Autonomous Navigation of AGVs in Unknown Cluttered Environments: log-MPPI Control Strategy

Ihab S. Mohamed1, Kai Yin2, and Lantao Liu1

Abstract—Sampling-based model predictive control (MPC) optimization methods, such as Model Predictive Path Integral (MPPI), have recently shown promising results in various robotic tasks. However, it might produce an infeasible trajectory when the distributions of all sampled trajectories are concentrated within high-cost even infeasible regions. In this study, we propose a new method called log-MPPI equipped with a more effective trajectory sampling distribution policy which significantly improves the trajectory feasibility in terms of satisfying system constraints. The key point is to draw the trajectory samples from the normal log-normal (NLN) mixture distribution, rather than from Gaussian distribution. Furthermore, this work presents a method for collision-free navigation in unknown cluttered environments by incorporating the 2D occupancy grid map into the optimization problem of the sampling-based MPC algorithm. We first validate the efficiency and robustness of our proposed control strategy through extensive simulations of 2D autonomous navigation in different types of cluttered environments as well as the cartpole swing-up task. We further demonstrate, through real-world experiments, the applicability of log-MPPI for performing a 2D grid-based collision-free navigation in an unknown cluttered environment, showing its superiority to be utilized with the local costmap without adding additional complexity to the optimization problem. A video demonstrating the real-world and simulation results is available at https://youtu.be/_uGWQEFJSN0.

Index Terms—Autonomous vehicle navigation, sampling-based MPC, MPPI, occupancy grid map path planning.

I. INTRODUCTION AND RELATED WORK

Designing a safe, reliable, and robust control methodology for autonomous navigation of autonomous ground vehicles (AGVs) in unknown cluttered environments (such as dense forests, crowded offices, corridors, warehouses, etc.) has been known as a great challenge. Such a navigation task requires the AGV to navigate safely with full autonomy while avoiding getting trapped in local minima and collision with static and dynamic obstacles while moving towards the goal, as well as respecting various system constraints. To this end, the robot should be capable of perceiving its surrounding environment and then reacting adequately. This subsequently results in a complex optimization control problem that is difficult to be solved in real-time [1].

One of the well-established and promising control strategies for collision-free navigation is Model Predictive Control (MPC) strategy, owing to its flexibility and ability to compute good control policy in the presence of the hard and soft constraints of the system to be controlled. It leverages a receding horizon strategy to plan a sequence of optimal control inputs over a prediction time-horizon; then, the first control input in the sequence is applied to the system, while the remaining control sequence is used for warm-starting the optimization at the next time-step [2]. The existing MPC-based schemes can be mainly categorized into gradient-based and sampling-based trajectory optimization methods. The gradient-based MPC methods have been successfully applied to real robotic systems, obtaining smooth collision-free trajectories in the presence of obstacles and other constraints [3]–[6]. However, the gradient-based frameworks are typically based on strong assumptions: the cost function, and sometimes system constraints, need to be differentiable in order to leverage the gradient for computing the optimal solution. Unfortunately, in practice, the optimization problem often involves non-convex and non-differentiable objective function with discontinuous collision avoidance constraints, and it can be computationally difficult to be solved in real-time. To alleviate these issues, for example, the non-differentiable system constraints can be reformulated into smooth and differentiable ones as presented in [7]. A promising alternative is the sampling-based optimization methods such as Model Predictive Path Integral (MPPI) control strategy [8] that (i) makes no assumptions or approximations on the objective functions and system dynamics, (ii) can be effectively applied on highly dynamic systems, and (iii) benefits from the parallel nature of sampling and high computational capacities of Graphics Processing Units (GPUs) utilized for speeding up the optimization.

Recently, MPPI or sampling-based MPC has been successfully applied to a wide variety of robotic applications, starting from aggressive driving [9], [10] and autonomous flight [1], [11] and ending with visual servoing [12] and reactive manipulation [13], showing outstanding performance in the presence of non-convex and discontinuous objectives,
without adding any additional complexity to the optimization problem. Despite the attractive characteristics of MPPI [14], much like any sampling-based optimization method, it might generate an infeasible control sequence (i.e., infeasible trajectory) when the distributions of all sampled trajectories drawn from the system dynamics are unfortunately surrounding some infeasible region. This may inevitably lead to either violating the system constraints or increasing the risk of being trapped in local minima. For tasks such as collision-free navigation in cluttered environments, it has been observed that the collisions may not be avoided, as the intensive simulations demonstrated in Section IV. To mitigate this problem, authors in [15] employed an iterative Linear Quadratic Gaussian (iLQG) control, as an ancillary controller, on top of MPPI for tracking the planned trajectory. Similarly, in [16], MPPI is augmented with a nonlinear $L_1$ adaptive controller to compensate for the model uncertainty. Lately, the covariance steering (CS) principle is incorporated within the MPPI algorithm, aiming to introduce adjustable trajectory sampling distributions [17]. However, the proposed solutions in [16] and [17] have not been experimentally validated on real robotic systems.

With the aim of mitigating the previous shortcomings and improving the performance of MPPI, we propose a new strategy, called log-MPPI control strategy, that provides more flexible and efficient distributions of the sampled trajectories, without the need for (i) integrating an ancillary controller on top of MPPI [15], [16], or (ii) adding a feedback term along with the injected artificial noise that requires the system dynamics to be linearized and converts the original optimization problem into a convex one [17]. The key idea of log-MPPI is that the trajectories (or, the control input updates) are sampled from the product of normal and log-normal distribution (namely, $NLN$ mixture), instead of sampling from only Gaussian distribution. With such a sampling strategy, a small injected noise variance can be utilized so that violating system constraints can be avoided; yet, we can still get desirable sampled rollout trajectories that are well spread-out for covering large state-space. In summary, the contributions of this work can be summarized as follows:

1) We provide a new sampling strategy based on normal log-normal mixture distribution. This new sampling method provides more efficient trajectories than the vanilla MPPI variants, ensuring a much better exploration of the state-space of the given system and reducing the risk of getting stuck in local minima, leading the robot to ultimately find feasible trajectories that avoid collisions, as revealed in Section IV-B.

2) Then, we show through the cartpole swing-up task given in Section IV-A the robustness of log-MPPI to run with a significantly fewer number of trajectories, opening up a new avenue for the sampling-based MPC algorithm to be run on a standard CPU instead of a GPU. We thereafter incorporate the 2D grid map, as a local costmap, into the sampling-based MPC algorithm for performing collision-free navigation in either static or dynamic unknown cluttered environments, as depicted in Section IV-C.

3) Finally, in Section V, we experimentally demonstrate our control strategy for a 2D grid-based navigation in an unknown indoor cluttered environment shown in Fig. 1; to the best of the authors’ knowledge, this is the first attempt to experimentally achieve grid-based collision-free navigation based on sampling-based MPC.

II. PRELIMINARIES

In this section, we define the optimal control problem to be solved and provide a brief review of MPPI.

A. Constrained Control Problem

Consider a discrete-time system with state $x_k \in \mathbb{R}^n$, control input $u_k \in \mathbb{R}^m$, and underlying dynamics $x_{k+1} = f(x_k, v_k)$. Let $v_k = u_k + \delta u_k \sim \mathcal{N}(u_k, \Sigma_u)$ where $\delta u_k \sim \mathcal{N}(0, \Sigma_u)$ represents the injected disturbance with a zero-mean and covariance $\Sigma_u$. Let $U = (u_0, u_1, \ldots, u_{N-1}) \in \mathbb{R}^{m \times N}$ denote the control sequence over a finite time-horizon $N$, while $X = (x_0, x_1, \ldots, x_{N-1}) \in \mathbb{R}^{n \times N}$ denotes the resulting state trajectory. Let $O^{\text{rob}}(x_k)$ and $O^{\text{obs}}$ be the area occupied by the robot and the obstacles, respectively. Our objective is to find a control sequence $U$ that generates a collision-free trajectory which allows the robot to navigate from the initial state $x_0$ to its desired state $x_f$ while minimizing a cost function $J$. This optimization problem can be formulated by MPPI as:

$$
\min_{U} J = \mathbb{E} \left[ \phi(x_N) + \sum_{k=0}^{N-1} \left( q(x_k) + \frac{1}{2} u_k^T R u_k \right) \right],
$$

subject to:

$$
x_{k+1} = f(x_k, v_k),
$$

$$
O^{\text{rob}}(x_k) \cap O^{\text{obs}} = \emptyset, \quad h(x_k, u_k) \leq 0,
$$

where $\phi(x_N)$, $q(x_k)$, and $R \in \mathbb{R}^{m \times m}$ denote the terminal cost, state-dependent running cost, and positive definite control weighting matrix, respectively. MPPI solves the problem by minimizing the objective (1a) subject to system dynamics (1b) and system constraints such as collision avoidance and control constraints (1c).

B. Review of MPPI

Unlike the gradient-based MPC methods, MPPI does not compute gradients to find the optimal solution; i.e., it is a derivative-free trajectory optimization strategy. Moreover, it makes no assumptions on the objective functions and system dynamics; i.e., highly nonlinear and non-convex functions can be easily employed. At each time-step, MPPI draws $M$ trajectories, in parallel, from the system dynamics using GPU ensuring a real-time performance. These parallel trajectories are then evaluated according to its expected cost. The cost-to-go of each rollout $\tau$ over a time-horizon $N$ is given by

$$
\tilde{S}(\tau) = \phi(x_N) + \sum_{k=0}^{N-1} \tilde{q}(x_k, u_k, \delta u_k),
$$

where $\tilde{q}(x_k, u_k, \delta u_k)$ refers to the instantaneous running cost which consists of the sum of state-dependent running cost $q(x_k)$ and quadratic control cost; it is defined as [8]

$$
\tilde{q} = q(x_k) + \frac{(1-\nu^{-1})}{2} \delta u_k^T R \delta u_k + u_k^T R u_k + \frac{1}{2} u_k^T R u_k,
$$

where $\nu$, $R$, and $\delta u_k$ are the discount factor, the control weighting matrix, and the injected artificial noise that requires the system dynamics to be linearized and converts the original optimization problem into a convex one [17].
where $\nu \in \mathbb{R}^+$ determines how aggressively the state-space is explored. Afterwards, the control sequence is updated based on a weighted average cost over all sampled trajectories. As described in [8], the optimal control sequence $\{u_k\}_{k=0}^{N-1}$ can be approximated as

$$
u_{k+1} = u_k + \frac{\sum_{m=1}^{M} \exp\left(-\frac{1}{\lambda}S(\tau_{k,m})\right) \delta u_{k,m}}{\sum_{m=1}^{M} \exp\left(-\frac{1}{\lambda}S(\tau_{k,m})\right)}, \quad (4)$$

where $S(\tau_{k,m}) \in \mathbb{R}^+$ is the cost-to-go of the $m^{th}$ trajectory at $k^{th}$ step and $\lambda \in \mathbb{R}^+$ is so-called the inverse temperature which determines how selective the weighted average of the trajectories is [10]. The control sequence is then smoothed using a Savitzky-Golay (SG) filter. Finally, the first control $u_0$ in the sequence is applied to the system, while the remaining control sequence of length $N-1$ is used for warm-starting the optimization at the next time-step.

III. log-MPPI Control Strategy

Our goal is to design a new sampling and control approach, the log-MPPI, to further improve the classic MPPI performance. Here, we briefly describe the difference between MPPI and log-MPPI: as previously discussed in Section II, MPPI does not update the injected control noise variance $\Sigma_n$, and the state-space exploration is carried out by adjusting $\nu$ (see Eq. (3)). However, if $\nu$ is too large, MPPI produces control inputs with significant chatter [8]. Similarly, as stated in [15], a higher value of $\Sigma_n$ might result in violating the system constraints and eventually state diverging. One solution could be updating, at each iteration, the variance [13]. Instead, we inject the log-normal along with normal distribution ensuring much better state exploration with a low variance value which well respects the system constraints and providing better performance with a fewer number of samples.

A. Log-normal Distribution

In log-normal theory, a positive random variable $X$ is log-normally distributed, i.e., $X \sim \mathcal{LN}(\mu, \sigma^2)$, if the natural logarithm of $X$ has a normal distribution with mean $\mu$ and variance $\sigma^2$, i.e., $\ln X \sim \mathcal{N}(\mu, \sigma^2)$. Thus, the probability density function (pdf) of the random variable $X$ is given by

$$f_X(x) = \frac{1}{\ln(2\pi)\sigma^2} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right), \quad x \in \mathbb{R}^+, \quad (5)$$

where the mean and variance of a $\mathcal{LN}(\mu, \sigma^2)$ distribution are given by

$$\mathbb{E}(X) = e^{(\mu+0.5\sigma^2)} \quad \text{and} \quad \text{Var}(X) = e^{(\mu+\sigma^2)}(e^{\sigma^2} - 1). \quad (6)$$

In spite of the fact that both normal and log-normal distributions are unbounded distributions, the log-normal distribution is asymmetric and positively skewed to the right, where the range of values lies in an interval of $[0, +\infty]$. Therefore, the pdf, $f_X(x)$, starts at zero and increases to its mode, then decreases thereafter. The degree of skewness increases as $\sigma$ increases, for a given $\mu$. Similarly, for the same $\sigma$, the pdf’s skewness increases as $\mu$ increases. In addition, the most attractive feature of such distribution compared to the alternative default distributions (such as normal, gamma, and Weibull distributions) is its capability of capturing a large range with a long right-tail, making it convenient to model large values and hence large uncertainties [18].

B. Normal Log-normal (LN) Mixture

Broadly speaking, if $X$ and $Y$ are two independent random variables, described by probability density functions $f_X(x)$ and $f_Y(y)$, then the probability density function of the product $Z = XY$ is given by

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x)f_Y(y) \frac{1}{|x|} dx. \quad (7)$$

More specifically, suppose that $X \sim \mathcal{N}(\mu_X, \sigma^2_X)$ and $Y \sim \mathcal{LN}(\mu_Y, \sigma^2_Y)$. Then, the random variable $Z = XY \sim \mathcal{LN}(\mu_X, \sigma^2_Y)$ can be labelled as normal log-normal (LN) mixture, where its mean $\mu_{LN}$ and variance $\sigma^2_{LN}$ are given by [19]

$$\mu_{LN} = \mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y) = \mu_X e^{(\mu_Y+0.5\sigma^2_Y)}, \quad (8)$$

$$\sigma^2_{LN} = \text{Var}(XY) = \mathbb{E}(X^2)\mathbb{E}(Y^2) - (\mathbb{E}XY)^2, \quad (9)$$

Let us consider the case where $\mu_X = 0$, i.e., $X \sim \mathcal{N}(0, \sigma^2)$, which is commonly used in sampling-based MPC strategies. Thus, the pdf of $Z$, given in (7), is defined as

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x) \frac{1}{2\pi^2\sigma^4} \exp\left(-\frac{x^2}{2\sigma^2} - \frac{(\ln x - \mu_X)^2}{2\sigma^2}\right) \frac{1}{x^2} dx, \quad (9)$$

which can be solved analytically [20]. It is noteworthy that $f_Z(z) = f_Z(-z)$, indicating a symmetric distribution around 0 as shown in Fig. 2(c). In addition, $Z$ can be written as a smooth function of two independent normal distributions, i.e., $Z = XY = X e^{W}$, where $W \sim \mathcal{N}(\mu_Y, \sigma^2_Y)$. When $\sigma_Y$ becomes smaller, $X$ places the mass around 0. This makes the tail of $Z$ lighter than the lognormal distribution. Fig. 2 illustrates different distributions with differing parameters.

C. log-MPPI Control Strategy

We develop our method on top of the MPPI in [8] for integrating the LN mixture sampling. Although the original derivation of MPPI is based on the controlled dynamics driven by Brownian motion noise, it can be approximately applied to the LN mixture, particularly for small $\sigma_n$. We provide a discussion on the effect of LN mixture noise on the dynamics in Appendix A. The major difference is that the trajectories, drawn from the discrete-time dynamics system $x_{k+1}$, are sampled from the LN policy, rather than from the Gaussian policy. Accordingly, the control input updates $\delta u_k$ is defined as $\delta u_k = \delta u^n_k + \delta u^\ln_k \sim \mathcal{LN}(0, \Sigma_n)$, where $\delta u^n_k \sim \mathcal{N}(0, \Sigma_n)$, $\delta u^\ln_k \sim \mathcal{LN}(\mu_{ln}^n, \Sigma_{ln}^n)$, $\Sigma_n = \sigma^2_{ln}I_m$, $\Sigma_{ln} = \sigma^2_{ln}I_m$, and $I_m$ denotes an $m \times m$ identity.
matrix. To ensure that $\delta \mathbf{u}_k$ is stochastically independent from $\delta \mathbf{u}_k$, Eq. (6) is employed in order to compute $\mu_{ln}$ and $\Sigma_{ln}$, considering $\mu = 0$ and $\sigma^2 = \sigma_n$ (namely, standard deviation of $\delta \mathbf{u}_k$). Similarly, Eq. (8) is utilized for computing $\mu_{nl,n}$ and $\sigma^2_{nl,n}$, taking into account that $\mu_{nl,n} = 0$ as $\mu_n = 0$.

![Fig. 3: Distribution of 2500 sampled trajectories generated by (a) MPPI with $\delta \mathbf{u}_k \sim \mathcal{N}(0,0.002\mathbf{I}_2)$ and (b) log-MPPI with $\delta \mathbf{u}_k \sim \mathcal{N}(0,0.017\mathbf{I}_2)$, where the robot is assumed to be located at $x = [x, y, \theta]^T = [0, 0, 0]^T$ in (m), (m), (rad)]

To demonstrate the advantages of sampling from the $\text{NLN}$ distribution on the performance of the $\text{MPPI}$ algorithm, here we provide a concrete example. The basic idea is to use a small variance (so that we can avoid violating system constraints); yet, we can still get desirable sampled trajectories that are well spread-out for covering large state-space. Specifically, (i) we first draw random samples, namely, $X \equiv \delta \mathbf{u}_k$, from a normal distribution with $\mu_n = 0$ and $\sigma^2_n = 0.002$ (see Fig. 2(a)); (ii) then, another set of random samples, namely, $Y \equiv \delta \mathbf{u}_k$, are generated from the corresponding log-normal distribution with $\mu_l = 1.023$ and $\sigma^2_l = 0.048$, as illustrated in Fig. 2(b); (iii) finally, the random variable $Z \equiv \delta \mathbf{u}_k$ that represents the product of those two independent variables is generated from the $\text{NLN}$ distribution with $\mu_{ln} = 0$ and $\sigma^2_{ln} = 0.017$. Now, let us draw $M$ sampled trajectories from $\delta \mathbf{u}_k \sim \mathcal{N}(0,0.002\mathbf{I}_2)$, as depicted in Fig. 3(a), considering the discrete-time kinematics model of the robot given in (11) and control schemes parameters listed in Section IV-B1, where $M = 2500$. In a similar way, Fig. 3(b) shows the sampled trajectories from $\delta \mathbf{u}_k \sim \mathcal{N}(0,0.017\mathbf{I}_2)$. It is interesting to observe in Fig. 3 that the distributions of the sampled trajectories generated by the log-MPPI algorithm are more flexible and efficient than the ones generated by the classical MPPI, resulting in (i) better exploration of the state-space, and (ii) reducing the probability of getting trapped in local minima.

One might argue that injecting the same control variance to the normal distribution can lead to similar results. Here, we provide the advantages of the $\text{NLN}$ distribution through (i) an analysis from the dynamics perspective in Appendix A where we show that even with the same variance, the proposed scheme can be more efficient due to the random drift term in dynamics, and (ii) the extensive simulation results carried out in the next section which show a much better exploration with more than 30% reduction in the injected noise variance $\Sigma_u$.

IV. SIMULATION-BASED EVALUATION

We evaluated and compared the two control strategies on a simulated cartpole system and a goal-oriented AGV autonomous navigation task in 2D cluttered environments.

A. Cartpole Swing-up Task

To demonstrate the impact of drawing trajectories from the $\text{NLN}$ distribution policy, instead of Gaussian policy, on the behavior of the sampling-based $\text{MPC}$ algorithm to assess the practical stability of our proposed control strategy, especially with a significantly fewer number of trajectories, we applied $\text{MPPI}$ and $\text{log-MPPI}$ on a simulated cartpole system.

1) Simulation Setup: The main objective is to swing up and stabilize the cartpole for 12s. The cartpole dynamics model are taken from [8] with assigning the same values to the system variables, while the pole length $l$ is set to 1m and the instantaneous running cost function is formulated as:

$$q(x) = 10x^2 + 10^3(1 + \cos(\theta))^2 + 2\dot{\theta}^2 + 2\dot{x}^2, \quad (10)$$

where $x$ and $\dot{x}$ are the horizontal position and velocity of the cart, while $\theta$ and $\dot{\theta}$ denote the angle and angular velocity of the pole. For both control schemes, the simulations were performed with a time prediction $t_p$ of 2s, a control frequency of 50Hz (sequentially, $N = 100$), a 1000 sampled rollouts at each time-step $\Delta t$, an exploration variance $\nu$ of 1000, an inverse temperature $\lambda$ of 0.07, and a control weighting matrix $R$ of $\frac{1}{2}\Sigma_{\pi}^{-1}$ with $\Sigma_{\pi} = 0.0225$ for log-MPPI. The Savitzky-Galoy (SG) convolutional filter, which is utilized for smoothing the control sequence $\{u_k\}_{k=0}^{N-1}$ computed by Eq. (4), is applied with a quintic polynomial function, i.e., $n_{sg} = 5$, and a window length $l_{sg}$ of 51.

2) Simulation Results: We tested the robustness of our proposed algorithm by changing the noise variance $\Sigma_u$, and number of sampled trajectories $M$, as illustrated in Fig. 4. In Figs. 4(a) and 4(b), the simulations are carried out considering two different values of $M$ (namely, $M = 1000$ and 5), while keeping the injected noise variance the same, namely, $\Sigma_u = 0.283$. We can notice from Fig. 4(a) that our control scheme achieves a slightly faster convergence; the cartpole converges to the desired configuration (i.e., $x = 0$ m and $\theta = \pi \text{ rad}$) within 39.9 s compared to 48.8 s when MPPI is used. Figure 4(b) demonstrates that the impact of decreasing $M$ on the behavior of MPPI is appreciably higher than its impact on log-MPPI, as the former produced control input that ultimately leads to a higher transient overshoot of $\theta$ and higher positioning error (about 7.6 cm). On the other side, log-MPPI still performs well with a very slightly positioning error, without compromising its robustness level and convergence rate, which opens up a new avenue for sampling-based MPC algorithm to be run on a standard CPU instead of a GPU, with a fewer number of samples. Furthermore, it is noteworthy that MPPI can achieve similar or better performance of log-MPPI by increasing $\Sigma_u$ at least 73%, as depicted in Fig. 4(c).

2Empirically, we observed that the lower the $R$, the better the performance. Accordingly, the behavior of MPPI will be slightly worse if $R$ is assigned to a high value as that in log-MPPI, which is $R = 0.233$. 

\footnote{Unlike in [17], the proposed method explores the environment and samples trajectories more efficiently than MPPI for the same injected noise $\Sigma_u$.}
B. Autonomous Navigation in Cluttered Environments

With the aim of demonstrating the prospective advantages of our proposed log-MPPI control strategy compared to the classical MPPI, extensive simulations are conducted in goal-oriented AGV autonomous navigation tasks in 2D cluttered environments.

1) Simulation Setup: In this work, we consider the kinematics model of a differential wheeled robot. The kinematics equations that govern the motion of the robot is expressed as

$$\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & 0 & 0 \\
\sin \theta & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix} \iff \dot{x} = R(\theta)u,$$

where $x = [x, y, \theta]^T \in \mathbb{R}^3$ represents the pose of the robot expressed in the world frame, $\theta$ is the orientation (or, heading) angle, the control $u = [v, \omega]^T \in \mathbb{R}^2$ denote the linear and angular velocities of the robot.

The parameters of both control strategies were set as follows: $t_p = 5$ s (i.e., $N = 250$), $M = 2500$, $\nu = 1200$, and $R = \lambda \Sigma_{\nu}^{-\frac{1}{2}}$. However, for MPPI, the inverse temperature $\lambda$ and the control noise variance $\Sigma_u = \text{Diag}(\sigma_u^2, \sigma_u^2)$ (herein, $\Sigma_u \equiv \Sigma_u$) are set to 0.572 and Diag(0.023, 0.028), respectively, while in the case of log-MPPI they are set to much lower values, namely, 0.169 and Diag(0.017, 0.019), respectively. In fact, those two hyperparameters were chosen based on the intensive simulations carried out in Tests #1 and #2 in Table I. It can be noticed that $\Sigma_u$, in the case of log-MPPI, is basically computed from a normal distribution with a variance of $\Sigma_u = \text{Diag}(0.002, 0.0022)$. For the SG filter parameters, we set $n_{sg}$ and $l_{sg}$ to 3 and 51, respectively. The real-time execution of MPPI and log-MPPI is carried out on an NVIDIA GeForce GTX 1660 Ti laptop GPU, where all algorithms were written in Python and were implemented on a differential wheeled robot, namely, ClearPath Jackal robot, integrated with the Robot Operating System (ROS) framework.

Within this work, trajectories are sampled on a GPU using the discrete-time kinematics model given in Eq. (11), where the state-dependent cost function of the 2D navigation task is simply formulated as

$$q(x) = q_{state}(x) + q_{obs}(x),$$

where $q_{state}(x) = (x - x_f)^T Q (x - x_f)$ is a quadratic cost function utilized for enforcing the robot current state $x$ to reach its desired state $x_f$, and $Q = \text{Diag}(5, 5, 2) \forall v_{des} \leq 1 \text{ m/s}$, otherwise $Q = \text{Diag}(2.5, 2.5, 2)$. $q_{obs}(x) = 10^7 C_{\text{crash}}$ heavily penalizes trajectories that collide with obstacles, where $C_{\text{crash}}$ is a Boolean variable that indicates the collision with obstacles.

2) Simulation Scenarios: We considered four various scenarios for evaluating the performance of the proposed control framework in cluttered environments. In Scenario #1, the intensive simulations (namely, $N_T = 100$ tasks) are carried out by taking into account different values of $\lambda$ and $\Sigma_u$, with the aim of (i) choosing the best sets of hyperparameters that respect the control constraints, then (ii) assessing the performance in the following scenarios. We choose $\lambda \in [0.01, 10]$ and $\Sigma_u \in [0.0004, 0.16]$, while a cluttered environment, with obstacles placed 2 m away, has been used for assessing the performance. In the last three scenarios, we randomly generated three different types of forests, each type has 50 forests, i.e., $N_T = 50$ tasks, and each forest represents a 50 m x 50 m cluttered environment. In the first type (i.e., Scenario #2), the obstacles were, on average, 1.5 m apart (namely, $d_{\text{obs}} = 1.5$ m), whilst in the second (Scenario #3) and third (Scenario #4), they placed 2 m and 3 m away, respectively. For the first three scenarios, the maximum desired velocity $v_{des}$ of the robot is set to 1.5 m/s, while in the latter it is allowed for the robot to navigate with its maximum velocity which is 2 m/s.

3) Performance Metrics: To achieve a fair performance comparison of the two control schemes: (i) first, in all simulations, the robot has to reach the same desired pose, namely, $x_f = [50, 50, 0]^T$, from the predefined initial pose $x_0 = [0, 0, 0]^T$ (in [m], [m], [deg]); (ii) second, we define a set of metrics so as to assess the overall performance such as the number of successful tasks $S_T$, success rate $S_r$, average robot trajectory length $l_{av}$ to reach $x_f$ from $x_0$, number of successful tasks with a shorter route (i.e., robot trajectory) towards the goal $N_{\text{tr}}$, and average traveling speed $v_{av}$. The task is considered to be successful if the robot successfully reaches the desired goal without colliding with obstacles. Note that $l_{av}$, $N_{\text{tr}}$, and $v_{av}$ are only computed for successful tasks $S_T$ that are successfully completed by both control schemes.

4) Simulation Results: The general performance of our proposed control schemes are summarized in Table I, considering
TABLE I: Overall performance of the two control schemes, where the gray cells represent better results.

| Test | Scheme   | $N_T$ | $|S_T|$ | $|S_R|$ [%] | $v_{av}$ [m] | $\Sigma_{av}$ | $v_{dc}$ [m/s] |
|------|----------|-------|--------|------------|--------------|---------------|---------------|
| #1   | MPPI     | 100   | 48     | 48         | 61           | 1.5           | 1.5           |
| #2   | log-MPPI | 100   | 61     | 61         | 21           | 1.5           | 1.5           |
| #3   | MPPI     | 50    | 46     | 92%        | 75.29 (33/46) | 1.34          | 1.34          |
| #4   | log-MPPI | 50    | 48     | 96         | 75.19 (31/40) | 1.34          | 1.34          |
| #5   | MPPI     | 50    | 46     | 92%        | 76.52 (9/40)  | 1.27 ± 0.18   | 1.27 ± 0.18   |
| #6   | log-MPPI | 50    | 48     | 96         | 75.29 (33/46) | 1.34 ± 0.12   | 1.34 ± 0.12   |
| #7   | MPPI     | 50    | 50     | 100        | 72.17 (21/50) | 1.82 ± 0.039  | 1.82 ± 0.039  |
| #8   | log-MPPI | 50    | 50     | 100        | 72.09 (29/50) | 1.84 ± 0.037  | 1.84 ± 0.037  |

The motive behind considering the first 60 tasks (namely, Tests #1 and #2) is that we empirically observed that assigning higher values to $\Sigma_u$ increases the possibility of violating the control constraints.

Fig. 5: Behavior of MPPI and log-MPPI in a 50 m x 50 m cluttered environment, where obstacles were 2 m apart.

MPPI is 77.8 m compared to 79.92 m for the classical MPPI.

C. Autonomous Navigation in Unknown Environments

In Section IV-B, where autonomous navigation in cluttered environments is performed, it is assumed that the costmap that represents the environment is priori known, limiting the applicability of the control schemes as unknown or partially observed environments are the most dominant in robotics applications [1]. To this end, a 2D costmap created by the costmap_2d ROS package is utilized for storing information about the robot’s surrounding obstacles [21] (see Fig. 6(b)). It employs the sensor data acquired from the environment to build a 2D or 3D occupancy grid of the data, where each cell of the occupancy is typically expressed as occupied, free, or unknown; in our case, a 2D occupancy grid map is sufficient for 2D robot navigation. Thereafter, this occupancy grid is utilized as a local costmap to be fed directly into the sampling-based MPC algorithm, for achieving collision-free navigation in either static or dynamic unknown environments.

Fig. 6: Snapshot of (a) our Jackal robot located in a forest-like environment, and (b) the corresponding 2D local costmap.

1) Simulation Setup: We considered the same simulation setup previously described in Section IV-B1, where the collision indicator function $q_{obs}(x)$, given in (12), is herein computed based on the local costmap (i.e., 2D grid map) built by the robot on-board sensor; in this work, the Clearpath Jackal robot is endowed with a Velodyne VLP-16 LiDAR sensor. The size of the robot-centered 2D grid map is set to 240 cell x 240 cell with a resolution (grid size) of 0.05 m/cell.

2) Simulation Scenarios: For the benchmark, two types of 50 m x 50 m forest-like maps in Gazebo environment are utilized, as depicted in Fig. 6(a). The first type (namely, Forest #1) contains tree shaped obstacles with a density of 0.1 trees/m$^2$, while the latter (i.e., Forest #2) with a density...
of 0.2 trees/m². In the case of Forest #1, \( v_{\text{des}} \) is set to 2 m/s, while it is reduced to 1.5 m/s in the case of Forest #2. Another scenario (namely, Corridor #1) is considered in which the robot navigates along a 6 m × 20 m corridor, with \( v_{\text{des}} = 1.5 \text{ m/s} \), in the presence of 8 pedestrians, each pedestrian holding a maximum velocity of \( v_{\text{ref}} = 0.3 \text{ m/s} \).

3) Performance Metrics: Here, we conduct a comparison between the two control schemes in the aspect of the number of collisions \( N_{\text{crash}} \), average trajectory length \( t_{\text{av}} \), average execution time per iteration \( t_{\text{mppi}} \) of the control algorithm. The desired poses (in [(m), [m], [deg]]) are defined as follows:

\[
G_1 = [0, 0, 0]^T, G_2 = [20, 20, 45]^T, G_3 = [-18, 2, 0]^T, G_4 = [20, -21, 90]^T, \quad \text{then} \quad G_5 = [20, 20, 0]^T.
\]

For the sake of simplicity, for Forest #2, the robot navigates autonomously from \( G_1 \) to only \( G_3 \), then stops.

4) Simulation Results: Table II summarizes the performance statistics of the two proposed control strategies in Forest #1 and Forest #2, considering 10 trials for each. As anticipated, the obtained results demonstrate that log-MPPI has a more flexible and efficient trajectories sampling distribution policy, resulting in (i) reducing the probability of getting stuck in a local minima (e.g., in Forest #2, \( N_{\text{crash}} = 1 \) compared to \( N_{\text{crash}} = 7 \) when MPPI is used), and (ii) improving the quality of the generated trajectory as \( t_{\text{av}} \) is appreciably shorter, especially in Forest #2. Furthermore, we can emphasize that both control schemes guarantee a real-time performance (as \( t_{\text{mppi}} < 20 \text{ ms} \)), showing the superiority of the sampling-based MPC algorithm to be deployed with 2D grid maps without adding any additional complexity to the optimization problem.

For a 2D grid-based navigation in the dynamic environment (namely, Corridor #1), the simulation results demonstrate that the autonomous vehicle successfully avoids moving agents, as shown in the supplementary video. However, we empirically noticed that the more the deployed agents, the noisier the 2D costmap, increasing the risk of being trapped in local minima.

TABLE II: Performance comparisons of MPPI and log-MPPI.

| Indicator | Forest #1 | Forest #2 |
|-----------|-----------|-----------|
|           | MPPI | log-MPPI | MPPI | log-MPPI |
| \( N_{\text{crash}} \) | 2 | 0 | 7 | 1 |
| \( l_{\text{av}} [\text{m}] \) | 158.71 ± 1.54 | 157.91 ± 0.54 | 76.12 ± 3.31 | 72.64 ± 0.80 |
| \( t_{\text{mppi}} [\text{ms}] \) | 13.72 ± 1.34 | 13.98 ± 0.68 | 13.70 ± 4.26 | 13.47 ± 1.46 |

V. REAL-WORLD DEMONSTRATION

We experimentally demonstrate the applicability of the proposed control strategies for achieving a 2D grid-based collision-free navigation in an unknown indoor cluttered environment.

1) Experimental Setup: The simulation setup formerly described in Sections IV-B1 and IV-C1 is also employed for the experimental validation. However, as the indoor environment size is tiny compared to that used for the simulation scenarios, we set \( v_{\text{des}} = 0.75 \text{ m/s} \) and \( t_{\text{p}} = 8 \text{ s} \), while the 2D grid map size is decreased to half of its nominal value (i.e., 120 cell × 120 cell) to ensure a real-time implementation of the control strategies. Our experimental platform is a fully autonomous Clearpath Jackal robot equipped with a 16-beam Velodyne LiDAR sensor utilized for (i) generating the local costmap, and (ii) estimating the robot’s pose using LOAM [22].

VI. CONCLUSION AND FUTURE WORK

In this work, we proposed an extension to the classical MPPI algorithm (namely, log-MPPI) in which the control input updates \( \delta \mathbf{u}_k \) are sampled from the normal log-normal (NLN) mixture distribution, rather than from Gaussian distribution. We also presented a sampling-based MPC framework for collision-free navigation in either static or dynamic unknown cluttered environments, by directly integrating the occupancy grid as a local costmap into the sampling-based MPC algorithm. We empirically demonstrated that the trajectory samples generated by log-MPPI are more flexible and efficient than the ones generated by MPPI, with a more than 30% reduction in the injected noise variance \( \Sigma_u \) when MPPI is employed. This subsequently results in exploring much better the state-space of the controlled system and reducing the risk of getting stuck in local minima. We demonstrated in real-world environment the possibility of feeding directly the local costmap into the optimal control problem without adding any additional complexity to the control problem, as well as ensuring a real-time performance of the proposed control strategy. In the future, we plan to implement our

Fig. 7: Planned trajectory by log-MPPI using the 2D costmap previously described in Fig. 1(b) (left to right) at \( t = 8 \text{ s} \) (\( G_1 \rightarrow G_2 \)), \( t = 20 \text{ s} \) (\( G_2 \rightarrow G_3 \)), \( t = 31 \text{ s} \) (\( G_3 \rightarrow G_4 \)), and \( t = 47 \text{ s} \) (\( G_4 \rightarrow G_2 \)).

TABLE III: Performance statistics of the two control strategies.

| Scheme | \( N_{\text{crash}} \) | \( l_{\text{av}} [\text{m}] \) | \( v_{\text{av}} [\text{m/s}] \) | \( t_{\text{mppi}} [\text{ms}] \) |
|--------|----------------|----------------|----------------|----------------|
| MPPI   | 0 | 40.25 ± 0.13 | 0.55 ± 0.13 | 11.43 ± 0.24 |
| log-MPPI | 0 | 38.95 ± 0.17 | 0.57 ± 0.14 | 11.18 ± 0.08 |
control scheme on standard CPUs rather than GPUs, aiming to reduce the computational burden. Furthermore, we will explore methods for vanishing the possibility of getting stuck in local minima and studying the theoretical stability of sampling-based MPC.

ACKNOWLEDGEMENT

The authors would like to thank Grady Williams, Ziyi Wang, and Evangelos Theodorou for the fruitful discussions for improving the work.

REFERENCES

[1] I. S. Mohamed, G. Allibert, and P. Martinet. “Model predictive path integral control framework for partially observable navigation: A quadrotor case study,” in 16th Int. Conf. on Control, Automation, Robotics and Vision (ICARCV), Shenzhen, China, Dec. 2020, pp. 196–203.

[2] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. “Constrained model predictive control: Stability and optimality,” Automatica, vol. 36, no. 6, pp. 789–814, 2000.

[3] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter. “Collision-free path integral control for reactive manipulation,” in IEEE Int. Conf. on Robotics and Automation (ICRA), 2016, pp. 8266–8272.

[4] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos. “Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles,” IEEE robotics and automation letters, vol. 5, no. 4, pp. 6001–6008, 2020.

[5] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora. “Model predictive contouring control for collision avoidance in unstructured dynamic environments,” IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4459–4466, 2019.

[6] M. A. Abbas, R. Milman, and J. M. Eklund. “Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles,” Canadian journal of electrical and computer engineering, vol. 40, no. 1, pp. 12–22, 2017.

[7] X. Zhang, A. Liniger, and F. Borrelli. “Optimization-based collision avoidance,” IEEE Transactions on Control Systems Technology, vol. 29, no. 3, pp. 972–983, 2020.

[8] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. “Constrained model predictive control: Stability and optimality,” Automatica, vol. 36, no. 6, pp. 789–814, 2000.

[9] M. A. Abbas, R. Milman, and J. M. Eklund. “Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles,” Canadian journal of electrical and computer engineering, vol. 40, no. 1, pp. 12–22, 2017.

[10] ———. “Information-theoretic model predictive control: Theory and applications to autonomous driving,” IEEE Transactions on Robotics, vol. 34, no. 6, pp. 1603–1622, 2018.

[11] H. Lu, Q. Zong, S. Lai, B. Tian, and L. Xie. “Real-time perception-limited motion planning using sampling-based MPC,” IEEE Transactions on Industrial Electronics, 2022.

[12] I. S. Mohamed, G. Allibert, and P. Martinet. “Sampling-based MPC for constrained vision based control,” in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2021, pp. 3753–3758.

[13] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, R. Famos, and B. Boots. “STORM: an integrated framework for fast joint-space model-predictive control for reactive manipulation,” in Conference on Robot Learning. PMLR, 2022, pp. 750–759.

[14] I. S. Mohamed. “MPPI-VS: Sampling-based model predictive control strategy for constrained image-based and position-based visual servoing,” arXiv preprint arXiv:2104.04925, 2021.

[15] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou. “Robust sampling based model predictive control with sparse objective information,” in Robotics: Science and Systems, 2018.

[16] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou. “LC1-adaptive MPPI architecture for robust and agile control of multicopters,” in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2020, pp. 7661–7666.

[17] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras. “Improving model predictive path integral using covariance steering,” arXiv preprint arXiv:2109.12147, 2021.

[18] N. L. Johnson, S. Kotz, and N. Balakrishnan. “Lognormal distributions,” in Continuous univariate distributions, volume 2. John wiley & sons, 1995, vol. 289.

[19] V. K. Rohatgi and A. M. E. Saleh. An introduction to probability and statistics, 3rd ed., D. J. Balding, N. A. Cressie, G. M. Fitzmaurice, G. H. Givens, H. Goldstein, G. Molenberghs, D. W. Scott, A. F. Smith, R. S. Tsay, and S. Weisberg, Eds. John Wiley & Sons, 2015.

[20] P. K. Clark. “A subordinated stochastic process model with finite variance for speculative prices,” Econometrica: journal of the Econometric Society, pp. 155–155, 1973.

[21] E. Marder-Eppstein, D. V. Lu!, and D. Herschberger. Costmap 2d package. [Online]. Available: http://wiki.ros.org/costmap_2d

[22] J. Zhang and S. Singh. “LOAM: Lidar odometry and mapping in real-time.” in Robotics: Science and Systems, vol. 2, no. 9. Berkeley, CA, 2014, pp. 1–9.

APPENDIX A

ANALYSIS OF THE NEW SAMPLING STRATEGY ON MPPI

In this appendix, we provide a brief interpretation for the trajectory rollout behavior using the proposed sampling method. Let us examine the effect on the dynamics due to the change of noise. The original dynamics of MPPI reads (i.e., Eq. (53) in [15], ignored constant)

$$dx_t = f(x_t, t) dt + G(x_t, t) u(x_t, t) dt + G(x_t, t) dB(t),$$

where $B(t)$ is Brownian motion. In the discrete version, if $\epsilon$ in the sampling is replaced by $Z = \epsilon Y = \epsilon W$, where $W$ is an independent Gaussian vector of $\epsilon$, we may consider this randomness as a new disturbance to the original dynamics. Although it is difficult to obtain an exact dynamics corresponding to the proposed method, we may use the following approximation. Assume that the original $d(B(t))$ is replaced by $d(B(t)) + \mu(t) dt + \sigma(t) dB(t)$, where $B(t)$ is a standard Brownian motion, independent of $B(t)$. To simplify notation, denote $\mu(t) + \sigma(t) B(t)$ by $W_1$, and $\kappa = \frac{1}{2\sigma^2}$. By Ito’s formula, we can get the following computation for $d(B(t)e^{W_1(t)})$:

$$e^{W_1(t)} dB(t) + B(t) e^{W_1(t)} dt + d[B(t), W_1(t)] = e^{W_1(t)} dB(t) + B(t) \left( \sigma(t)e^{W_1(t)} dB(t) + \kappa e^{W_1(t)} dt \right).$$

Thus, the sampling dynamics can be viewed as a modified one

$$dx_t = f(x_t, t) dt + G(x_t, t) u(x_t, t) dt + G(x_t, t) dB(t) e^{W_1(t)} = f(x_t, t) dt + G(x_t, t) \left( u(x_t, t) + \kappa B(t) e^{W_1(t)} \right) dt + G(x_t, t) \left( e^{W_1(t)} dB(t) + \sigma(t) B(t) e^{W_1(t)} dB_1(t) \right).$$

We have two observations. First, the drift term (the term with $dt$) is modified to $f(x, t) + G(x, t) u(x, t) + \kappa G(x, t) B(t) e^{W_1(t)}$. This can be thought of a random drift term, compared to the deterministic counterpart in the original dynamics. But the mean of the drift term remains the same with the original one. Since the drift term can be viewed as the “trend” of the path for each sample path, the proposed scheme has more diverse “trends” of the trajectories than the original one. The new sampled trajectories turn to spread out much more that than in the original dynamics, indicating that this scheme can explore more spaces than the original one. This leads to a more efficient sampling scheme than the normal distribution, even with the similar variance. Second, the noise term can be much more flexible to tune the variance of the sampling trajectories as it contains more parameters.