Model Predictive Path Integral Control

Application to Autonomous Driving

Vaggelis Dorovatas

NTUA

NeuroFuzzy Control Project

Table of Contents

Table of Contents

Motivation

- Traditional autonomous driving systems decouple Path Planning and Control Modules
- Fine for lane keeping, turning, and parking
- But not enough for when the vehicle performing in its limits
- Crucial application, since executing appropriate aggressive maneuvers is important in avoiding or mitigating collisions

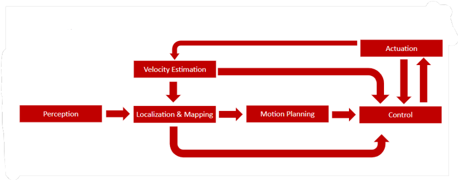


Figure: An Autonomous Driving Pipeline

Approach

- Main limitation of decoupled approach is that the path planner has no knowledge of the underlying vehicle dynamics.
- **Idea**: Integrate Control and Path Planning into one Module
- Now, the problem can be formulated as a stochastic optimal control problem where:
  - we define a *cost function* on the state and control input
  - the goal is to *minimize the total cost* (expected cumulative cost)
  - subject to the *constraints* of the *stochastic* dynamics of the vehicle

Challenges when using Stochastic Optimal Control

- State space can be too high dimensional for global methods like solving the HJB equation
- It involves **non-linear dynamics and non-convex objectives**
- Variations of MPC have been developed but most of them rely on tools from constrained optimization, which means convexification of cost function and approximation of the dynamics

Idea

Develop a sampling-based (MPC-variant) algorithm which can optimize for general non-convex cost criteria and general non-linear dynamics

Table of Contents

Mathematical Formulation

- The problem can be approached from two mathematical theories, yielding the same result (different assumptions)
- Stochastic Optimal Control, in particular, **Path Integral Control Theory**
- **Information Theory**
- They offer distinct but complementary assurances and interpretations of the result
- The performed analysis is inspired by [6] and [2]

Path Integral Control

- First assumption: (Control-affine Dynamics)

$$dx = [f(x_t, t)dt + G(x_t)u(x_t, t)]dt + B(x_t, t)dw. \tag{1}$$

- The stochastic HJB and boundary condition:

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

$$V(x_T, T) = \phi(x_T). \tag{3}$$

- Above equation is quadratic with respect to u:

$$u^* = -R^{-1} G^T \nabla_x V. \tag{4}$$

Path Integral Control

We now perform 3 steps:

- STEP 1

$$-\partial_t V = q + f^T \nabla_x V - \frac{1}{2}\nabla_x V^T G R^{-1} G^T \nabla_x V + \frac{1}{2} tr(BB^T \nabla_{xx} V) \tag{5}$$

$$V(x_T, T) = \phi(x_T). \tag{6}$$

- STEP 2

$$V(x,t) = -\lambda log(\Psi(x,t)) \tag{7}$$

- STEP 3

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

$$B(x,t)B(x,t)^T = \lambda G(x,t)R(x,t)^{-1}G(x,t)^T \tag{8}$$

Path Integral Control

- The stochastic HJB equation becomes linear with respect to Ψ:

$$\partial_t \Psi(x_t, t) = \frac{\Psi(x_t, t)}{\lambda} q(x_t, t) - f(x_t, t)^T \Psi_x - \frac{1}{2} tr(B(x_t, t) B(x_t, t)^T \Psi_{xx}). \quad (9)$$

- Leverage Path Integral Theory, which relates **linear** PDEs to expectations. The solution to the linear PDE for the initial condition x_0 and time $t = 0$ is:

$$\Psi(x_{t_0}, 0) = \mathbb{E}_{\mathbb{P}}\left[exp\left(-\frac{1}{\lambda} \int_0^T q(x, t) dt \right) \Psi(x_T, T) \right]. \quad (10)$$

Path Integral Control

- Because $V(x,t) = -\lambda log(\Psi(x,t))$ and $V(x_T, T) = \phi(x_T)$:

$$\Psi(x_T) = e^{-\frac{1}{\lambda}\phi(x_T)} \tag{11}$$

- Plugging this back to the previous equation yields:

$$\Psi(x_{t_0}, 0) = \mathbb{E}_{\mathbb{P}}\left[exp\left(-\frac{1}{\lambda}S(\tau)\right)\right], \tag{12}$$

$$S(\tau) = \phi(x_T) + \int_{t_0}^{T} q(x_t, t)dt. \tag{13}$$

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

Path Integral Control - Deriving the Optimal Control

- Reminder: $u^* = -R^{-1}G^T\nabla_x V$
- Compute V_x as a function of Ψ ([4])
- The result is:

$$u^* dt = R^{-1}G^T(GR^{-1}G^T)^{-1}\frac{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(\tau))Bdw]}{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(\tau))]} \tag{14}$$

- If we discretize and set $B = G\sqrt{\Sigma}$, we get:

$$u^* = \frac{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(\tau))v_k]}{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(\tau))]}, \tag{15}$$

where Σ is the covariance of v_k (normal with zero mean)

Path Integral Control - Deriving the Optimal Control

- Notice that u^* can be approximated using **Monte Carlo estimation**:

$$u^* \approx \frac{1}{I}\sum_{i=0}^{I-1} \frac{exp(-\frac{1}{\lambda}S(\tau_i))v_{k,i}}{\frac{1}{I}\sum_{n=0}^{I-1} exp(-\frac{1}{\lambda}S(\tau_n))} = \sum_{i=0}^{I-1} w_i v_{k,i}. \tag{16}$$

- **The solution here represents the optimal control input at time t and state s_t**

Information Theoretic framework

- Free-Energy:

$$\mathcal{F}(S, \mathbb{P}, x_0, \lambda) = -\lambda log\left(\mathbb{E}_{\mathbb{P}}\left[exp\left(-\frac{1}{\lambda}S(V)\right)\right]\right). \tag{17}$$

- KL-Divergence:

$$\mathbb{KL}(\mathbb{Q}||\mathbb{P}) = \mathbb{E}_{\mathbb{Q}}\left[log\left(\frac{d\mathbb{Q}}{d\mathbb{P}}\right)\right]. \tag{18}$$

- We assume that Random-Nikodym derivative $\frac{d\mathbb{Q}}{d\mathbb{P}}$ exists [3]

Information Theoretic framework

- Upper Bound for Free-Energy:

$$\mathcal{F}(S, \mathbb{P}, x_0, \lambda) \leq \mathbb{E}_{\mathbb{Q}}[S(V)] + \lambda \mathbb{KL}(\mathbb{Q}||\mathbb{P}) \tag{19}$$

- Achieving the Upper Bound (Optimal Distribution):

$$\frac{d\mathbb{Q}^*}{d\mathbb{P}} = \frac{exp(-\frac{1}{\lambda}S(V))}{\mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(V))]} \tag{20}$$

- Push the controlled distribution $\mathbb{Q}$ as close as possible to the optimal $\mathbb{Q}^*$:

$$U^* = \underset{U}{\operatorname{argmin}}\, \mathbb{KL}(\mathbb{Q}^*||\mathbb{Q}_{U,\Sigma}) \tag{21}$$

Information Theoretic framework

- A general stochastic discrete system:

$$x_{t+1} = F(x_t, v_t) \tag{22}$$

- A control input u_t is applied to the system via noise addition:

$$v_t \sim \mathcal{N}(u_t, \Sigma) \tag{23}$$

- The control sequence input and the noisy actual input sequence to the system over T timesteps:

$$U = (u_0, u_1, .., u_{T-1}), \tag{24}$$
$$V = (\nu_0, \nu_1, .., \nu_{T-1}). \tag{25}$$

Information Theoretic framework

- Uncontrolled and controlled pdfs:

$$p(V) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m|\Sigma|)^{\frac{1}{2}}} exp\left(-\frac{1}{2}\nu_t^T\Sigma^{-1}\nu_t\right). \tag{26}$$

$$q(V|U,\Sigma) = \prod_{t=0}^{T-1} \frac{1}{((2\pi)^m|\Sigma|)^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\nu_t^T - u_t)\Sigma^{-1}(\nu_t - u_t)\right). \tag{27}$$

- Cost Function:

$$C(x_1, x_2, ..., x_T) = \phi(x_T) + \sum_{t=1}^{T-1} q(x_t). \tag{28}$$

Information Theoretic framework

- After performing the same analysis as the continuous time case, we get:

$$u_t^* = \int q^*(V)\nu_t dV = \mathbb{E}_{\mathbb{Q}^*}[\nu_t], \tag{29}$$

$$t = \{0, 1, ..., T - 1\} \tag{30}$$

Table of Contents

Interpretations of the solution

- *Path Integral Control Theory Standpoint*:
  u_t^* is the optimal control input for (t, s_t)
- *Information Theoretic Framework*:
  u^* is the optimal control input distribution in the sense that is the closest to the optimal distribution, which achieves trajectories with the lowest cost. Samples drawn from the controlled distribution $\mathbb{Q}$ will have the same mean with the samples drawn from $\mathbb{Q}^*$
- In the case of a *uni-modal* optimal distribution guarantee from infromation thereotic framework is enough
- In the case of a *multi-modal* optimal distribution we need the **symmetry breaking** assurance of Path Integral Theory (the mean is not necessary a meaningful option)

Symmetry Breaking

- *Path Integral Theory*: optimal control for current state
- *Information Theory*: optimal control sequence as the mean of the optimal distribution

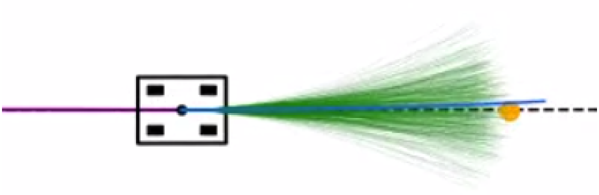


Figure: Symmetry Breaking

Delayed Choice

- As the variance of the system noise increases, the choice gets delayed [2]

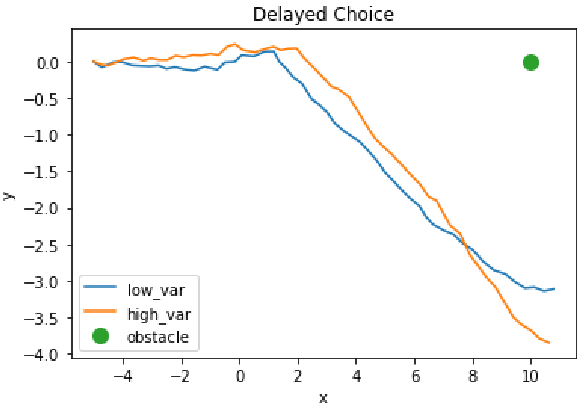


Figure: Delayed Choice

MPPI

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

$$x_{t+1} = x_t + F(x_t, v_t)\Delta t, \tag{31}$$
$$v_t \sim \mathcal{N}(u_t, \Sigma) \tag{32}$$

- general enough form (even continuous systems need to be approximated with discrete systems for MC sampling)
- No control-affine dynamics assumption

- Re-write equation 29:

$$u_t^* = \int q(V) \frac{q^*(V)}{p(V)} \frac{p(V)}{q(V)} \nu_t dV. \tag{33}$$

- The optimal control can be written as an expectation with respect to the controlled distribution:

$$u_t^* = \mathbb{E}_{\mathbb{Q}_{U,\Sigma}}[w(V)\nu_t], \tag{34}$$

$$w(V) = \frac{q^*(V)}{p(V)} exp\left(\sum_{t=0}^{T-1} -\frac{1}{2}\nu_t^T \Sigma^{-1} u_t + u_t^T \Sigma^{-1} u_t\right), \tag{35}$$

$$= \frac{1}{\eta} exp\left(-\frac{1}{\lambda}S(V) + \sum_{t=0}^{T-1} -\frac{1}{2}\nu_t^T \Sigma^{-1} u_t + u_t^T \Sigma^{-1} u_t\right), \tag{36}$$

- where $\eta = \mathbb{E}_{\mathbb{P}}[exp(-\frac{1}{\lambda}S(V)]$

MPPI

- 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:

$$w(\mathcal{E}) = \frac{1}{\eta} exp\left(-\frac{1}{\lambda}\left(S(U+\mathcal{E}) + \lambda \sum_{t=0}^{T-1} \frac{1}{2} u_t^T \Sigma^{-1}(u_t + 2\epsilon_t)\right)\right). \tag{37}$$

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

$$\eta = \sum_{n=1}^{N} exp\left(-\frac{1}{\lambda}\left(S(U+\mathcal{E}_n) + \lambda \sum_{t=0}^{T-1} \frac{1}{2} u_t^T \Sigma^{-1}(u_t + 2\epsilon_t^n)\right)\right), \tag{38}$$

- with each of the N samples drawn from the system with U as the control input sequence.

MPPI

- The resulting iterative update law is:

$$u_t^{i+1} = u_t^i + \sum_{n=1}^{N} w(\mathcal{E}_n)\epsilon_t^n \tag{39}$$

MPPI

Algorithm 2: MPPI

Given: $\mathbf{F}$: Transition Model;
K: Number of samples;
T: Number of timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \ldots \mathbf{u}_{T-1})$: 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 $\mathcal{E}^k = \{\epsilon_0^k, \epsilon_1^k, \ldots \epsilon_{T-1}^k\}$;
    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: The Algorithm [5]

Learned Dynamics

- For simulation data, use the MPPI controller with the ground truth model, whereas a human driver is used in real world experiments
- Then, run the MPPI algorithm with the neural network model, augment the dataset from the system interactions, and re-train the neural network using the augmented dataset

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: Model Learning [5]

Table of Contents

Autonomous Driving Pipeline

- The track is unknown
- One identifying round for constructing the global track map
- Then, high performance (MPPI)

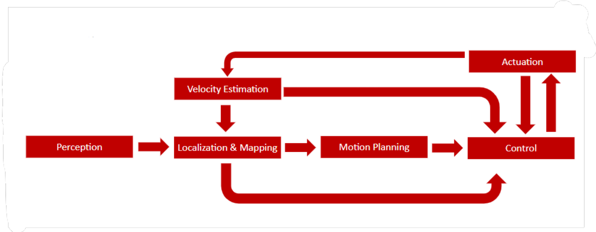


Figure: Autonomous Driving Pipeline

- Known track, represented by the (x, y) coordinates of the cones with respect to a global frame
- **State** s_t represented by a vector $s_t = f(position, velocity, orientation)$
- The **control** vector is $u = (steer, accel)$
- Maps:

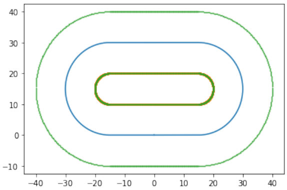


Figure: Track 1

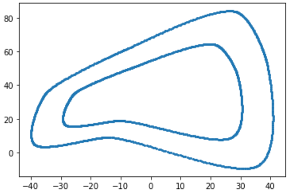


Figure: Track 2

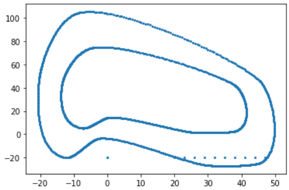
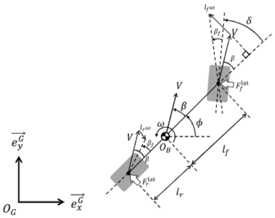


Figure: Track 3



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

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

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

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

Figure: Dynamic Bicycle Model

Cost Function

Based on [5]:

$$C(x_t) = w_1 C_{track} + w_2 C_{speed} + w_3 C_{slip} \tag{40}$$
$$\phi(x_T) = 100000C, \tag{41}$$

where:

- $C = 1_{crash}$,
- C_{track} larger as the distance to the closest cone is decreased,
- $C_{speed} = (v - v_{des})^2$,
- C_{slip} is a function of the slip angle defined as $-arctan(\frac{v_y}{|v_x|})$,
- Tuning of weights w_1, w_2, w_3 required

Experiments

Main Goals:

- Navigate through the track at a high speed (all maps)
- Avoid obstacles present in the path
- While doing so,
- Investigate the effect of the algorithm's hyperparameters

Showcasing that the algorithm works

Testing the algorithm in the 3 tracks:

- track 1
- track 2
- track 3

Effect of Time Horizon T

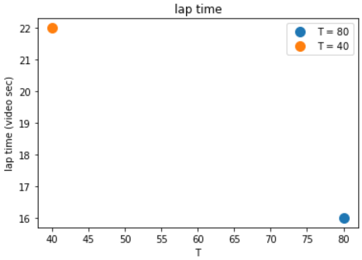


Figure: Lap Time and T

- Performance for $T = 40$ (Track 1)
- Performance for $T = 80$ (Track 1)
- Complex Map and small T
- Complex Map and large T

Effect of Time Horizon T

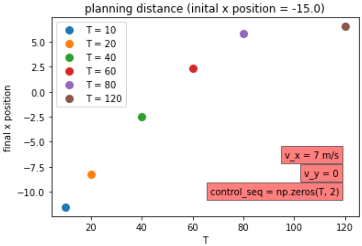


Figure: Planning Distance(x) as T increases

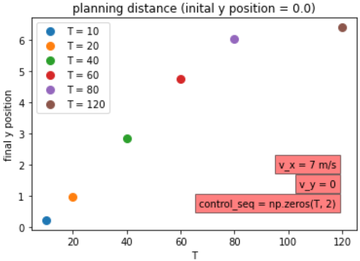


Figure: Planning Distance(y) as T increases

- Inefficient Obstacle Avoidance ($T = 40$)
- Efficient Obstacle Avoidance ($T = 80$)
- Inefficient Planning ($T = 40$)
- Efficient Planning ($T = 80$)

Effect of Σ

- Small Σ
- Med Σ
- Large Σ

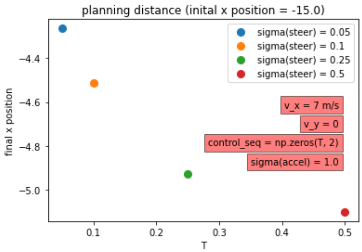


Figure: Planning Distance(x) as as $\Sigma(steer)$ increases

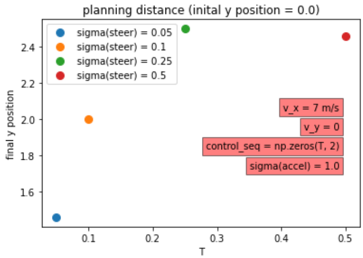


Figure: Planning Distance(y) as $\Sigma(steer)$ increases

Effect of Σ

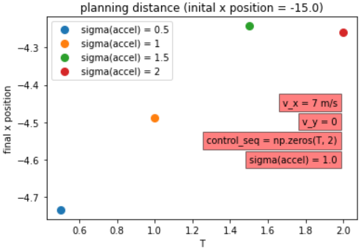


Figure: Planning Distance(x) as $\Sigma(accel)$ increases

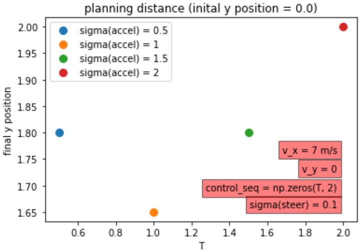


Figure: Planning Distance(y) as $\Sigma(accel)$ increases

- Obstacles at $(0, 30)$ and $(30, 12)$

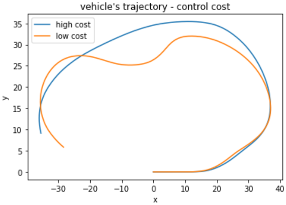


Figure: Vehicle's Trajectory

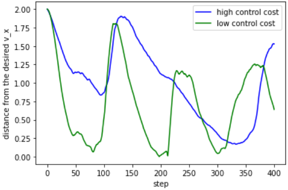


Figure: Distance from desired velocity

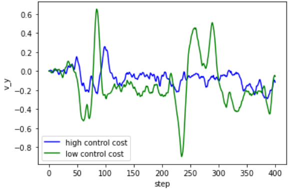


Figure: v_y

Effect of α (Control Cost)

- Obstacles at $(0, 30)$ and $(30, 12)$

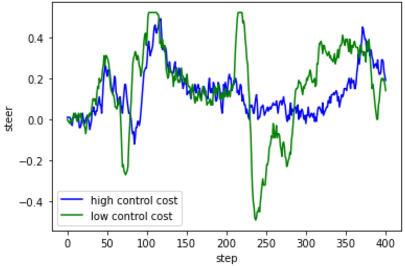


Figure: Steer values as a function of control cost

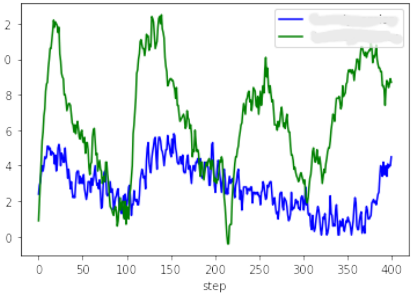


Figure: Accel variations as a function of control cost

Effect of N (number of samples)

- Obstacles at $(0, 30)$ and $(30, 12)$
- $T = 80$

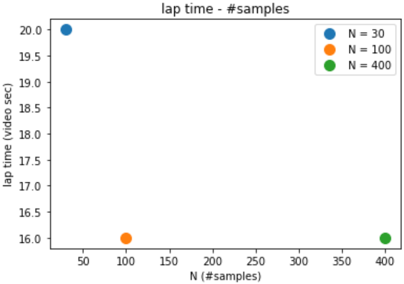


Figure: Lap Time

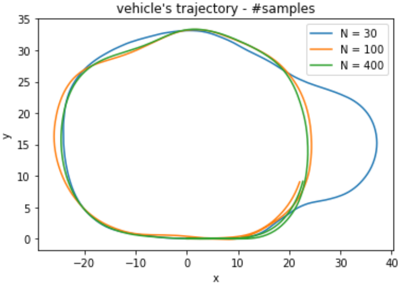


Figure: Vehicle's trajectory

Effect of N (number of samples)

- Obstacles at $(0, 30)$ and $(30, 12)$
- $T = 80$

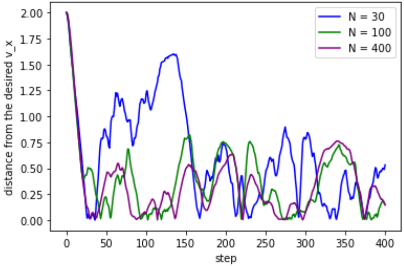


Figure: Distance from desired v_x

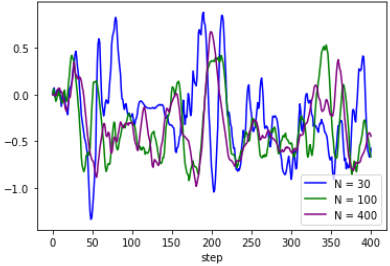


Figure: v_y

Corrupted Dynamics and Σ

- Obstacles at $(0, 30)$ and $(30, 12)$
- $T = 40$
- $N = 500$ $\Sigma(steer) = 0.05$ and $\Sigma(accel) = 1$

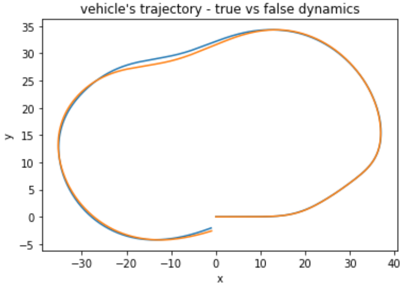


Figure: Vehicle's Trajectory

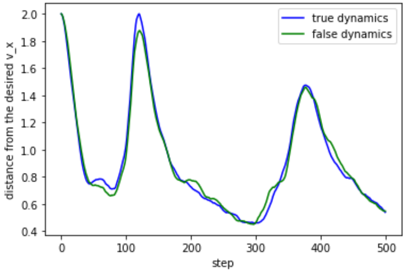


Figure: Distance from desired v_x

Corrupted Dynamics and Σ

- Obstacles at $(0, 30)$ and $(30, 12)$
- $T = 40$
- $N = 500$ $\Sigma(steer) = 0.5$ and $\Sigma(accel) = 2$

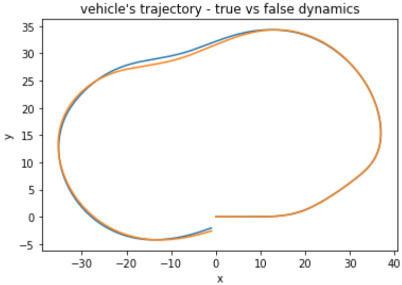


Figure: Vehicle's Trajectory

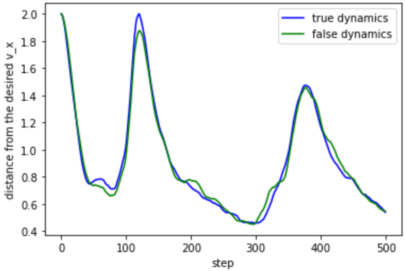


Figure: Distance from desired v_x

Corrupted Dynamics and Σ

- Obstacles at $(0, 30)$ and $(30, 12)$
- $T = 40$
- $N = 500$
- $\Sigma(steer) = 0.5$ and $\Sigma(accel) = 2$

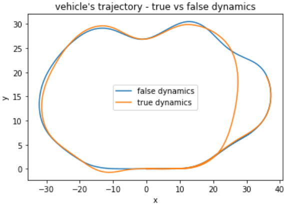


Figure: Vehicle's Trajectory

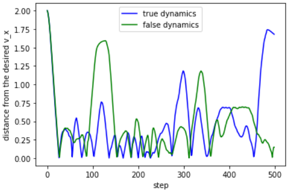


Figure: v_x

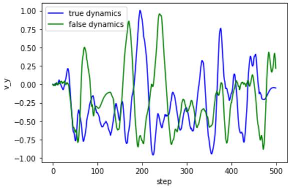


Figure: v_y

Table of Contents

Further Directions

- Depending of the chosen dynamical model you get different theoretical guarantees
- Practically, the algorithm works for general discrete systems
- Good Practises:
  - pick an appropriate cost function depending on the objectives (tuning the weights is crucial)
  - choose a large T (trade-off: computations $O(T * N)$)
  - noise variance not too large
  - #samples N is important too (again, trade-off: computations $O(T * N)$)
  - GPU (Nvidia GTX 750 Ti in the original paper [5])

Table of Contents

Further Directions

- Online Learning (Correction) of Model Dynamics ([6])
- Path Integrals for Policy improvement(PI^2, [4])
- Informed Information Theoretic Model Predictive Control ([1])

[1] Raphael Kusumoto et al. "Informed information theoretic model predictive control". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2047–2053.

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

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

[4] 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.

[5] 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.

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