2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 1-5, 2023. Detroit, USA

GP-guided MPPI for Efficient Navigation in Complex Unknown Cluttered Environments

Ihab S. Mohamed*, Mahmoud Ali*, and Lantao Liu

***Abstract*— Robotic navigation in unknown, cluttered environments with limited sensing capabilities poses significant challenges in robotics. Local trajectory optimization methods, such as Model Predictive Path Intergal (MPPI), are a promising solution to this challenge. However, global guidance is required to ensure effective navigation, especially when encountering challenging environmental conditions or navigating beyond the planning horizon. This study presents the GP-MPPI, an *online* learning-based control strategy that integrates MPPI with a local perception model based on Sparse Gaussian Process (SGP). The key idea is to leverage the learning capability of SGP to construct a variance (uncertainty) surface, which enables the robot to learn about the navigable space surrounding it, identify a set of suggested subgoals, and ultimately recommend the optimal subgoal that minimizes a predefined cost function to the local MPPI planner. Afterward, MPPI computes the optimal control sequence that satisfies the robot and collision avoidance constraints. Such an approach eliminates the necessity of a global map of the environment or an offline training process. We validate the efficiency and robustness of our proposed control strategy through both simulated and real-world experiments of 2D autonomous navigation tasks in complex unknown environments, demonstrating its superiority in guiding the robot safely towards its desired goal while avoiding obstacles and escaping entrapment in local minima. The GPU implementation of GP-MPPI, including the supplementary video, is available at `https://github.com/IhabMohamed/GP-MPPI`.**

I. Introduction and Related Work

Autonomous navigation of mobile robots in unknown, cluttered, and unpredictable environments with limited sensor capabilities is a challenging task owing to the inherent uncertainty and complexity of such environments. To tackle this challenge, a *receding-horizon* strategy such as Model Predictive Control (MPC) is commonly employed. The MPC control framework allows the robot to simultaneously plan a short trajectory (sequence of actions), following which the robot executes the immediate action while planning a subsequent trajectory. To successfully achieve receding-horizon planning, the robot must consider both safety and persistent feasibility, where *safety* is achieved by avoiding collisions with any obstacles while executing a planned trajectory, and *persistent feasibility* is maintained by always generating a safe trajectory that does not result in dead-ends or local minima while progressing towards the desired goal.

One of the significant challenges in robot motion planning is that the desired goal is often situated beyond the planning horizon, which requires the use of local subgoals or *cost-to-go* heuristics for motion safety and persistent feasibility. A common strategy is to rely on single-query motion planning algorithms, such as A* and RRT^{X}, to identify feasible paths that direct the local planner towards its desired goal [1], [2]. For instance, the RRT^{X} algorithm, introduced in [2], incorporates replanning techniques from Dynamic Rapidly-exploring Random Trees (DRRT) and Rapid-exploring Random Trees (RRT*) algorithms to adjust the path during exploration based on environmental changes. However, due to its high computational demands, implementing this algorithm in *real-time* on a robot can be challenging.

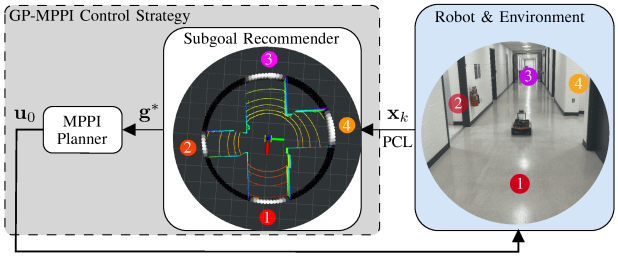


Fig. 1: Architecture of our proposed GP-MPPI control strategy, which comprises two main components: the GP-subgoal recommender and the local planner, the MPPI. First, the GP-subgoal recommender observes the surrounding environment and suggests the optimal subgoal position $\mathbf{g}^*$ to the local motion planner, where four colored circles represent the GP-recommended subgoals. MPPI then computes the optimal control sequence, which minimizes the distance to $\mathbf{g}^*$ while avoiding collision with obstacles and respecting system constraints, followed by executing the first optimal control $\mathbf{u}_0$ to the robot.

One alternative method to achieve efficient solutions for motion planning problems is the integration of MPC with data-driven methods, also known as learning-based MPC [3]. To name a few, a subgoal planning policy using Deep Reinforcement Learning (DRL) is recently proposed to guide the local MPC planner to navigate in crowded surroundings [4], [5]. Similarly, RL was utilized to choose the next subgoal from a set of predefined possibilities [6], which guides the robot through challenging environments with dead-end corridors while also prevents the MPC planner from getting trapped in local minima. Another related work that combines learning with MPC is POLO which aims to enhance MPC performance by learning a global value function [7]. Most of these approaches typically rely on either offline training or having access to the global map of the environment. In addition, many recent studies have suggested combining Gaussian Process (GP) with MPC to learn system dynamics, leading to better control performance and robustness to uncertainty [8].

Another research avenue employed gap-based techniques that identify gaps as *free* spaces between obstacles, enabling a robot to move through them while avoiding local minima and obstacles. The first developed method was the Nearness Diagram (ND) [9], but many of its variants exhibited undesired oscillatory motion. To overcome these limitations, robotics researchers have developed techniques that rely on the ge-

Authors are with the Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN 47408 USA (e-mail: {mohamedi, alimaa, lantao}@iu.edu)
*Ihab S. Mohamed and Mahmoud Ali equally contributed to this work.
This work is supported by National Science Foundation with grant numbers 2006886 and 2047169.

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-9190-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/IROS55552.2023.10341382

ometry of the gap. One such technique is the Follow-the-Gap Method (FGM), which selects a gap based on its area and computes the robot's heading using the gap center's direction relative to both the robot and the final goal [10]. Another approach is the sub-goal seeking method, which assigns a cost to each sub-goal based on the goal heading error with respect to the robot and the gap heading, and then selects the sub-goal with the lowest cost (error) [11]. The Admissible Gap (AG) method [12], an iterative algorithm that takes into account the exact shape and kinematic constraints of the robot, identifies possible admissible gaps, and selects the nearest gap as the goal.

Different from all these strategies, our proposed framework leverages a Sparse variant of Gaussian Process (SGP) which is a new perception model by "abstracting" local perception data so that the local sub-goal for navigation can be naturally extracted. Specifically, we introduce the GP-MPPI control strategy, which enhances the state-of-the-art sampling-based MPC, Model Predictive Path Integral (MPPI) [13], by incorporating the GP-subgoal recommender policy. Such a policy takes advantage of the SGP occupancy model to learn about the navigable space surrounding the robot, identifies a set of suggested subgoals, and ultimately recommends the optimal subgoal that minimizes a predefined cost function to the MPPI local planner, as demonstrated in Fig. 1. Subsequently, MPPI computes the optimal control sequence that satisfies the robot and collision avoidance constraints while moving towards the recommended subgoal, followed by executing the first optimal control $\mathbf{u}_0$ to the robot. In summary, the contributions of this work can be summarized as follows:

1) We propose an *online* learning-based control strategy that recommends subgoals solely based on local sensory information, ensuring safety and persistent feasibility; such an approach eliminates the need for a global map of the environment or an offline training process as in RL techniques, resulting in a more flexible and agile control framework that can be easily deployed in different unexplored environments, as revealed in Section III.
2) To the best of the authors' knowledge, this is the first attempt to utilize the SGP occupancy model in conjunction with sampling-based trajectory optimization methods, specifically MPPI, to efficiently explore the navigable space surrounding the robot.
3) In Sections IV and V, we validate our GP-MPPI control strategy for collision-free navigation in complex and unknown cluttered environments, using both simulation and experimental demonstrations; by comparing it with two baseline sampling-based approaches (namely, MPPI [13], and log-MPPI [14]), we show its effectiveness in overcoming local minima that may arise when the sampled trajectories of MPPI are concentrated in high-cost regions or due to challenging environmental conditions.

II. Preliminaries

To provide the necessary background for our proposed work, in this section, we formulate the optimal control problem and present a concise overview of the MPPI control strategy that can be utilized to address this problem, along with a brief introduction to the Sparse Gaussian Process (SGP) which is the backbone of our GP-subgoal recommender policy.

A. Problem Formulation

Consider a nonlinear discrete-time stochastic dynamical system

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k + \delta\mathbf{u}_k\right), \tag{1}$$

with $\mathbf{x}_k \in \mathbb{R}^{n_x}$ and $\mathbf{u}_k \in \mathbb{R}^{n_u}$ representing the state of the system and its control input, respectively. The disturbance introduced into the control input, $\delta\mathbf{u}_k$, is modeled as a zero-mean Gaussian noise with co-variance $\Sigma_{\mathbf{u}}$. Given a finite time-horizon N, we define the control sequence $\mathbf{U}$ as $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}]^\top \in \mathbb{R}^{n_u N}$ and the resulting state trajectory of the system being controlled as $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{n_x(N+1)}$. Furthermore, $\mathcal{X}^d$ is used to represent the d-dimensional space with $\mathcal{X}_{rob}(\mathbf{x}_k) \subset \mathcal{X}^d$ and $\mathcal{X}_{obs} \subset \mathcal{X}^d$ representing the robot's occupied area and obstacles' area, respectively. Let $\mathbf{x}_s$ and $\mathbf{x}_f$ denote the initial and desired (goal) state of the robot, respectively. Given $\mathcal{X}_{rob}(\mathbf{x}_k), \mathcal{X}_{obs}, \mathbf{x}_s$, and $\mathbf{x}_f$, we aim to find the optimal control sequence, $\mathbf{U}$, that allows the robot to safely and efficiently navigate from its initial state, $\mathbf{x}_s$, to the desired state, $\mathbf{x}_f$, by avoiding both getting stuck in local minima and collisions with obstacles, while minimizing a cost function J. The optimization problem at hand can be approached utilizing the classical MPPI control strategy described in [13]. This optimization can be mathematically expressed as in (2), with the objective of minimizing the cost function, J, which is comprised of the expectation of a combination of state terminal cost $\phi(\mathbf{x}_N)$, running cost $q(\mathbf{x}_k)$, and control inputs $\mathbf{u}_k$, weighted by the positive-definite matrix $R \in \mathbb{R}^{n_u \times n_u}$, taking into consideration the system dynamics outlined in (2b) and constraints such as collision avoidance and control constraints as stated in (2c).

$$\min_{\mathbf{U}} \quad J = \mathbb{E}\left[\phi(\mathbf{x}_N) + \sum_{k=0}^{N-1}\left(q(\mathbf{x}_k) + \frac{1}{2}\mathbf{u}_k^\top R\mathbf{u}_k\right)\right], \tag{2a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k + \delta\mathbf{u}_k\right), \delta\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{u}}), \tag{2b}$$

$$\mathcal{X}_{rob}(\mathbf{x}_k) \cap \mathcal{X}_{obs} = \emptyset, \ \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \tag{2c}$$

$$\mathbf{x}_0 = \mathbf{x}_s, \ \mathbf{u}_k \in \mathbb{U}, \ \mathbf{x}_k \in \mathbb{X}. \tag{2d}$$

B. Overview of MPPI Control Strategy

In order to solve the optimization control problem defined in (2), MPPI leverages Monte Carlo simulation to generate a significant number of *real-time* simulated trajectories by propagating them from the underlying system dynamics. It then evaluates the *cost-to-go* of each trajectory based on a predefined cost function and updates the optimal control sequence by considering a weighted average cost from all of the simulated trajectories. More details are given in [13], [14]. Subsequently, each trajectory τ_i in the time-horizon N can have its *cost-to-go* evaluated as given in (3), where the *cost-to-go* $\tilde{S}(\tau_i)$ is calculated as the sum of the terminal state cost $\phi(\mathbf{x}_N)$ and the instantaneous running cost $\tilde{q}(\mathbf{x}_k, \mathbf{u}_k, \delta\mathbf{u}_{k,i})$ over all time steps. The instantaneous running cost, $\tilde{q}$, expressed in (4), is comprised of the state-dependent running cost $q(\mathbf{x}_k)$ and the quadratic control cost $q(\mathbf{u}_k, \delta\mathbf{u}_k)$, where $\gamma_{\mathbf{u}} = \frac{\nu-1}{2\nu}$ and the aggressiveness in exploring the state-space

is determined by the parameter $\nu \in \mathbb{R}^+$. Specifically,

$$\tilde{S}(\tau_i) = \phi(\mathbf{x}_N) + \sum_{k=0}^{N-1} \tilde{q}(\mathbf{x}_k, \mathbf{u}_k, \delta\mathbf{u}_{k,i}) \; \forall i \in \{0, \cdots, M-1\}, \tag{3}$$

$$\tilde{q} = \underbrace{q(\mathbf{x}_k)}_{\text{State-dep.}} + \underbrace{\gamma_{\mathbf{u}} \delta\mathbf{u}_{k,i}^\top R \delta\mathbf{u}_{k,i} + \mathbf{u}_k^\top R \delta\mathbf{u}_{k,i} + \frac{1}{2}\mathbf{u}_k^\top R \mathbf{u}_k}_{q(\mathbf{u}_k, \delta\mathbf{u}_k)\text{: Quadratic Control Cost}}. \tag{4}$$

As outlined in (5) from [13], the optimal control sequence $\{\mathbf{u}_k\}_{k=0}^{N-1}$ in the vanilla MPPI algorithm is iteratively updated by taking a weighted average cost from all simulated trajectories, where $\tilde{S}(\tau_m)$ represents the *cost-to-go* of the m^{th} trajectory, and $\lambda \in \mathbb{R}^+$ denotes the "inverse temperature", which regulates the selectiveness of the weighted average of the trajectories. After smoothing the resulting control sequence with a Savitzky-Galoy filter [15], the first control $\mathbf{u}_0$ is executed in the system, with the remaining sequence utilized as a warm-start for the next optimization step. Formally,

$$\mathbf{u}_k \leftarrow \mathbf{u}_k + \frac{\sum_{m=0}^{M-1} \exp\left(\frac{-1}{\lambda}\tilde{S}(\tau_m)\right) \delta\mathbf{u}_{k,m}}{\sum_{m=0}^{M-1} \exp\left(\frac{-1}{\lambda}\tilde{S}(\tau_m)\right)}. \tag{5}$$

C. Sparse Gaussian Process

Gaussian Process (GP) is a well-established non-parametric model described by a mean function $m(\mathbf{z})$ and a co-variance function $k(\mathbf{z}, \mathbf{z}')$ (also referred to as kernel function), where $\mathbf{z} \in \mathbb{R}^{n_g}$ is the input to the GP [16]; it can be mathematically expressed as

$$f(\mathbf{z}) \sim \mathcal{GP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}')). \tag{6}$$

Let $\mathcal{D} = \{(\mathbf{z}_i, y_i)\}_{i=1}^n$ denote a dataset consisting of n input-output pairs, where each output $y_i \in \mathbb{R}$ is assumed to be the sum of an unknown underlying function $f(\mathbf{z}_i)$ and Gaussian noise ϵ_i with a zero-mean and variance σ^2, i.e., $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. In the context of GP regression, to estimate the output y^* for a given new input $\mathbf{z}^*$, the following GP prediction equation is employed

$$\begin{aligned} p(y^*|\mathbf{y}) &= \mathcal{N}(y^*|m_{\mathbf{y}}(\mathbf{z}^*), k_{\mathbf{y}}(\mathbf{z}^*, \mathbf{z}^*) + \sigma^2), \\ m_{\mathbf{y}}(\mathbf{z}) &= K_{\mathbf{z}n}\left(\sigma^2 I + K_{nn}\right)^{-1}\mathbf{y}, \\ k_{\mathbf{y}}(\mathbf{z}, \mathbf{z}') &= k(\mathbf{z}, \mathbf{z}') - K_{\mathbf{z}n}\left(\sigma^2 I + K_{nn}\right)^{-1} K_{n\mathbf{z}'}, \end{aligned} \tag{7}$$

where $m_{\mathbf{y}}(\mathbf{z})$ and $k_{\mathbf{y}}(\mathbf{z}, \mathbf{z}')$ are the GP posterior mean and co-variance functions, respectively, while $K_{nn} \in \mathbb{R}^{n \times n}$ refers to the $n \times n$ co-variance matrix of the training inputs and $K_{\mathbf{z}n} \in \mathbb{R}^n$ is n-dimensional row vector of kernel function values between $\mathbf{z}$ and the training inputs, with $K_{n\mathbf{z}} = K_{\mathbf{z}n}^\top$. Achieving a more accurate GP prediction requires the optimization of hyper-parameters, such as kernel parameters Θ and noise variance σ^2, by maximizing the log marginal likelihood

$$\log p(\mathbf{y}) = \log\left[\mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \sigma^2 I + K_{nn}\right)\right]. \tag{8}$$

The standard GP can be computationally intensive due to its complexity of $\mathcal{O}(n^3)$, where n represents the number of training instances. To mitigate this issue, various approximation methods, collectively known as Sparse Gaussian Process (SGP), have been developed as an alternative approach. Instead of using the complete training data, SGP employs a smaller set of m_s training points, called *inducing points* Z_{m_s}, resulting in a more efficient process and a lower computation complexity of $\mathcal{O}(nm_s^2)$ [17]–[19]. Our present work leverages the variational SGP method, proposed in [19], to approximate the true posterior of a GP $p(f|\mathbf{y})$ using an approximated variational posterior distribution $q(f, f_{m_s})$, where f_{m_s} are the values of the underlying function f at the *inducing points* Z_{m_s}. This approximation is done by augmenting the true posterior with the variable f_{m_s} such as $p(f, f_{m_s}|\mathbf{y}) = p(f|f_{m_s})p(f_{m_s}|y)$. Then, the approximated variational distribution $q(f, f_{m_s})$ can be factorized in the same manner as the augmented true posterior, as follows

$$q(f, f_{m_s}) = p(f|f_{m_s})\phi(f_{m_s}), \tag{9}$$

where $\phi(f_{m_s})$ is an unconstrained variational distribution over f_{m_s} and $p(f|f_{m_s})$ is the conditional GP prior. By minimizing the Kullback-Leibler (KL) divergence between the approximated and true posteriors, $\mathbb{KL}[q(f, f_{m_s})||p(f|\mathbf{y})]$, the variational SGP obtains estimates of the inducing inputs Z_{m_s} and hyperparameters (Θ, σ^2).

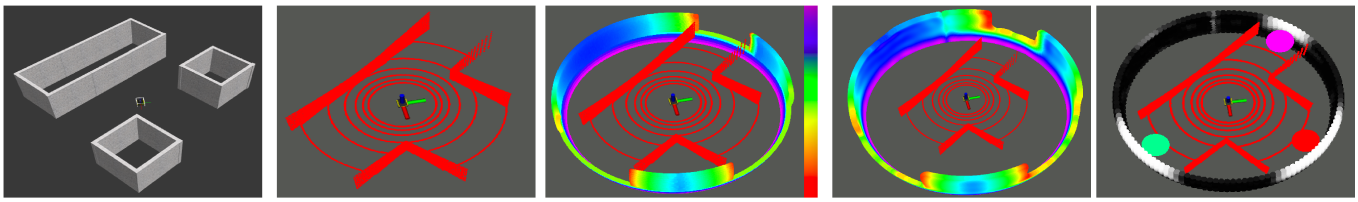
III. GP-MPPI Control Strategy

The goal of our present research, as outlined in (2), is to determine the optimal control sequence $\mathbf{U} = \{\mathbf{u}_k\}_{k=0}^{N-1}$ that enables safe and efficient navigation of the mobile robots through complex and unknown cluttered environments, while avoiding collisions with obstacles and getting trapped in local minima. Although the MPPI control framework, as summarized in [20], has many positive attributes, it is prone to generating *infeasible* control sequences or trajectories, particularly when the distribution of all sampled trajectories are concentrated within high-cost regions. To mitigate this issue, new sampling strategies proposed in [14], [21] have enabled more efficient exploration of the state-space, allowing the algorithm to find better solutions and potentially reduce the risk of trapping in local minima. Nevertheless, for specific tasks such as the one depicted in Fig. 3(b), eliminating the local minima remains a potential challenge that needs to be tackled.

One solution could be incorporating MPPI with a global planner, such as the solution presented in [22], which utilizes the RRT algorithm to guide MPPI. Instead, we introduce the GP-MPPI control strategy, a new *online* navigation technique that leverages the SGP occupancy model to learn about the navigable space surrounding the robot. Specifically, we introduce the GP-subgoal recommender policy, which identifies a set of recommended subgoals and subsequently suggests the optimal subgoal that minimizes a predefined cost function to the MPPI local planner, as depicted in Fig. 1 and explained in detail in Section III-B. Unlike conventional methods, a distinctive aspect of the proposed control strategy is that it does not require either a global map for long-term planning or an offline training process.

A. SGP Occupancy Surface Representation

Our proposed GP-subgoal recommendation policy relies on our earlier work presented in [23], [24], where we transformed pointcloud data into an *occupancy surface* and modeled it using a Sparse Gaussian Process (SGP). Within this approach, the *occupancy surface* takes the form of a 2D circular surface centered around the sensor origin and has a predefined radius



(a) Gazebo environment (b) Raw pointcloud (c) Original occupancy surface (d) SGP occupancy surface (e) SGP variance surface

Fig. 2: Illustrative example of the SGP occupancy model, with (a) the Jackal robot located in an unknown cluttered environment, (b) the raw pointcloud built by the onboard sensor, (c) the original occupancy surface where warmer colors indicate less occupancy, (d) the SGP occupancy surface reconstructed by our model, and (e) the SGP variance surface with white regions denoting GP frontiers and colored circles representing the navigation subgoals.

of r_{oc}. This surface serves as the projection space for all observed points, which are represented in spherical coordinates $(\theta_i, \alpha_i, r_i)$, where $(\theta_i, \alpha_i, r_i)$ correspond to the azimuth, elevation, and radius values of each observed point, respectively. Each point $\mathbf{z}_i$ on the occupancy surface is defined by two attributes: the azimuth and elevation angles $\mathbf{z}_i = (\theta_i, \alpha_i)$, and assigned an *occupancy value* $f(\mathbf{z}_i)$ that is a function of the point radius r_i, such as $f(\mathbf{z}_i) = r_{oc} - r_i$. Afterward, the probability of occupancy $f(\mathbf{z})$ over the occupancy surface is modeled by an SGP occupancy model, as follows

$$f(\mathbf{z}) \sim \mathcal{SGP}\left(m(\mathbf{z}), k\left(\mathbf{z}, \mathbf{z}'\right)\right),$$
$$k\left(\mathbf{z}, \mathbf{z}'\right) = \sigma_f^2 \left(1 + \frac{\left(\mathbf{z} - \mathbf{z}'\right)^2}{2\alpha\ell^2}\right)^{-\alpha}, \tag{10}$$

where σ_f^2 is the signal variance, l is the length-scale, and α is the relative weighting factor that manipulates large and small scale variations. In our SGP model, the point's occupancy to radius relation is encoded as a zero-mean function, $m(\mathbf{z}) = 0$, where the occupancy value of the non-observed points is set to zero. The Rational Quadratic (RQ) kernel, $k\left(\mathbf{z}, \mathbf{z}'\right)$, is selected as the SGP kernel due to its ability to model functions that vary across different length-scale [16]. This characteristic makes the RQ kernel well-suited for modeling the occupancy surface.

In Fig. 2, we present a concrete example of the SGP occupancy model applied to our Jackal robot, which is equipped with a Velodyne VLP-16 LiDAR and located in an unknown cluttered environment, as depicted in Fig 2(a). The figure also illustrates the raw pointcloud generated by the onboard sensor (Fig 2(b)), as well as the original occupancy surface, which represents the projection of the point clouds onto the 2D circular surface with radius r_{oc}, where warmer colors indicate areas of lower occupancy (Fig 2(c)). Furthermore, Fig 2(d) exhibits the SGP occupancy surface reconstructed by the SGP occupancy model, as previously expressed in (10). The precision of the SGP occupancy model is intensively evaluated in our previous work [23], where the results showed that an SGP occupancy model comprising of 400 inducing points generates a reconstructed point cloud with an average error of approximately 12 cm.

B. GP-Subgoal Recommender Policy

The primary advantage of GP and its variants, compared to other modeling techniques, is their ability to provide a measure of variance, which indicates the level of uncertainty, along with a function estimate (i.e., mean). More precisely, in the context of the occupancy surface, the SGP occupancy model prediction, as defined in (7), provides both mean μ_{oc_i} and variance σ_{oc_i} values for each point on the surface, where the mean represents the expected occupancy while the variance reflects the uncertainty associated with the predicted occupancy. Consequently, constructing the SGP occupancy surface is accompanied by an SGP variance surface that captures the uncertainty in the occupancy estimate, as depicted in Fig. 2(e).

Within this research, we have opened up a new avenue for effectively utilizing the SGP variance surface as a reliable indicator for distinguishing between occupied and free spaces around the robot, where regions with variances higher than a certain threshold V_{th} correspond to free space, while low-variance regions indicate occupied space. In fact, the variance surface changes across observations due to variations in the number and distribution of observed points employed in the training of the SGP model. As a result, the variance threshold V_{th} is considered to be a variable that relies on the distribution of the variance across the surface and can be calculated as $V_{th} = K_m v_m$, where $K_m \in \mathbb{R}^+$ is a tuning parameter and v_m represents the mean of the variance distribution. To identify free navigable spaces, we define a *Gaussian Process frontier* (namely, GP frontier) as the centroid point (θ_i, α_i) of each high variance region. These GP frontiers $\{f_i\}_{i=1}^{\mathcal{F}}$ serve as local recommended subgoals (see colored circles in Fig. 2(e)). Unlike the well-known frontier concept introduced in [25], it is worth noting that our GP frontier does not rely on a global occupancy map; instead, it is extracted from the uncertainty of the current observation.

Following the identification of the GP frontiers by the SGP model, a cost function J_{gp} is utilized to determine the optimal GP frontier f^* that guides the local planner (in our case, MPPI) towards the desired state $\mathbf{x}_f$. Our cost function J_{gp}, given in (11), has been established with two distinct terms. The first term, as introduced in [26], calculates the distance d_{fs} between a GP frontier f_i and the desired state $\mathbf{x}_f$. This distance criterion is used to identify the GP frontier closest to $\mathbf{x}_f$. The second term, inspired by the direction criterion proposed in [11], evaluates the direction θ_{f_i} of a GP frontier with respect to the robot heading. This criterion prioritizes a GP frontier that aligns better with the robot heading.

$$J_{gp}\left(f_i\right) = k_{dst} d_{fs} + k_{dir} \theta_{fi}^2,$$
$$f^* = \arg\min_{f_i \in \mathcal{F}} \left(J_{gp}\left(f_i\right)\right), \tag{11}$$

where k_{dst}, k_{dir} are weighting factors. The GP frontier direction θ_{f_i} is squared to indicate the absolute direction. Finally, the local planner receives the optimal subgoal $\mathbf{g}^*$, obtained by acquiring the Cartesian coordinate of the optimal

GP frontier f^*, which leads the robot to its desired state $\mathbf{x}_f$.

C. Real-Time GP-MPPI Control Algorithm

Algorithm 1 summarizes the *real-time* control cycle of the GP-MPPI algorithm, which includes two primary components: the local MPPI motion planner (described earlier in Section II-B) and the GP-subgoal recommender (explained in Section III-B). Each time-step Δt, the GP policy recommends the optimal subgoal $\mathbf{g}^*$, the current state is estimated, and a $M \times N$ random control variations $\delta\mathbf{u}$ are generated (lines 2 : 4). Then, M trajectories are simulated in parallel, propagated from the system dynamics defined in (1), and evaluated using (3) (lines 5 : 13). It is noteworthy that the minimum sampled cost trajectory, denoted as $\tilde{S}_{\min}$, among all simulated trajectories prevents numerical overflow or underflow without affecting the optimality of the algorithm [27]. After that, the optimal control sequence $\{\mathbf{u}_k\}_{k=0}^{N-1}$ is updated, smoothed with a Savitzky-Galoy filter, and the first control $\mathbf{u}_0$ is applied to the system (lines 14 : 18), while the remaining sequence of length $N-1$ is slid down to be utilized at next time-step (lines 19 : 22). In lines 25 to 38, the function known as *GP-SubgoalRecommender* is described, which takes a pointcloud input (PCL) and returns the optimal subgoal $\mathbf{g}^*$ for the local planner. To optimize the hyper-parameters Θ and inducing points Z_{m_s} of the SGP occupancy model, the pointcloud data is transformed into training data $\mathcal{D}$ (lines 26 : 29). The mean occupancy μ_{oc} and variance σ_{oc} are then estimated over the surface Z^*, and the GP frontiers are defined as those with $\sigma_{oc} > V_{th}$, where the centroids of these frontiers are converted to Cartesian coordinates (lines 30 : 34). Finally, the cost function J_{gp} in (11) is used to select the optimal subgoal $\mathbf{g}^*$ (lines 35 : 37).

In this study, we introduce two operating modes for the GP-MPPI algorithm: the simple mode (SM) and the recovery mode (RM). Under the simple mode, MPPI consistently leverages the optimal subgoal $\mathbf{g}^*$ suggested by the GP policy. In contrast, in the recovery mode, MPPI generates the optimal control sequence that steers the robot towards its desired state $\mathbf{x}_f$, adhering to the recommended subgoal only when the robot is at risk of encountering local minima. Such local minima occur when the robot's linear velocity is zero ($v = 0$) and its current state $\mathbf{x}_k$ does not match $\mathbf{x}_f$ (i.e., $\mathbf{x}_k \neq \mathbf{x}_f$). Thanks to the optimal control sequence $\{\mathbf{u}_k\}_{k=0}^{N-1}$ obtained by MPPI, we can efficiently anticipate the occurrence of local minima by imposing a condition on the mean of the predicted linear velocities over the time-horizon N, expressed as follows:

$$\mu_{\mathbf{u}} = \frac{1}{N}\sum_{i=0}^{N-1} |v_i| < \mathbf{u}_{th}, \tag{12}$$

where $\mathbf{u}_{th} \in \mathbb{R}^+$ represents a control switching threshold set based on N. If this condition is fulfilled, then MPPI will follow the subgoal recommended by the GP rather than navigating directly towards its desired state $\mathbf{x}_f$.

Algorithm 1 Real-Time GP-MPPI Control Algorithm

Given:
M, N: Number of rollouts (samples) & timesteps
$(\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}) \equiv \mathbf{U}$: Initial control sequence
$f, \Delta t$: Dynamics & time-step size
$\phi, q, \lambda, \nu, \Sigma_{\mathbf{u}}, Q, R$: Cost & control hyper-parameters
SGF: Savitzky-Galoy (SG) convolutional filter
PCL, $\mathbf{Z}^*$: Pointcloud & 2D variance surface (Grid)
$\Theta, m_s, k_{dst}, k_{dir}, k_m$: GP policy hyper-parameters

1: **while** *task not completed* **do**
2: $g^* \leftarrow$ *GP-SubgoalRecommender(PCL)*,
3: $\mathbf{x}_0 \leftarrow$ *StateEstimator()*, $\mathbf{x}_0 \in \mathbb{R}^{n_x}$
4: $\delta\mathbf{u} \leftarrow$ *GaussianNoiseGenerator()*, $\delta\mathbf{u} \in \mathbb{R}^{M\times N}$
5: **for** $m \leftarrow 0$ **to** $M-1$ *in parallel* **do**
6: $\mathbf{x} \leftarrow \mathbf{x}_0$, $\tilde{S}(\tau_m) \leftarrow 0$, $\tilde{S}(\tau_m) \in \mathbb{R}^+$
7: **for** $k \leftarrow 0$ **to** $N-1$ **do**
8: $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + f(\mathbf{x}_k, \mathbf{u}_k + \delta\mathbf{u}_{k,m})\Delta t$,
9: $\tilde{S}(\tau_m) \leftarrow \tilde{S}(\tau_m) + \tilde{q}$,
10: **end for**
11: $\tilde{S}(\tau_m) \leftarrow \tilde{S}(\tau_m) + \phi(\mathbf{x}_N)$,
12: **end for**
13: $\tilde{S}_{\min} \leftarrow \min_m[\tilde{S}(\tau_m)]$, $\forall m = \{0, \ldots, M-1\}$
14: **for** $k \leftarrow 0$ **to** $N-1$ **do**
15: $\mathbf{u}_k \leftarrow \mathbf{u}_k + \frac{\sum_{m=0}^{M-1} \exp\left(\frac{-1}{\lambda}\left[\tilde{S}(\tau_m) - \tilde{S}_{\min}\right]\right)\delta\mathbf{u}_{k,m}}{\sum_{m=0}^{M-1} \exp\left(\frac{-1}{\lambda}\left[\tilde{S}(\tau_m) - \tilde{S}_{\min}\right]\right)}$,
16: **end for**
17: $\mathbf{u} \leftarrow$ *SGF*$(\mathbf{u})$,
18: $\mathbf{u}_0 \leftarrow$ *SendToActuators*$(\mathbf{u})$,
19: **for** $k \leftarrow 1$ **to** $N-1$ **do**
20: $\mathbf{u}_{k-1} \leftarrow \mathbf{u}_k$,
21: **end for**
22: $\mathbf{u}_{N-1} \leftarrow$ *ControlSequenceInitializer*$(\mathbf{u}_{N-1})$,
23: Check for task completion
24: **end while**
25: **function** *GP-SubgoalRecommender(PCL)*,
26: $(\theta_i, \alpha_i, r_i) \leftarrow$ *Cartesian2Spherical(PCL*(x_i, y_i, z_i)*)*,
27: $oc_i = r_{oc} - r_i$, $\mathcal{D} = \{(\mathbf{z}_i, oc_i)\}_{i=1}^{n}$,
28: $f(\mathbf{z}) \sim \mathcal{SGP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}'))$, $k \leftarrow RQ$,
29: *Optimize* $(\Theta, Z_{m_s}) \leftarrow \mathcal{D}$,
30: $(\mu_{oc}, \sigma_{oc}) \leftarrow$ *$\mathcal{SGP}$-Predict*(Z^*),
31: $v_m \leftarrow$ *Mean*(σ_{oc}), $V_{th} \leftarrow K_m v_m$,
32: *GP-Frontiers* $\leftarrow (\sigma_{oc} > V_{th})$,
33: $(\theta_{f_i}, \alpha_{f_i}) \leftarrow$ *CentroidOfGP-Frontiers*,
34: $(x_{f_i}, y_{f_i}, 0) \leftarrow$ *Spherical2Cartesian*$(\theta_{f_i}, \alpha_{f_i}, r_{oc})$,
35: $d_{fs} \leftarrow$ *EuclideanDistance*$((x_{f_i}, y_{f_i}), \mathbf{x}_f)$,
36: $f^* = \arg\min_{f_i \in \mathcal{F}}(J_{gp}(f_i))$,
37: $\mathbf{g}^* \leftarrow (x_{f^*}, y_{f^*}, \theta_{f^*})$,
38: **return** $\mathbf{g}^*$
39: **end function**

IV. Simulation-Based Evaluation

In this section, the effectiveness of our proposed control strategy is assessed and compared with both vanilla MPPI and log-MPPI control strategies in a goal-oriented autonomous ground vehicle (AGV) navigation task conducted in 2D cluttered environments of unknown nature.

A. Simulation Setup:

In this study, we consider the kinematics model of a differential wheeled robot presented in [14], specifically the

fully autonomous ClearPath Jackal robot, where the robot's position and orientation in the world frame are given by $\mathbf{x} = [x, y, \theta]^\top \in \mathbb{R}^3$, and the control input $\mathbf{u} = [v, \omega]^\top \in \mathbb{R}^2$ denotes the robot's linear and angular velocities. Our autonomous AGV platform is equipped with a 16-beam Velodyne LiDAR sensor utilized for two key functions: (i) constructing the SGP variance surface, and (ii) generating the local costmap.

The simulations for all proposed control schemes were conducted with the following parameters: a prediction time of $6\,\mathrm{s}$, a control frequency of $30\,\mathrm{Hz}$ (i.e., $N = 180$), sampling 2528 rollouts per time-step Δt, and an exploration variance ν of 1200. Additionally, a control weighting matrix R, expressed as $\lambda \Sigma_n^{-\frac{1}{2}}$, is utilized. In the case of MPPI and GP-MPPI, the inverse temperature λ and the control noise co-variance $\Sigma_{\mathbf{u}} = \Sigma_n = \mathrm{Diag}\left(\sigma_v^2, \sigma_w^2\right)$ are both set to 0.572 and $\mathrm{Diag}\,(0.023, 0.028)$, respectively. However, for log-MPPI, different values of 0.169 and $\mathrm{Diag}\,(0.017, 0.019)$ are used for these parameters, along with a normal distribution that has a co-variance of $\Sigma_n = \mathrm{Diag}\,(0.002, 0.0022)$ (For more details, refer to [14]). The Savitzky-Galoy (SG) convolutional filter is utilized with a quadratic polynomial function, i.e., $n_{sg} = 2$, and a window length l_{sg} of 51. The occupancy surface was constructed with an occupancy radius r_{oc} of 5 meters, a full azimuth range of -180^o *to* 180^o, and elevation height of 0^o *to* 15^o. The SGP occupancy model was designed with 400 inducing points ($Z_m = 400$), where the GP frontiers were identified based on a variance threshold of $V_{th} = K_m v_m$, where K_m was set to 0.4. For the distance and direction factors K_{dst} and K_{dir} of the cost function J_{gp}, we assigned weighting factors of 5 and 4, respectively. To enable the recovery mode of the GP-MPPI, we have set the control threshold, $\mathbf{u}_{th}$, to $0.55\,[\mathrm{m/sec}]$. All the proposed control schemes, which are written in Python and integrated with the Robot Operating System (ROS) framework, are executed in *real-time* on an NVIDIA GeForce GTX 1660 Ti laptop GPU, with the GP-subgoal recommender built on *GPflow* [28].

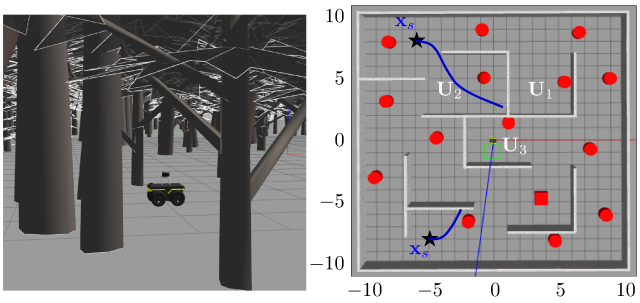
To accomplish the 2D navigation task, we adopt a state-dependent cost function described in (13), which comprises two terms. The first term, with $Q = \mathrm{Diag}(2.5, 2.5, 5)$, aims to steer the robot towards its desired state, whereas the second term incorporates a Boolean variable $\mathbb{I}_{\text{crash}}$ to heavily penalizes collisions with obstacles.

$$q(\mathbf{x}_k) = (\mathbf{x}_k - \mathbf{x}_f)^\top Q (\mathbf{x}_k - \mathbf{x}_f) + 10^3 \mathbb{I}_{\text{crash}}. \tag{13}$$

Since the robot is operating in unknown environments, it relies on a 2D costmap to maintain a record of obstacles in its vicinity. This costmap is generated by analyzing sensor data from the environment and constructing a 2D occupancy grid, with each cell typically categorized as *occupied*, *free*, or *unknown* [29]. The generated occupancy grid is subsequently employed as a 2D local costmap, feeding directly into the sampling-based MPC algorithm, enabling safe and collision-free navigation. The robot-centered *2D* local costmap, which is built by the on-board Velodyne VLP-16 LiDAR sensor, has a size of $200\,\mathrm{cell} \times 200\,\mathrm{cell}$ and a grid resolution of $0.05\,\mathrm{m/cell}$. Finally, throughout the simulations, the maximum linear velocity $v_{\max}$ of the robot is set to $1.5\,\mathrm{m/s}$.

B. Simulation Scenarios and Performance Metrics:

The benchmark evaluation utilizes two types of Gazebo simulation environments, as depicted in Fig. 3. The first type, referred to as *Forest #1*, is a $50\,\mathrm{m} \times 50\,\mathrm{m}$ forest-like environment characterized by tree-shaped obstacles with a density of $0.2\,\mathrm{trees/m^2}$; The other type, named *Maze #1*, is a $20\,\mathrm{m} \times 20\,\mathrm{m}$ maze-like environment with three U-shaped rooms (i.e., U_1, U_2, and U_3), as well as various other obstacles (highlighted in red in Fig. 3(b))[1]. In the first scenario, denoted as *Forest #1*, the robot is directed to navigate from an initial pose $\mathbf{x}_s = [-5, -8, 0]^\top$ to a desired pose $\mathbf{x}_f = [20, 20, 45]^\top$ in ([m], [m], [deg]). Meanwhile, in *Maze #1*, we conducted two separate control missions to (i) evaluate the robustness of our proposed control strategy, and (ii) examine its performance under the two different operating modes, previously described in Section III-C. The first mission, MU_1, requires the robot to navigate from $\mathbf{x}_s = [-5, -8, 60]^\top$ to a desired pose $\mathbf{x}_f = [4, 4, 45]^\top$ located inside U_1; while, in the second mission, named MU_2, the robot starts at $\mathbf{x}_s = [-6, 8, 0]^\top$, crosses U_2, and reaches a desired pose of $\mathbf{x}_f = [8, -8, 170]^\top$.



(a) Gazebo forest-like environment (b) Gazebo maze-like environment

Fig. 3: Snapshot of our Jackal robot located in (a) forest-like and (b) maze-like environments, with blue lines denoting the robot's paths generated by log-MPPI, which eventually guide it to local minima.

To ensure a fair and comprehensive comparison of the three control schemes, we have established a set of performance metrics, including the *task completion percentage* $\mathcal{T}_{\text{c}}$, the *average distance* traveled by the robot d_{av} to reach $\mathbf{x}_f$ from $\mathbf{x}_s$, the *average linear velocity* v_{av} of the robot within the cluttered environment, and the *percentage of assistance* $\mathcal{A}_{\text{gp}}$ provided by the GP-subgoal recommender policy to MPPI when the recovery mode is utilized. The successful task completion entails the robot reaching the target position without encountering obstacles or getting trapped in local minima $\mathcal{R}_{\text{lm}}$.

C. Simulation Results:

We evaluated the effectiveness of the proposed control strategies in *Forest #1* and *Maze #1* (i.e., MU_1 & MU_2) through 10 trials each, and the resulting performance statistics are summarized in Table I. The performance results demonstrate that, as expected, the proposed GP-MPPI control strategy outperforms both the vanilla MPPI and log-MPPI as the autonomous vehicle successfully accomplished all control missions (with $\mathcal{T}_{\text{c}} = 100\%$) without getting stuck in local

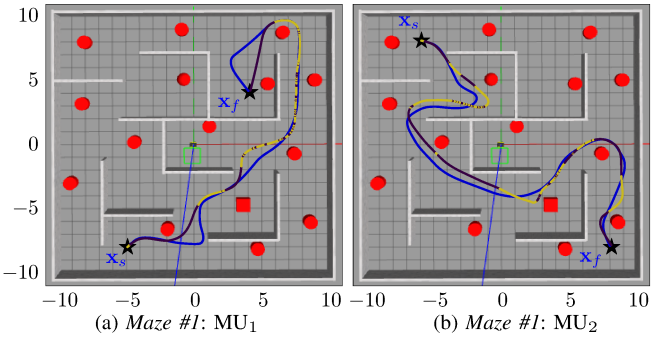
[1]To evaluate the local planner's obstacle avoidance capability, the red obstacles are intentionally made undetectable as occupied space by the GP-subgoal recommender, as occupancy elevation height is set to a higher value.

minima or colliding with obstacles (i.e., $\mathcal{R}_{\text{lm}} = 0$), despite having limited perception range and incomplete knowledge of the environment. In contrast, in *Forest #1*, log-MPPI achieved a task completion rate $\mathcal{T}_{\text{c}}$ of 95.72% over 10 trials, compared to 86.87% when MPPI was utilized. Additionally, log-MPPI encountered local minima only twice, while MPPI was trapped six times. Nevertheless, both control methods were unable to complete any of the trials in MU_1 and MU_2 due to the challenging environmental conditions (refer to the robot trajectories generated by log-MPPI in Fig. 3(b)). Additionally, our proposed approach in *Forest #1* provided a shorter route towards the desired state $\mathbf{x}_f$, especially when the recovery mode (RM) is activated, similar to the optimal trajectory of the baselines, with an average linear velocity v_{av} of $1.30\,\text{m/s}$, which approaches the maximum specified velocity of $1.5\,\text{m/s}$.

TABLE I: Comparison of performance statistics for proposed control strategies performing missions in *Forest #1* and *Maze #1* (MU_1 and MU_2), where the gray cells indicate better results.

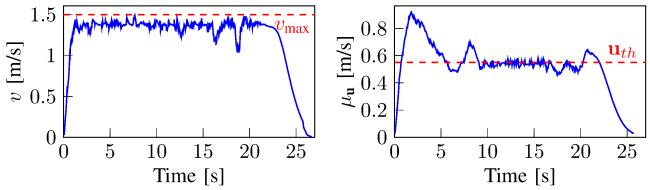
Indicator	***Baselines***		***Ours (GP-MPPI)***	
	MPPI	*log-MPPI*	*Mode: SM*	*Mode: RM*
Mission: *Forest #1*, $v_{\max} = 1.5\,\text{m/s}$				
$\mathcal{T}_{\text{c}}$ [%] ($\mathcal{R}_{\text{lm}}$)	86.87 (6)	95.72 (2)	100 (0)	100 (0)
d_{av} [m]	38.60 ± 0.04	38.62 ± 0.39	40.19 ± 0.54	39.16 ± 0.35
v_{av} [m/s]	1.29 ± 0.01	1.18 ± 0.24	1.29 ± 0.01	1.30 ± 0.01
$\mathcal{A}_{\text{gp}}$[%]	–	–	–	11.04 ± 3.58
Mission: MU_1, $v_{\max} = 1.5\,\text{m/s}$				
$\mathcal{T}_{\text{c}}$[%] ($\mathcal{R}_{\text{lm}}$)	10.93 (10)	10.96 (10)	100 (0)	100 (0)
d_{av} [m]	3.58 ± 0.03	3.59 ± 0.06	34.48 ± 1.85	32.74 ± 0.41
$\mathcal{A}_{\text{gp}}$[%]	–	–	–	46.82 ± 5.33
Mission: MU_2, $v_{\max} = 1.5\,\text{m/s}$				
$\mathcal{T}_{\text{c}}$[%] ($\mathcal{R}_{\text{lm}}$)	22.66 (10)	22.76 (10)	100 (0)	100 (0)
d_{av} [m]	8.82 ± 0.08	8.86 ± 0.09	38.92 ± 1.11	41.74 ± 1.36
$\mathcal{A}_{\text{gp}}$[%]	–	–	–	45.88 ± 4.89

Concerning the two modes of GP-MPPI, it is observed that activating the recovery mode (RM) during *Forest #1* and MU_1 missions improves the average distance traveled d_{av} by the robot. For instance, in MU_1, d_{av} was approximately $32.74\,\text{m}$ with RM, whereas with the simple mode (SM), which consistently relies on the subgoal recommended by GP, d_{av} was roughly $34.48\,\text{m}$. On the other hand, during the MU_2 mission, the RM produced a slightly longer robot trajectory than the SM since operating our proposed GP-MPPI in the RM strikes a balance between the state-dependent cost function that directs the robot to follow a direct route towards the desired state and the optimal subgoal recommended by the GP policy that forces the robot to avoid the dead-ends associated with rooms U_2 and U_3 on its way to $\mathbf{x}_f$, as illustrated in Fig. 4(b). We can also see that, due to the presence of U-shaped rooms in *Maze #1*, the GP provides more assistance, represented by $\mathcal{A}_{\text{gp}}$, than in *Forest #1*. In Fig. 4, we illustrate through an example from the conducted trials the robot trajectories generated by GP-MPPI under the two operating modes in *Maze #1*. We can clearly observe that our proposed control strategy successfully achieves collision-free navigation in both modes, without getting stuck in local minima. As an example, Fig. 5(a) displays the velocity profile of the robot during the MU_1 mission shown in Fig. 4(a), while using GP-MPPI with RM, along with its corresponding mean of the predicted linear velocities $\mu_{\mathbf{u}}$ over the given time-horizon N (see Fig. 5(b)). The mean values that fall below the switching threshold $\mathbf{u}_{th}$, set at $0.55\,[\text{m/sec}]$, denote the intervals where the RM is active, and are visually emphasized in yellow in Fig. 4(a).



(a) *Maze #1*: MU_1 (b) *Maze #1*: MU_2

Fig. 4: Behavior of GP-MPPI in a $20\,\text{m} \times 20\,\text{m}$ maze-like environment under the two operating modes, where the blue trajectory indicates the simple mode; in the recovery mode, the yellow color indicates the trajectory's segments where GP assisted MPPI to avoid local minima.



(a) Linear velocity v (b) Mean of linear velocities $\mu_{\mathbf{u}}$

Fig. 5: Robot velocity profile and the corresponding mean of predicted linear velocities for GP-MPPI with RM on the MU_1 mission.

V. Real-World Demonstration

In this section, we experimentally demonstrate the applicability of our proposed control strategy in achieving a safe 2D grid-based collision-free navigation in a complex and unknown indoor cluttered environment.

1) Experimental Setup and Validation Environment: To conduct our experimental validation, we used the simulation setup previously outlined in Section IV-A, except for (i) setting the maximum speed $v_{\max}$ to $1.0\,\text{m/s}$ to avoid the robot localization error associated with using the RealSense camera as a source of localization, (ii) setting the occupancy radius r_{oc} to $3.0\,\text{m}$, and (iii) decreasing the size of the 2D grid map to $120\,\text{cell} \times 120\,\text{cell}$. We also decreased the recovery mode switching threshold $\mathbf{u}_{th}$ to $0.3\,\text{m/s}$ to be compatible with the updated $v_{\max}$. Additionally, to ensure *real-time* execution of the GP-subgoal recommender policy, we decrease the resolution of the SGP variance surface to one-third of its original value along the azimuth axis while keeping the original resolution along the elevation axis. We employed an L-shaped indoor corridor environment measuring $9\,\text{m} \times 14\,\text{m}$ for experimental validation. The environment has a varying width between $1.8\,\text{m}$ and $2.8\,\text{m}$ and contains randomly placed boxes-like obstacles,

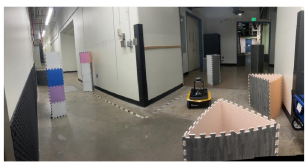


Fig. 6: Panoramic photo of our L-shaped indoor environment.

as depicted in Fig. 6. The assigned control mission of the robot is to navigate from $\mathbf{x}_s = [0, 0, 0]^\top$ and arrive at $\mathbf{x}_f = [7.5, 13, 90]^\top$.

2) Experimental Results: The performance statistics of our proposed GP-MPPI control scheme, gathered from four trials conducted in our indoor environment, are summarized in Table II for the two operating modes. From all trials, we can conclude that both operating modes provide collision-free navigation in the cluttered environment with an average linear velocity of $0.80\,\mathrm{m/sec}$, without the risk of being trapped in local minima (as $\mathcal{R}_{\text{lm}} = 0$) while moving towards its desired state. This ensures the safety and consistent feasibility of the receding-horizon planning. In contrast, it is observed that the vanilla MPPI and log-MPPI consistently failed to complete any of the trials due to being trapped in the first edge of the L-shaped environment. However, MPPI managed to avoid such traps with the aid of the GP-subgoal recommender policy in the recovery mode (RM), which provides an average assistance percentage $\mathcal{A}_{\text{gp}}$ of roughly 31.36%. More details about the simulation and experimental results, including the behavior of the baselines, are provided in the supplementary video: https://youtu.be/et9t8X1wHKI.

TABLE II: Performance statistics of the two modes of GP-MPPI.

Mode	$\mathcal{T}_{\text{c}}$ [%] ($\mathcal{R}_{\text{lm}}$)	d_{av} [m]	v_{av} [m/sec]	$\mathcal{A}_{\text{gp}}$[%]
SM	100 (0)	20.06 ± 0.21	0.80 ± 0.012	–
RM	100 (0)	20.18 ± 0.30	0.76 ± 0.066	31.36 ± 11.72

VI. Conclusion

In this work, we proposed the GP-MPPI control strategy, which comprises two primary components: the GP-subgoal recommender policy and the local planner, the MPPI. First, the GP-subgoal recommender utilized the learning capacity of SGP to create a reliable SGP variance surface, which served as an indicator for differentiating between occupied and free spaces around the robot. Consequently, a set of suggested subgoals was identified, and the optimal subgoal that minimizes a predefined cost function was recommended to the local MPPI planner. Based on the recommended subgoal, MPPI computes the optimal control input that enables the robot to navigate towards the goal efficiently and safely while accounting for its dynamics and avoiding collisions. By conducting a combination of simulated and real-world experiments, we have shown that our proposed control strategy is superior to the vanilla MPPI and log-MPPI methods in achieving efficient and safe navigation in unknown and complex environments, thereby avoiding the risk of getting stuck in local minima.

References

[1] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Trans. on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[2] M. Otte and E. Frazzoli, "RRT$^{\text{X}}$: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[3] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robot, and Auto. Syst.*, vol. 3, pp. 269–296, 2020.

[4] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robot. and Automat. Lett.*, vol. 6, no. 3, pp. 4616–4623, 2021.

[5] M. Lodel, B. Brito, A. Serra-Gómez, L. Ferranti, R. Babuška, and J. Alonso-Mora, "Where to look next: Learning viewpoint recommendations for informative trajectory planning," in *Int. Conf. on Robot. and Automat. (ICRA)*, pp. 4466–4472, IEEE, 2022.

[6] C. Greatwood and A. G. Richards, "Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control," *Autonomous Robots*, vol. 43, pp. 1681–1693, 2019.

[7] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," in *Int. Conf. on Learning Representations*, 2019.

[8] J. Wang, M. T. Fader, and J. A. Marshall, "Learning-based model predictive control for improved mobile robot path following using gaussian processes and feedback linearization," *J. of Field Robotics*, 2023.

[9] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. on Robot. and Automat.*, vol. 20, no. 1, pp. 45–59, 2004.

[10] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: Follow the gap method," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.

[11] C. Ye and P. Webb, "A sub goal seeking approach for reactive navigation in complex unknown environments," *Robotics and Autonomous Systems*, vol. 57, no. 9, pp. 877–888, 2009.

[12] M. Mujahed, D. Fischer, and B. Mertsching, "Admissible gap navigation: A new collision avoidance approach," *Robotics and autonomous systems*, vol. 103, pp. 93–110, 2018.

[13] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[14] I. S. Mohamed, K. Yin, and L. Liu, "Autonomous navigation of AGVs in unknown cluttered environments: log-MPPI control strategy," *IEEE Robot. and Automat. Lett.*, vol. 7, no. 4, pp. 10240–10247, 2022.

[15] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[16] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press, 2005.

[17] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," in *Proc. of the 16th annual conf. on neural information processing syst.*, pp. 609–616, 2003.

[18] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, p. 1257, 2006.

[19] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intell. and statist.*, pp. 567–574, PMLR, 2009.

[20] 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)*, pp. 3753–3758, 2021.

[21] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *Int. Conf. on Robot. and Automat. (ICRA)*, pp. 1478–1484, 2022.

[22] C. Tao, H. Kim, and N. Hovakimyan, "RRT guided model predictive path integral method," *arXiv preprint arXiv:2301.13143*, 2023.

[23] M. Ali and L. Liu, "Light-weight pointcloud representation with sparse gaussian process," in *Int. Conf. on Robot. and Automat. (ICRA)*, pp. 4931–4937, 2023.

[24] M. Ali and L. Liu, "GP-Frontier for local mapless navigation," in *Int. Conf. on Robot. and Automat. (ICRA)*, pp. 10047–10053, 2023.

[25] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. of the second int. conf. on Autonomous agents*, pp. 47–53, 1998.

[26] K. Yan and B. Ma, "Mapless navigation based on 2D LIDAR in complex unknown environments," *Sensors*, vol. 20, no. 20, p. 5802, 2020.

[27] 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), pp. 196–203, Dec. 2020.

[28] A. G. d. G. Matthews, M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," *J. Mach. Learn. Res.*, vol. 18, no. 40, pp. 1–6, 2017.

[29] E. Marder-Eppstein, D. V. Lu, and D. Hershberger, "Costmap_2d package."