t^T\Sigma^{-1}u_t\right), \tag{58}$$

where $\eta = \mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(V)]$. We note that the fraction $\frac{q^*(V)}{p(V)}$ takes the above form because of the result shown in equation 12, which describes the optimal distribution (in the information theoretic framework).

We make a change of variables $u_t + \epsilon_t = \nu_t$ and denote the noise sequence as $\mathcal{E} = (\epsilon_0, .., \epsilon_{T-1})$. Then $w(\mathcal{E})$ is defined as:

$$w(\mathcal{E}) = \frac{1}{\eta}exp\left(-\frac{1}{\lambda}\left(S(U+\mathcal{E}) + \lambda\sum_{t=0}^{T-1}\frac{1}{2}u_t^T\Sigma^{-1}(u_t + 2\epsilon_t)\right)\right). \tag{59}$$

Performing a trick to decouple the temperature parameter (λ) from the control costs ([14]) yields:

$$w(\mathcal{E}) = \frac{1}{\eta}exp\left(-\frac{1}{\lambda}\left(S(U+\mathcal{E}) + \lambda(1-\alpha)\sum_{t=0}^{T-1}\frac{1}{2}u_t^T\Sigma^{-1}(u_t + 2\epsilon_t)\right)\right). \tag{60}$$

The normalization term η can be approximated using the Monte-Carlo estimate:

$$\eta = \sum_{n=1}^{N} exp\left(-\frac{1}{\lambda}\left(S(U+\mathcal{E}_n) + \lambda\sum_{t=0}^{T-1}\frac{1}{2}u_t^T\Sigma^{-1}(u_t + 2\epsilon_t^n)\right)\right), \tag{61}$$

with each of the N samples drawn from the system with U as the control input sequence. The resulting iterative update law is:

$$u_t^{i+1} = u_t^i + \sum_{n=1}^{N} w(\mathcal{E}_n)\epsilon_t^n \tag{62}$$

The result of this optimization is that the expectation of inputs sampled from the controlled distribution is the same as the expectation of the inputs sampled from the optimal distribution, ($\mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[v_t] = \mathbb{E}_{\mathbb{Q}^*}[v_t]$). As discussed, under the control-affine assumptions, this is equivalent to computing the optimal control.

The iterative procedure (eq. 62) exists to improve the Monte-Carlo estimate by using a more accurate importance sampler (more samples). If we could compute the expectation with zero error, then we could minimize the KL-Divergence between the controlled and optimal distribution in a single step. Notice that the final algorithm tries to iteratively improve the importance sampler by increasing the number of samples. It does not use any idea similar to the Cross-Entropy method for Path Integral [1] which was briefly discussed in the previous section (so the idea here is not to improve u to sample

less but iteratively increase the samples and be as close as possible to the true full expectation which yields the optimal u).

We note that as this is a model-based algorithm, so the computations of the optimal controls requires knowledge of Σ, which can be obtained in two ways. First, the trajectories can be generated purely in simulation, where the noise is generated from a random number generator. Second, trajectories could be generated by a real system, and the noise would be computed from the difference between the actual and the predicted system behavior. In the literature [14] there are methods to add a term that enhances the natural system's noise (maybe to achieve more diverse trajectories).

The final algorithm can be found below:

Algorithm 2: MPPI

Given: $\mathbf{F}$: Transition Model;
K: Number of samples;
T: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, ...\mathbf{u}_{T-1})$: Initial control sequence;
Σ, ϕ, q, λ: Control hyper-parameters;
while *task not completed* **do**
  $\mathbf{x}_0 \leftarrow$ GetStateEstimate();
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} \leftarrow \mathbf{x}_0$;
    Sample $\mathcal{E}^k = \{\epsilon_0^k, \epsilon_1^k, \dots \epsilon_{T-1}^k\}$;
    for $t \leftarrow 1$ **to** T **do**
      $\mathbf{x}_t \leftarrow \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1} + \epsilon_{t-1}^k)$;
      $S(\mathcal{E}^k) \mathrel{+}= \mathbf{q}(\mathbf{x}_t) + \lambda \mathbf{u}_{t-1}^{\mathrm{T}} \Sigma^{-1} \epsilon_{t-1}^k$;
    $S(\mathcal{E}^k) \mathrel{+}= \phi(\mathbf{x}_T)$;
  $\beta \leftarrow \min_k [S(\mathcal{E}^k)]$;
  $\eta \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}(S(\mathcal{E}^k) - \beta)\right)$;
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $w(\mathcal{E}^k) \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(S(\mathcal{E}^k) - \beta)\right)$;
  for $t \leftarrow 0$ **to** $T-1$ **do**
    $\mathbf{u}_t \mathrel{+}= \sum_{k=1}^{K} w(\mathcal{E}^k) \epsilon_t^k$;
  SendToActuators($\mathbf{u}_0$);
  for $t \leftarrow 1$ **to** $T-1$ **do**
    $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t$;
  $\mathbf{u}_{T-1} \leftarrow$ Intialize($\mathbf{u}_{T-1}$);

Figure 2: The Algorithm [11]

Learned Dynamics The authors of [11] give a choice of completely learned dynamical model from data using a neural network. They create a dataset in two phases. In the first phase, they collect system identification data and then train the neural network. For simulation data, the MPPI controller with the ground truth model is run, whereas a human driver is used in real world experiments. They show that the algorithm under the learned model perform succesfully. The algorithmic scheme is shown in figure 3.

4 Application to Autonomous Driving

We proceed with the analysis and evaluation of MPPI in an autonomous driving context. As discussed in the introduction, the algorithm was specifically developed for such applications, particularly in the realm of aggressive autonomous driving. Before we describe the setup of the particular application (environment, state, dynamics), we provide an overview of a typical autonomous driving pipeline (a standard in the Formula Student contest). The pipeline is depicted in figure 4. We note that this is the pipeline used when **the track is unknown**.

Algorithm 1: MPPI with Neural Network Training

Input: Task, N: Iterations, M: Trials per iteration
$\mathcal{D} \leftarrow$ CollectBootstrapData();
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{F} \leftarrow$ Train($\mathcal{D}$);
  for $j \leftarrow 0$ **to** M **do**
    $\mathcal{D}_j \leftarrow$ MPPI($\mathbf{F}$, Task) ;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j$

Figure 3: Model Learning [11]

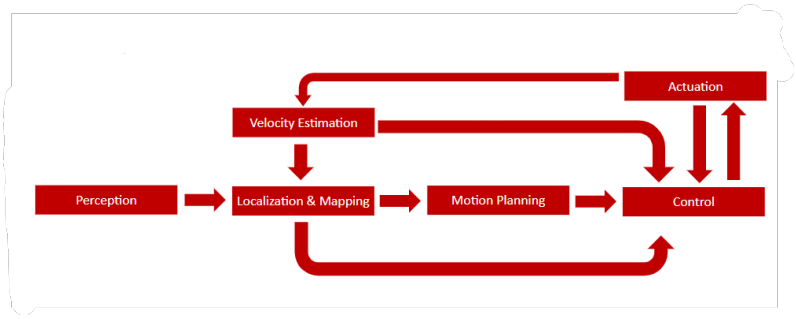


Figure 4: Autnomous Driving Pipeline

The perception module (camera or lidar) receives an input image from the environment and extracts features of interest (cone locations). These features, along with odometry measurements (relative position) filtered by the Velocity Estimation Module (utilizing a Kalman Filter), are passed to the Localization & Mapping Module. The objective of this module is to keep track of the vehicle's position inside a map constructed progressively as the vehicle explores the unknown track. Once the local map is generated, it is passed to the Path Planner, responsible for generating a path in the form of a sequence of points that the vehicle is required to follow. Finally, the Controller (PID, MPC) translates the produced path into an input control sequence (which "best achieves the path"). This sequence is further translated into low-level actuations. As discussed in the Introduction, in the described pipeline the Controller and Path Planner are decoupled. After the first round a complete map of the track has been constructed. Because of the large noise present in the system, designing a trajectory to be used as a reference path (input to the controls) is not the optimal solution if aiming for peak performance. The solution proposed here is utilizing MPPI for planning and control and hence integrating the Control and Path Planning Module, ultimately reaching a higher performance.

We note that the way the algorithm is implemented in the experiments (which will be described in detail), makes it potentially applicable also in the initial round, while the global track map is under construction. The idea is that MPPI considers the closest cones (local map) to generate a trajectory. Thus, if the time horizon aligns with the range of the perception module, the algorithm could be utilized in the initial round as well.

In the subsequent subsections, we present in detail the setup used to test the algorithm, highlighting its capability to achieve high performance while remaining within the track boundaries. Additionally, we conduct several experiments investigating the effect of the various hyperparameters of MPPI, aiming to gain a thorough understanding of the proposed algorithm.

4.1 Setup

In the conducted experiments, the environment (track) is known and represented by the (x, y) coordinates of the cones with respect to a global frame. At each timestep t, the vehicle is in a **state** s which is represented by a vector containing measurements of its position (with respect to the global

frame), velocity and orientation. The quantities in the state vector are dependent of the dynamic model that is chosen for the vehicle. The **control** vector is 2-dimensional where the first dimension is the steering angle and the second the acceleration (throttle). The majority of experiments are carried out on an oval map, as depicted in figure 32, due to its simplicity (the middle line is drawn too). However, two additional tracks (figures 33 and 29) with more complexity are also employed to demonstrate the effectiveness of MPPI, showcasing that the algorithm exhibits independence from the specific shape of the track.

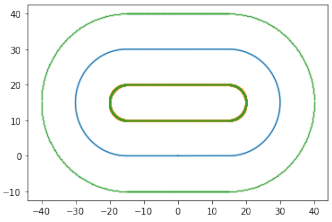


Figure 5: Track 1

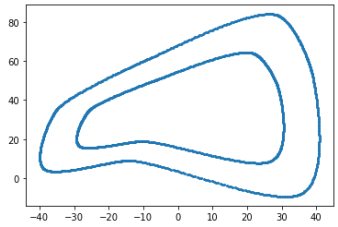


Figure 6: Track 2

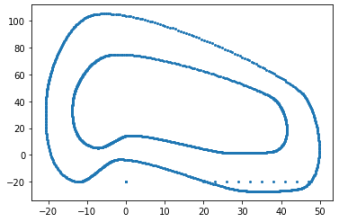


Figure 7: Track 3

4.2 Vehicle Dynamics

In this context, we do not extensively explore vehicle dynamics, as this falls within the scope of a separate study. We assume that a model close to reality is provided (with any mismatches considered as a noise component). The model used is the Dynamic Bicycle Model (figure 8). This is a famous model for autonomous driving applications and lies in between highly complex accurate models and simpler ones (like Kinematic Bicycle Model). The Dynamic Bicycle Model makes some assumptions about the underlying dynamics. First, the wheels of the front and rear axle are bound together into one front and one rear wheel. Reference point C is the center of gravity. For the lateral and yaw motion, the vehicle is considered as a rigid body whose dynamics are determined by the fundamental laws of motion. The tires receive only lateral forces (perpendicular to the respective tire center line) generated by a linear tire model. Finally, the steering angle is small (assumption for sine and cosine). The vehicle's mass is denoted by m. I_z is the moment of inertia about z-axis. C_f and C_r are parameters of the tire dynamics and denoted the cornering stiffness of the combined front and rear tires. W point out again that all these parameters are considered to be known and in practise they are estimated experimentally (Vehicle Dynamics subgroups).

This dynamic model results in a 6-dimensional state vector (8) which is used by the algorithm. The control vector consists of the steering angle δ and acceleration a:

$$u(x_t, t) = \begin{bmatrix} \delta \\ a \end{bmatrix} \tag{63}$$

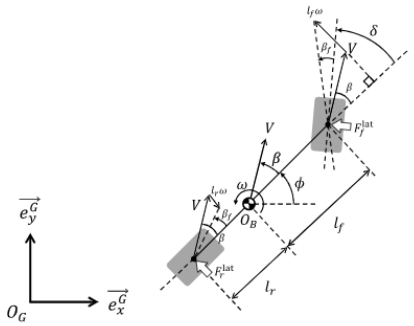
4.3 Cost Function

A crucial component of the MPPI algorithm is the state dependent cost function. That is because all the objectives we aim to achieve are expressed in the form of costs and integrated into the cost-to-go function. So this function must be designed carefully and by taking into consideration the desired goals. Here the primary objective is high performance while ensuring the vehicle remains within the track boundaries. We explored the advantages of the ability to incorporate discontinuous cost functions in the introduction. We can use large weighted indicator functions penalize instances of hitting the cones or slipping. In this work, we choose a cost function similar to [11]. Specifically:

$$C(x_t) = w_1 C_{track} + w_2 C_{speed} + w_3 C_{slip} \tag{64}$$

$$\phi(x_T) = 100000C. \tag{65}$$

where $C = \mathbb{1}_{crash}$. C_{track} is a function which grows larger as the distance to the closest cone is decreased, $C_{speed} = (v - v_{des})^2$ is the distance from the desired speed and C_{slip} is a function of



$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} p_x^G \\ p_y^G \\ \phi \\ v_x^B \\ v_y^B \\ \omega \end{bmatrix} = \begin{bmatrix} v_x^B\cos\phi - v_y^B\sin\phi \\ v_x^B\sin\phi + v_y^B\cos\phi \\ \omega \\ a\cos\beta - (F_f^{\text{lat}}\sin\delta)/m + v_y^B\omega \\ a\sin\beta + F_r^{\text{lat}}/m + F_f^{\text{lat}}\cos\delta/m - v_x^B\omega \\ (F_f^{\text{lat}}l_f\cos\delta - F_r^{\text{lat}}l_r)/I_z \end{bmatrix},$$

$$F_f^{\text{lat}} = -C_f\left(\frac{v_y^B + l_f\omega}{v_x^B} - \delta\right),$$

$$F_r^{\text{lat}} = -C_r\left(\frac{v_y^B - l_r\omega}{v_x^B}\right),$$

$$\beta = \tan^{-1}\left(\frac{v_y^B}{v_x^B}\right) \approx \frac{v_y^B}{v_x^B} \quad (\because v_y^B \ll v_x^B).$$

Figure 8: Dynamic Bicycle Model

the slip angle defined as $-arctan(\frac{v_y}{|v_x|})$, where v_x and v_y are the velocities shown in 8. In this slip function, we incorporate an indicator variable which activates if the magnitude of the slip angle exceeds a certain threshold (0.75). For the cost function to operate as intended, it requires careful tuning of weights w_1, w_2, w_3. Moreover, a quadratic control cost C_{cost} exists inside the equations of the algorithm, as discussed before, and is controlled by parameter α.

At each time step, the current observed state is passed as input. The algorithm utilizes the previously calculated control sequence (warm-start) and the received current state to estimate an updated control sequence, outputting only the first time step (first action). To evaluate each generated trajectory, MPPI utilizes the current state measurements and the global map. Specifically, the global map is sorted, and the algorithm employs an efficient binary search logic to find the closest cone to the vehicle with respect to its current position (the closest cone is needed for C_{track} and $\phi(x_T)$). By utilizing this cost function, the vehicle can successfully avoid obstacles (cones) present in the track while maintaining high speed.

4.4 Experiments

Since we described the setup, the maps and the form of the state and the control of the algorithm we continue to the experiments. The experiments showcase two key aspects: firstly, the algorithm's capability to navigate through the track at a high speed (tested in all maps), and secondly, its ability to avoid obstacles present in the path. While achieving these objectives, the algorithm takes some decisions in the form of the calculated optimal trajectory. We investigate how this trajectory changes with variations in some hyperparameters. Specifically, we vary the time horizon T, the variance of the noise Σ, the control cost α, and the number of samples (trajectories) N. Additionally, we examine the algorithm's performance (once again concerning these hyperparameters) when the dynamic model is highly deviating from the true one.

Videos illustrating various aspects of the ensuing analysis are provided (videos). Each experiment accompanied by a video will be explicitly referenced, allowing the reader to view the corresponding visual content.

4.5 The Delayed Case

Before we proceed to the practical experiments investigating the role of the different algorithm's parameters, we conduct an initial experiment of theoretical nature. We discussed in the previous section, the notion of the delayed choice. In particular, the property of Path Integral Control to delay a control choice because of the natural variance of the system. Moreover, as argued in [5], as the variance of the system noise increases, the choice gets more delayed. This can be seen experimentally in figure 4.5,where an obstacle is present in the vehicle's path, and the vehicle adjusts to avoid it (makes a decision) later when the noise variance is larger.

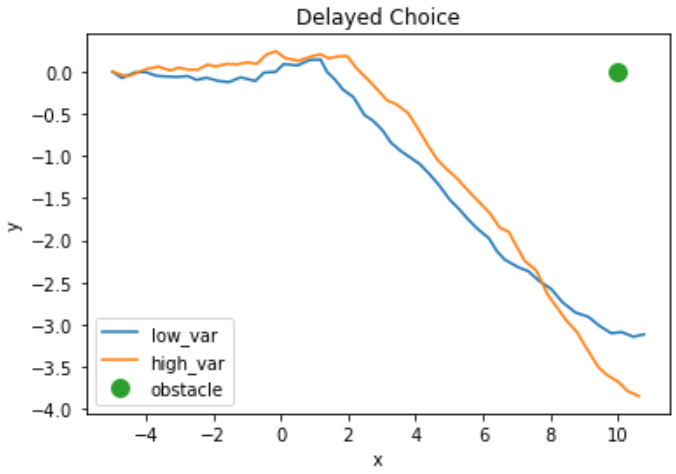


Figure 9: The Delayed Choice

4.5.1 Effect of T (time horizon)

We first investigate the algorithm's planning range concerning the time horizon T. This is shown in figures 18 and 19. In these plots the actual position of the vehicle is $(x, y) = (-15, 0)$, the velocity is $v_x = 7\frac{m}{s}$ and the control sequence for warm start consists of zeroes. By utilizing a larger time horizon we expect better planning which leads to higher performance. Indeed, this is the case. As shown in figure 12, the vehicle achieves a significally better lap time when the time horizon is larger. The reason behind this (which can be clearly seen in videos "mppi_test_simple_T_40.mp4" and "mppi_test_simple_T_80.mp4") is that when $T = 80$ the algorithm "sees" the corner and the boundaries in advance and can thus plan better (it is apparent that the velocity is closer to the desired one especially in the corners). The advantage of a larger T is further seen in videos "mppi_test_map2.mp4" and "mppi_test_map2_large_T.mp4", where in the case of a smaller T the algorithm struggles to accurately comprehend the track's structure, leading to wrong decisions and eventual failure (vehicle out of boundaries). This is shown in figures 4.5.1.

Further investigation on the effect of T yields even more interesting results. First, better planning is expected based on the planning distance plots. Indeed, as shown in videos "mppi_test_100_160.mp4" and "mppi_test_100_160_larger_T.mp4", when there are multiple obstacles and the algorithm needs to plan in advance an efficient trajectory to avoid them, utilizing a suitable value for T is vital for success. What is more surprising is that even in the case of a single obstacle a larger T offer a more efficient trajectory (videos "mppi_test_280.mp4" and "mppi_test_280_larger_T.mp4").

The conclusion drawn is that larger values of T contribute to a superior overall performance. However, this entails a trade-off due to the fact that the computations executed by the algorithm increase proportionally with $T \times N$, where N is the number of sampled trajectories.

4.5.2 Effect of Σ (variance in the control input)

Here, we investigate the role of Σ variance of the nosie added to the control. As discussed in the section that introduced the algorithm the idea is that Σ is a parameter of the stochastic system, so it needs to be computed. What we do here is to practically test different values of Σ to examine the quality

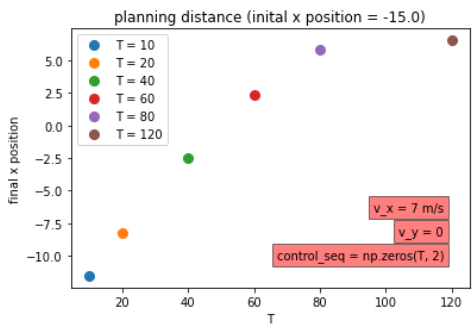


Figure 10: Planning Distance(x) as T increases

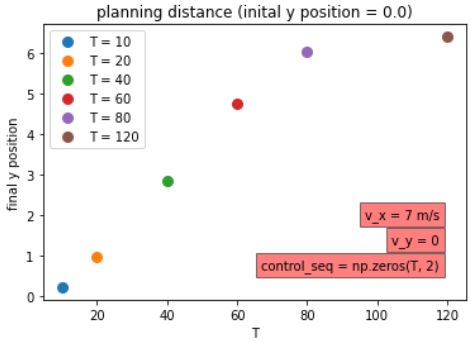


Figure 11: Planning Distance(y) as T increases

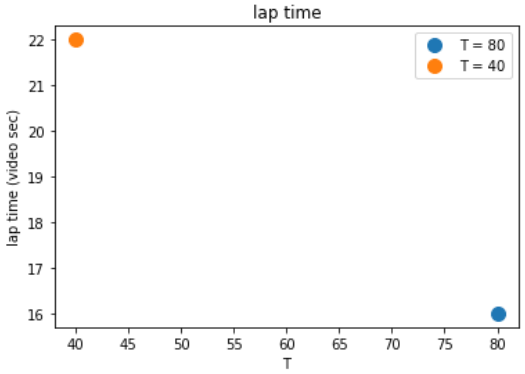


Figure 12: Lap Time and T

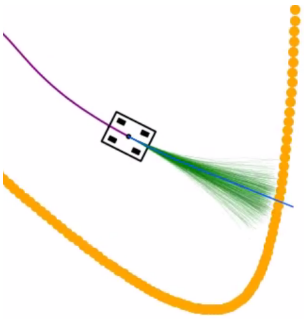


Figure 13: Small T frame 1

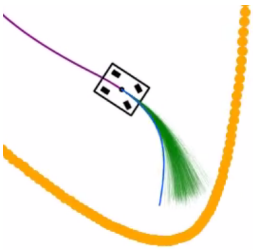


Figure 14: Small T frame 2 (wrong decision)

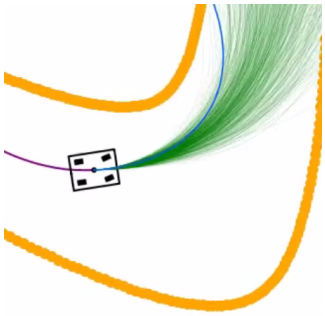


Figure 15: Large T (correct decision)

of the produced trajectories. The results can be seen in videos "mppi_test_280_small_sigma.mp4", "mppi_test_280_larger_sigma.mp4" and "mppi_test_280_larger_sigma_2.mp4". Furthermore, we plot the planning distance with respect to different values of Σ for steering and acceleration (figures 4.5.2 and 4.5.2).

4.5.3 Effect of α (control Cost)

Next, we investigate the role of α which is the parameter tuning the input control cost. We use the oval track with obstacles at $(0, 30)$ and $(30, 12)$. What we observe is that the higher the value of α the more the input control is penalized so we don't observe large variations in the vehicle's trajectory. This is apparent in figure 4.5.3 where higher cost results in a smoother trajectory. On the other hand,

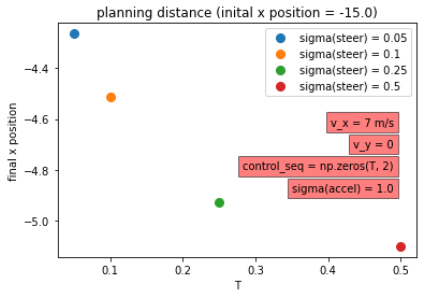


Figure 16: Planning Distance(x) as as $\Sigma(steer)$ increases

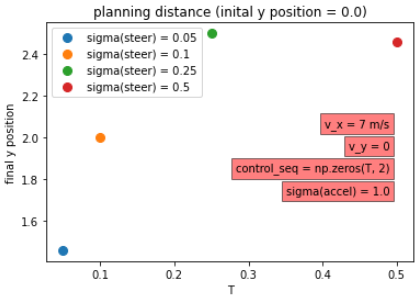


Figure 17: Planning Distance(y) as $\Sigma(steer)$ increases

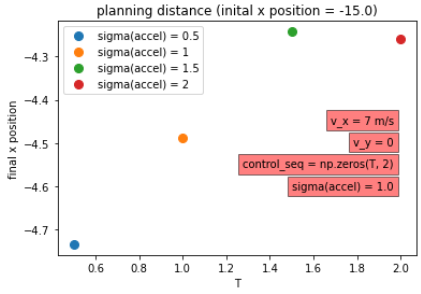


Figure 18: Planning Distance(x) as $\Sigma(accel)$ increases

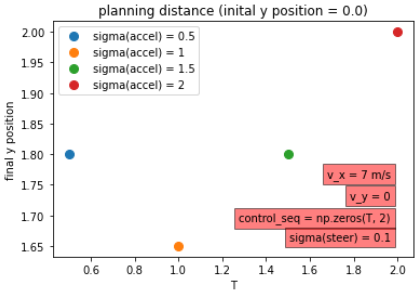


Figure 19: Planning Distance(y) as $\Sigma(accel)$ increases

the derivation from the desired velocity is higher. As we expected, the control variables take lower values 4.5.3.

Subsequently, we examine the impact of α, the parameter tuning the input control cost. A higher value of α implies a more significant penalty on the input control, leading to less variations in the vehicle's trajectory. This is evident in figure 4.5.3, where a higher cost results in a smoother trajectory. However, the deviation from the desired velocity is greater. As anticipated, the control variables take lower values, as depicted in figure 4.5.3.

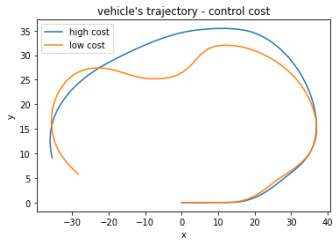


Figure 20: Vehicle's Trajectory

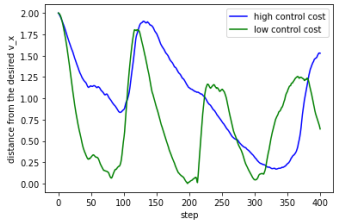


Figure 21: Distance from desired velocity

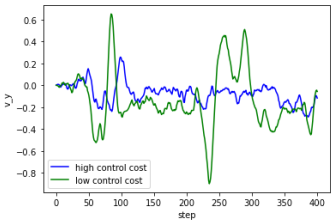


Figure 22: v_y

4.5.4 Effect of Noise in the Dynamics

Here, we tested the algorithm's ability in practice to generate controls that result in efficient trajectories. The idea is to highly corrupt the dynamical model of MPPI (in comparison with the true dynamical model) and use different values of Σ. So the state equation takes the form:

$$x_{t+1} = F(x_t, v_t) + w_t, \tag{66}$$

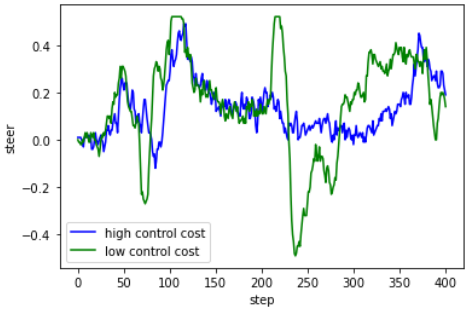


Figure 23: Steer values as a function of control cost

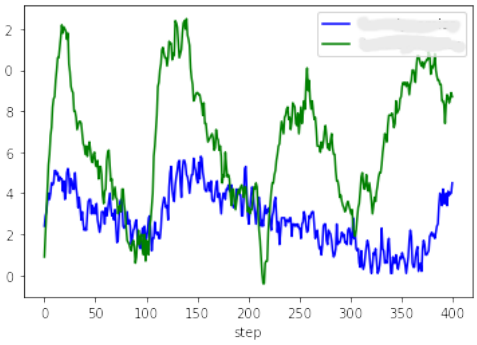


Figure 24: Accel variations as a function of control cost

where the modelling errors are represented as noise disturbance. There has been developed variants of MPPI that explicitly address this problem [13]. What we observe is that for low values of system variance Σ the algorithm behaves similar to when it has access to the true dynamics. When $Sigma$ is large, the behaviour is significantly different. The oval track is used, T is set to 40 and N to 500. There are obstacles at $(0, 30)$ and $(30, 12)$. Figure 4.5.4 shows the result for $\Sigma(steer) = 0.05$ and $\Sigma(accel) = 1$, while figure 4.5.4 shows the result for $\Sigma(steer) = 0.5$ and $\Sigma(accel) = 2$.

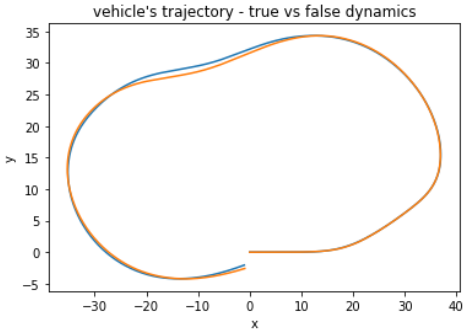


Figure 25: Vehicle's Trajectory

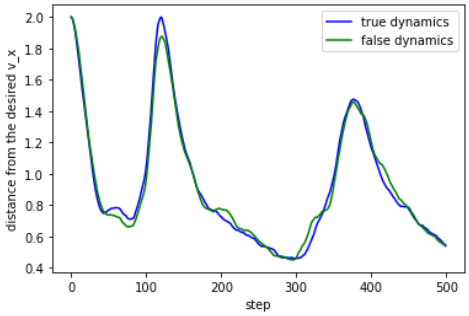


Figure 26: Distance from desired v_x

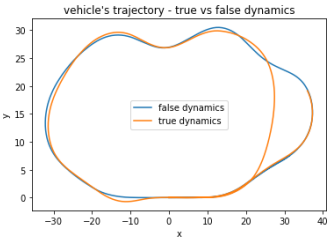


Figure 27: Vehicle's Trajectory

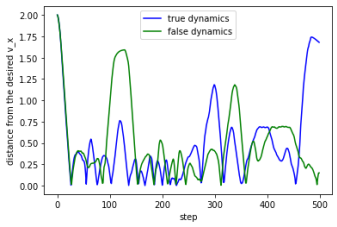


Figure 28: v_x

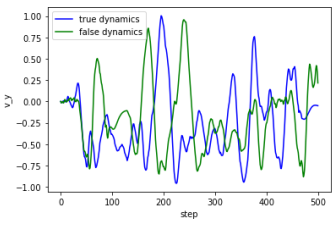


Figure 29: v_y

4.5.5 Effect of N (number of samples)

In this group of experiments, we analyze the effect of N (number of samples). We set $T = 80$, we use the oval map with obstacles at $(0, 30)$ and $(30, 12)$. Our theoretical results suggest that higher values of N will improve the performance (closer to the optimal control - true expectation). The experiments validate this assumption. As shown in figure 4.5.5 for $N = 30$ we observe a complete different behaviour (in the sense of the trajectory) compared to the behaviour for larger values of N. For higher values of N, the differences become less pronounced, as illustrated in figure 4.5.5 by the distance from the desired velocity.

Again we highlight the induced trade-off arising from the computations needed ($O(T \times N)$).

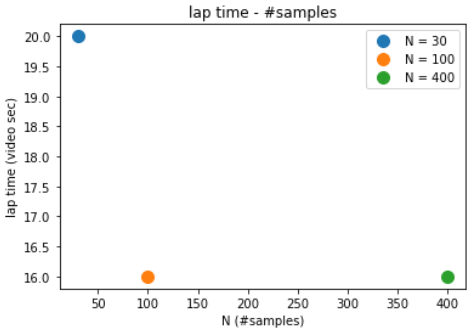


Figure 30: Lap Time

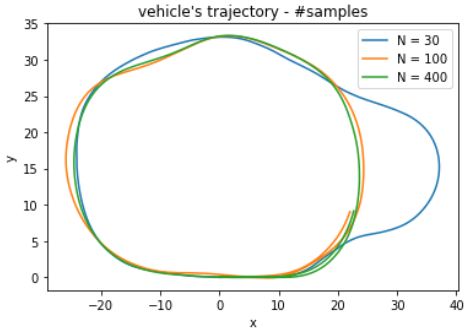


Figure 31: Vehicle's trajectory

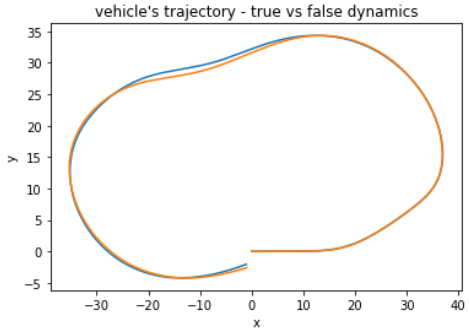


Figure 32: Vehicle's Trajectory

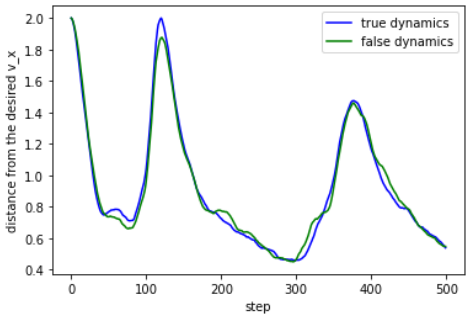


Figure 33: Distance from desired v_x

5 Conclusion

In this work, we studied practically and theoretically the Model Predictive Path Integral (MPPI) Algorithm. From the theoretical analysis it became apparent that depending on the chosen dynamical model different theoretical guarantees are induced, while the complete picture emerges when you combine the Path Integral and Information Theoretic frameworks. In practice, the algorithm works for general discrete systems (without the control-affine) assumptions. From the experiments conducted several conclusions were made. First, it is highly important to choose an appropriate cost function depending on the objectives and tune the weights by simulation. Next, choosing a large T is crucial for achieving high performance but the practitioner must be aware of the discussed trade-off (computations $O(T * N)$). Moreover, the system noise variance (which is applied to the control) needs to be identified for a solid performance (in agreement with the theoretical analysis) and can be enhanced if needed (again this idea is explored in [14]). Another vital parameter is the number of samples N. As demonstrated by the experiments, higher values contribute to improved performance, but the computational trade-off persists in this case as well. Finally, to be able to utilize this algorithm in a real setting a GPU implementation is needed to parallelize the online computations performed by the algorithm.

References

[1] "Adaptive importance sampling for control and inference". In: *Journal of Statistical Physics* 162 (2016), pp. 1244–1266.

[2] Pieter-Tjerk De Boer et al. "A tutorial on the cross-entropy method". In: *Annals of operations research* 134 (2005), pp. 19–67.

[3] Brian Goldfain et al. "Autorally: An open platform for aggressive autonomous driving". In: *IEEE Control Systems Magazine* 39.1 (2019), pp. 26–55.

[4] Muhammad Kazim et al. "Recent advances in path integral control for trajectory optimization: An overview in theoretical and algorithmic perspectives". In: *Annual Reviews in Control* 57 (2024), p. 100931.

[5] "Path integrals and symmetry breaking for optimal control theory". In: *Journal of statistical mechanics: theory and experiment* 2005.11 (2005), P11011.

[6] H.L. Royden and P. Fitzpatrick. *Real Analysis*. Prentice Hall, 2010. ISBN: 9780135113554. URL: https://books.google.gr/books?id=H65bQgAACAAJ.

[7] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 1994.

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

[9] Sep Thijssen and HJ Kappen. "Path integral control and state-dependent feedback". In: *Physical Review E* 91.3 (2015), p. 032104.

[10] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[11] Grady Williams et al. "Information theoretic MPC for model-based reinforcement learning". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1714–1721.

[12] Grady Williams et al. "Information-theoretic model predictive control: Theory and applications to autonomous driving". In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1603–1622.

[13] Grady Williams et al. "Robust Sampling Based Model Predictive Control with Sparse Objective Information." In.

[14] Grady Robert Williams. "Model predictive path integral control: Theoretical foundations and applications to autonomous driving". In: (2019).