Bayesian Optimization for Learning Nonlinear MPC in Autonomous Agent Navigation

Lorenzo Ortolani, Gabriel Voss, Gabriele Beltrami, Francesco Dorati, Tommaso Felice Banfi
Talos Robotics AI
Milan, Italy
{lorenzo.ortolani, gabriel.voss, gabriele.beltrami, francesco.dorati, tommasofelice.banfi}@talosrobotics.ai

***Abstract*—Real-time autonomous navigation in dynamic, unknown environments remains a fundamental challenge for mobile robotics. We propose a map-free framework that tightly integrates reactive rolling-horizon planning with nonlinear Model Predictive Control (MPC). At each control cycle, a LiDAR-based Gaussian occupancy representation is constructed and used to generate collision-free trajectories via A* search, which are then tracked by a CasADi/IPOPT MPC formulation incorporating a smooth sigmoid obstacle barrier. To improve robustness to parameter sensitivity, we adopt an offline Bayesian optimization scheme based on Tree-structured Parzen Estimators (TPE), which identifies near-optimal controller parameters with respect to a composite navigation objective. In addition, a Gaussian Process surrogate is used to analyze parameter sensitivity and provide insight into the optimization landscape.**

The proposed framework is robot-agnostic and is evaluated on the Unitree Go2 quadruped in simulation using Gazebo, followed by deployment on the physical robot. Experimental results show that parameters tuned in simulation transfer effectively to hardware, maintaining comparable performance without additional tuning. The full system achieves up to a 90.0% navigation success rate when deployed, along with a 38.9% average improvement in the evaluation metrics across simulated environments.

***Index Terms*—Autonomous Agent Navigation, Model Predictive Control, Non-Linear Optimization, Bayesian Optimization**

I. Introduction

Autonomous navigation in unknown, cluttered environments is crucial for mobile robots deployed in real-world settings. The challenge is compounded for legged platforms, where complex contact dynamics demand smooth and dynamically consistent motion commands. Classical decoupled architectures, where a global planner and low-level controller operate independently, often fail to capture the coupling between planning and dynamic feasibility [1], leading to jerky execution, constraint violations, and poor out-of-sample generalization.

Model Predictive Control (MPC) provides a principled framework for simultaneous path following and constraint satisfaction over a receding horizon [2]. However, real-time deployment is historically limited by two requirements: (i) a computationally efficient environment representation for obstacle avoidance, and (ii) well-calibrated cost-function weights that balance tracking accuracy, control effort, and safety. Manual tuning of these weights is time-consuming and environment-specific, with parameters that generalize poorly across different environments and navigation tasks [3].

Corresponding author: T. F. Banfi, tommasofelice.banfi@talosrobotics.ai

Fig. 1: A customized Unitree Go2 quadruped robot by Talos Robotics AI used for our real-world navigation experiments.

We address these limitations with a hierarchical, modular navigation stack designed as a general-purpose framework and validated on the Unitree Go2 quadruped. The primary contributions of this work include a map-free, rolling-horizon navigation stack combining Gaussian occupancy mapping, navigation graphs, A* planning, and nonlinear MPC with differentiable obstacle avoidance. To optimize this architecture, we introduce a Bayesian Optimization framework that uses TPE to sweep the 11-dimensional MPC parameter space across a training suite of simulation environments, identifying near-optimal weight configurations through a structured multi-metric composite score. Finally, we provide extensive experimental validation across three environments and hardware deployment on the Unitree Go2 in Fig. 1 performing several real-world navigation trials, confirming consistent performance gains and negligible sim-to-real degradation.

The remainder of this paper is organized as follows. Section II formulates the problem and reviews related work. Section III details the overall system architecture and the specifics about each module of the framework from the rolling-horizon path planner to the definition of the Bayesian Optimization pipeline. Sections IV and V present the experimental setup and results, discussing the performance and limitations. Section VII concludes and outlines future research directions.

II. Background

A. Problem Statement

Let the robot state at time step k be $\mathbf{x}_k = [p_{x,k},\, p_{y,k},\, \psi_k]^\top \in \mathcal{X} \subset \mathbb{R}^3$, where $(p_x, p_y) \in \mathbb{R}^2$ denotes the position in the world frame and $\psi \in (-\pi, \pi]$ is the heading angle. The control input is the body-frame velocity command $\mathbf{u}_k = [v_{x,k},\, v_{y,k},\, \omega_k]^\top \in \mathcal{U} \subset \mathbb{R}^3$, comprising forward velocity, lateral velocity, and yaw rate, respectively.

At each time step the agent receives a LiDAR point cloud $\mathcal{P}_k = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N_k}$ and a goal pose $\mathbf{x}_g \in \mathcal{X}$.

The navigation problem is formulated as a receding-horizon optimal control problem. At each planning cycle the agent solves:

$$\min_{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}} \; J(\mathbf{x}_{0:N},\, \mathbf{u}_{0:N-1},\, \mathcal{P}_k,\, \theta) \tag{1}$$

subject to:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad t = 0, \ldots, N-1 \tag{2}$$

$$d(\mathbf{x}_t, \mathcal{P}_k) \geq r_{\min}, \quad t = 0, \ldots, N \tag{3}$$

$$\mathbf{u}_t \in \mathcal{U}, \quad t = 0, \ldots, N-1 \tag{4}$$

where J is a composite cost function parametrized by $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$, f is the discrete-time robot dynamics, $d(\mathbf{x}_t, \mathcal{P}_k)$ is the minimum Euclidean distance from the predicted state $\mathbf{x}_t$ to any obstacle in the current point cloud, and $r_{\min}$ is the minimum safety clearance. The objective is to drive the robot from its current pose $\mathbf{x}_0$ to the goal $\mathbf{x}_g$ while satisfying (2)–(4) in real time, without any a priori knowledge of the environment.

The performance of the MPC (1) depends critically on the parameter vector θ, which encodes cost weights for tracking accuracy, control effort, and obstacle avoidance. The meta-problem of identifying an optimal θ is formulated as the black-box optimization:

$$\theta^* = \arg\max_{\theta \in \Theta} \; \mathcal{J}(\theta) \tag{5}$$

where $\mathcal{J}(\theta)$ is a composite performance score obtained by running the full navigation system under parameters θ.

B. Related Work

1) Legged Robot Navigation: Navigation for legged robots has been studied extensively in the context of whole-body planning and terrain-aware locomotion. Recent approaches integrate deep learning-based terrain estimation with classical planning [4] or employ model-free reinforcement learning for end-to-end navigation [5]. These methods either require pre-built maps or suffer from difficult sim-to-real transfer. Our framework abstracts leg dynamics through the CHAMP hierarchical locomotion controller [6], which exposes a body-frame velocity interface and enables the navigation layer to treat the robot as a holonomic mobile agent, simplifying planning and control design.

For local navigation, fast sampling-based planners like RRT* [7] and adaptive variants such as APEI-RRT* [8] are highly efficient and widely adopted. While we deploy a grid-based A* planner for its predictable deterministic paths, our framework is fundamentally planner-agnostic; the adaptive MPC pipeline can seamlessly track reference trajectories generated by any sampling-based or heuristic algorithm.

2) MPC for Mobile Robot Navigation: MPC has been applied to mobile robot navigation across differential-drive platforms, UAVs [9], and quadrupeds. Nonlinear MPC formulations handle the full kinematic model but require efficient nonlinear programming (NLP) solvers; CasADi [10] with IPOPT [11] provides a suitable combination for our 5-second horizon at 10 Hz. Obstacle avoidance within MPC is commonly implemented via hard distance constraints or smooth barrier functions [12]; we adopt a differentiable sigmoid penalty evaluated on the nearest LiDAR returns, which preserves NLP sparsity and IPOPT convergence properties.

3) Bayesian Optimization for Robotics: Bayesian Optimization (BO) has become the standard method for black-box hyperparameter tuning in robotics, widely applied to locomotion, control, and planning [13]. The COAt-MPC framework [3] provides performance-guaranteed BO for MPC parameter optimization, while [14] analyzes the theoretical properties of TPE, the acquisition strategy we adopt for its efficiency in high-dimensional mixed spaces.

III. Methodology

A. System Architecture

The proposed navigation stack is composed of five interconnected modules illustrated in Fig. 2: (i) an *Odometry Bridge* that converts raw EKF odometry into a unified pose representation; (ii) a *Mapping Module* that maintains both a reactive semi-persistent Gaussian occupancy grid and a topological *Navigation Graph* inspired by WildOS [15], which encodes high-level connectivity between previously visited locations and enables global planning via Dijkstra search; (iii) a *Global-to-Local Planning Layer*, where the Navigation Graph provides waypoints to a rolling-horizon A* planner operating on the occupancy grid at 1 Hz; (iv) a *Nonlinear MPC Tracker* formulated with CasADi [10] and solved with IPOPT [11], generating dynamically feasible velocity setpoints and converting MPC lookahead states into body-frame `/cmd_vel` commands for the locomotion layer; and (v) an offline *Bayesian Optimisation Module* based on Tree-structured Parzen Estimator (TPE) and Gaussian Processes (GP), which tunes the hyperparameters to improve trajectory efficiency and tracking performance.

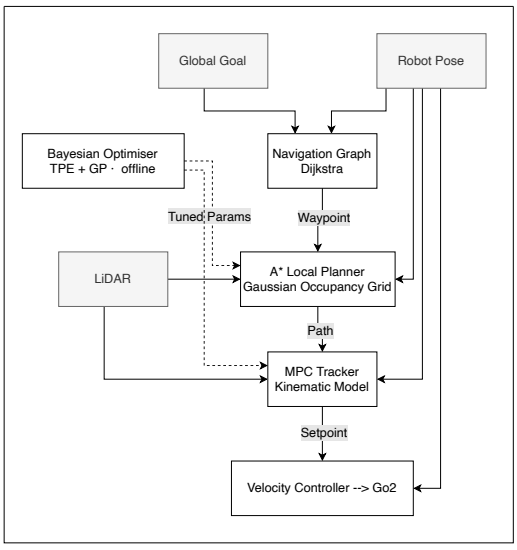


Fig. 2: System architecture of the proposed framework.

All modules communicate through ROS 2 topics, enabling full modularity and real-time operation at 10–20 Hz. The MPC parameter vector θ is adapted by the Bayesian Optimization pipeline in Section III-E.

B. Mapping

At each replanning cycle, a square local Gaussian Occupancy Grid of side $2h_w$ (default $h_w = 5\,\text{m}$, resolution $\Delta = 0.25\,\text{m/cell}$) is centered on the robot and rebuilt from scratch using only the current LiDAR point cloud $\mathcal{P}_k$. Each grid cell c encodes the probability of occupancy derived from the minimum Euclidean distance $d_{\min}(c, \mathcal{P}_k)$ from the cell center to any LiDAR return:

$$P(c) = 1 - \Phi\left(\frac{d_{\min}(c,\, \mathcal{P}_k)}{\sigma}\right) \tag{6}$$

where $\Phi(\cdot)$ is the standard normal CDF and $\sigma = 0.05\,\text{m}$ is the Gaussian spread parameter controlling obstacle inflation. Cells with $P(c) \geq P_{\text{thresh}} = 0.45$ are treated as hard obstacles; cells below this threshold incur a soft traversal penalty in A* proportional to their occupancy probability.

In addition to the reactive occupancy grid, the system maintains a persistent topological navigation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ corresponds to a semantically meaningful region of the environment and each edge $(u, v) \in \mathcal{E}$ represents a navigable transition weighted by the traversal cost estimated from the current occupancy grid. As the robot explores, new nodes are instantiated when the robot enters a previously unvisited region, and the graph is updated online.

C. Rolling-Horizon Path Planning

Since the occupancy grid covers only a local window, a rolling-horizon strategy is employed for distant goals. At each replanning tick the local planning target is:

$$\mathbf{x}_{\text{local}} = \begin{cases} \text{cell}(\mathbf{x}_g) & \text{if } \mathbf{x}_g \in \text{grid} \\ \text{ray}(\mathbf{x}_0 \to \mathbf{x}_g) \cap \partial\text{grid} & \text{otherwise} \end{cases} \tag{7}$$

The ray is parametrically clipped to the grid boundary, allowing the robot to advance one grid-width at a time. If the local target falls in an occupied cell, a breadth-first search (BFS) identifies the nearest free cell as a proxy, guaranteeing A* always receives a valid goal.

A* is executed on an 8-connected grid with the following edge traversal cost:

$$g(n \to n') = c_{\text{move}} \cdot \Delta \cdot (1 + w_{\text{obs}} \cdot P(n')) \tag{8}$$

where $c_{\text{move}} \in \{1, \sqrt{2}\}$ for cardinal and diagonal moves respectively and w_{obs} is the soft obstacle penalty weight. The admissible Euclidean heuristic is:

$$h(n) = \Delta \cdot \sqrt{(i_x - g_x)^2 + (i_y - g_y)^2} \tag{9}$$

Cells with $P \geq P_{\text{thresh}}$ are treated as hard obstacles, while sub-threshold cells incur the soft penalty (8), steering paths away from obstacle proximity. The resulting path is resampled at uniform spacing $\Delta s = 0.20\,\text{m}$ and smoothed with a moving-average kernel of width 5 to remove grid-induced zigzag artifacts while preserving endpoints.

D. MPC Formulation

1) State, Control, and Dynamics: The robot is modeled as a holonomic kinematic agent in the plane, with state $\mathbf{x}_k = [p_x, p_y, \psi]^\top$ and control $\mathbf{u}_k = [v_x, v_y, \omega]^\top$. The discrete-time dynamics under Forward Euler integration with step $\Delta t = 0.1\,\text{s}$ are:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} p_{x,k} + (v_{x,k}\cos\psi_k - v_{y,k}\sin\psi_k)\,\Delta t \\ p_{y,k} + (v_{x,k}\sin\psi_k + v_{y,k}\cos\psi_k)\,\Delta t \\ \psi_k + \omega_k\,\Delta t \end{bmatrix} \tag{10}$$

2) Objective Function: The MPC solves the following finite-horizon NLP over $N = 50$ steps (5 s horizon) at 10 Hz:

$$J = \mathbf{e}_N^\top \mathbf{Q}_T \mathbf{e}_N + J_{\text{obs}}(\mathbf{x}_N) + \sum_{k=0}^{N-1} \Big[\mathbf{e}_k^\top \mathbf{Q} \mathbf{e}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k + R_{\text{jerk}} \|\mathbf{u}_k - \mathbf{u}_{k-1}\|^2 + J_{\text{obs}}(\mathbf{x}_k) \Big] \tag{11}$$

where $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_{\text{ref},k}$ is the tracking error with respect to the A*-derived reference, $\mathbf{Q} = \text{diag}(Q_{xy}, Q_{xy}, Q_\psi)$, $\mathbf{Q}_T = Q_T \cdot \mathbf{Q}$, and $\mathbf{R} = \text{diag}(R_v, R_v, R_\omega)$. The jerk term R_{jerk} penalizes rapid control changes to enforce smooth commands. The NLP is compiled once at startup using CasADi [10] symbolic differentiation; subsequent solves update only numeric parameter values, enabling real-time feasibility. The MPC predicted trajectory is tracked by a proportional controller running at 20 Hz, which generates `/cmd_vel` commands for the CHAMP locomotion layer.

3) Sigmoid Obstacle Barrier: Obstacle avoidance is enforced as a smooth cost penalty through a logistic sigmoid barrier evaluated on the $M = 12$ nearest LiDAR returns $\{\mathbf{p}_j\}_{j=1}^{M}$ within a search radius of 3 m:

$$J_{\text{obs}}(\mathbf{x}_k) = \sum_{j=1}^{M} \frac{W}{1 + e^{\alpha\left(d(\mathbf{x}_k, \mathbf{p}_j) - r\right)}} \tag{12}$$

where $d(\mathbf{x}_k, \mathbf{p}_j) = \sqrt{(p_{x,k} - p_{j,x})^2 + (p_{y,k} - p_{j,y})^2 + \epsilon}$ with $\epsilon = 10^{-6}$ ensuring differentiability at zero distance. The numerically stable form $\frac{W}{2}\big(1 - \tanh\big(\frac{\alpha}{2}(d - r)\big)\big)$ is used in implementation to avoid overflow. Points beyond the search radius are replaced with far sentinels, keeping the NLP sparsity pattern constant and enabling warm starting. Here W is the barrier height, α controls steepness, and r is the safety radius. The control inputs are enforced via box constraints corresponding to the robot actuator limits.

4) Reference Trajectory and Warm Starting: The reference sequence $\{\mathbf{x}_{\text{ref},k}\}_{k=0}^{N}$ is generated by advancing along the A* path at cruise speed v_{ref} from the arc-length coordinate s_0 of the closest waypoint to the robot:

$$s_k = \min(s_0 + v_{\text{ref}} \cdot k \cdot \Delta t,\; L_{\text{path}}) \tag{13}$$

Reference position $\mathbf{p}_{\text{ref},k}$ is linearly interpolated along the path segment containing s_k; reference yaw is the path tangent $\psi_{\text{ref},k} = \text{atan2}(\Delta y, \Delta x)$. The previous IPOPT solution is shifted by one step and used as the warm-start initial guess.

E. Bayesian Optimization for MPC Tuning

1) Composite Performance Score: The MPC parameter vector $\theta \in \Theta \subset \mathbb{R}^{11}$ comprises: position tracking weights (Q_x, Q_y), yaw tracking weight Q_ψ, terminal cost weight Q_T, control effort weights $(R_{v_x}, R_{v_y}, R_\omega)$, jerk regularization weight R_{jerk}, obstacle cost weight W_{obs}, and obstacle shaping parameters $(\alpha_{\text{obs}}, r_{\text{obs}})$. Each parameter is associated with predefined lower and upper bounds, defining the search space as a bounded domain $\Theta = \prod_i [\theta_i^{\min}, \theta_i^{\max}]$.

Each function evaluation requires launching a full Gazebo simulation, making gradient-free, sample-efficient methods essential. The composite score $\mathcal{J}(\theta)$ aggregates different navigation scenarios $s \in S$ with scenario weights w_s:

$$\mathcal{J}(\theta) = \sum_s w_s \cdot \mathcal{J}_s(\theta), \qquad \sum_s w_s = 1 \tag{14}$$

A scenario includes various types of trajectories imposed on the robot in simulation, enabling it to learn optimal parameters under diverse dynamic configurations. Each per-scenario score $\mathcal{J}_s$ integrates five metrics. Let $\mathbf{p}_0$, $\mathbf{p}_f$, and $\mathbf{g}$ denote the robot start, final, and goal positions. *Goal progress* $\phi = \max(0, (d_0 - d_f)/d_0)$ measures the fraction of initial distance closed, where $d_0 = \|\mathbf{p}_0 - \mathbf{g}\|$, $d_f = \|\mathbf{p}_f - \mathbf{g}\|$. *Path efficiency* $\eta = \min(1, d_0/L)$ is the ratio of the straight-line distance to the total traversed arc length L. *Control smoothness* $s = e^{-\bar{j}/2}$ uses the exponentiated negative mean second-order finite difference of the command sequence as a jerk proxy:

$$\bar{j} = \frac{1}{K-2} \sum_{k=1}^{K-2} |\Delta^2 \mathbf{u}_k|_1, \quad \Delta^2 \mathbf{u}_k = \mathbf{u}_{k+1} - 2\mathbf{u}_k + \mathbf{u}_{k-1} \tag{15}$$

Obstacle avoidance $\mathcal{O}$ uses per-scan minimum LiDAR distances $\{d_k^{\text{obs}}\}$ to define danger ($d < 0.3\,\text{m}$) and warning ($0.3 \le d < 0.6\,\text{m}$) zone fractions:

$$\mathcal{O} = 0.5\,(1 - f_{\text{danger}}) + 0.3\,(1 - f_{\text{warning}}) + 0.2\,\min\left(\tfrac{\bar{d}}{2}, 1\right) \tag{16}$$

where $\bar{d}$ is the mean clearance capped at 2 m. *Time efficiency* $\tau = \min(1, T_{\text{ref}}/T_{\text{elapsed}})$ with $T_{\text{ref}} = d_0/0.5$ is only defined when the goal is reached. Two scoring branches are defined:

$$\mathcal{J}_s = \begin{cases} 0.35 + 0.2\,\eta + 0.15\,s + 0.2\,\mathcal{O} + 0.1\,\tau & \text{if goal reached} \\ 0.25\,\phi + 0.1\,\eta + 0.1\,s + 0.15\,\mathcal{O} & \text{else} \end{cases} \tag{17}$$

2) Tree-Structured Parzen Estimator: The offline BO employs Tree-Structured Parzen Estimators (TPE) [14], well-suited for high-dimensional mixed parameter spaces. TPE partitions the observed scores at the $\gamma = 0.15$ quantile threshold y^* and models two marginal densities:

$$\ell(\theta) = p(\theta \mid \mathcal{J}(\theta) < y^*), \quad g(\theta) = p(\theta \mid \mathcal{J}(\theta) \ge y^*) \tag{18}$$

Each marginal is estimated as a mixture of truncated Gaussians, one centred on each past observation. The acquisition is the Expected Improvement ratio:

$$\text{EI}(\theta) \propto \frac{\ell(\theta)}{g(\theta)} \tag{19}$$

The algorithm first draws $N_{\text{rand}} = 20$ trials uniformly (cold start), then iterates: refit ℓ and g, sample 20 candidates from $\ell(\theta)$, select the one with highest ℓ/g ratio, evaluate $\mathcal{J}$, and add to the dataset. This repeats until $N_{\max} = 120$ trials and the best configuration $\theta^*_{\text{offline}} = \arg\max_i \mathcal{J}(\theta_i)$ is deployed.

3) GP Surrogate for Sensitivity Analysis: An ARD Matérn-5/2 Gaussian Process is fitted to all accumulated observations for interpretability. The kernel is:

$$k(\theta, \theta') = \sigma_f^2 \left(1 + \sqrt{5}\,r + \tfrac{5}{3} r^2\right) e^{-\sqrt{5}\,r} + \sigma_n^2 \delta(\theta, \theta') \tag{20}$$

where $r = \sqrt{\sum_i (\theta_i - \theta_i')^2 / \ell_i^2}$ and ARD length scales $\{\ell_i\}$ are learned by maximizing the log marginal likelihood. The normalized inverse length-scale $\tilde{\ell}_i = (1/\ell_i)/\sum_j (1/\ell_j)$ quantifies parameter sensitivity: a short ℓ_i indicates that the score landscape varies rapidly with θ_i, marking it as decisive.

IV. Experimental Setup

All experiments are conducted on the Unitree Go2 quadruped. Locomotion is handled by the CHAMP hierarchical controller [6], which exposes a body-frame `/cmd_vel` velocity interface. A mid-range 3-D LiDAR provides filtered point clouds at 10 Hz. The full navigation stack uses Ros 2 and runs on a Jetson Orin Nano carried onboard, while the simulations are performed in Gazebo Fortress [16] with a planar-move plugin providing ground-truth odometry.

Three Gazebo worlds of increasing complexity are employed. Environments E1 and E2 serve as in-distribution training environments for offline Bayesian Optimization; E3 is held out as an out-of-distribution test scenario.

- **E1 – Open** (tuning): a sparse, low-obstacle environment representing easy navigation conditions.
- **E2 – Indoor Office** (tuning): a confined indoor environment with tables, narrow passages, doorways, and multiple rooms, presenting high navigation difficulty.
- **E3 – Warehouse** (testing): a held-out environment with dense shelves, narrow corridors, and complex passages, designed to assess out-of-distribution generalization.
- **E4 – Real Room** (testing): a real room of our laboratory arranged for the experiment with obstacles as furniture, boxes and moving people.

Each environment contains a representative set of navigation tasks of varying difficulty, including straight-line traversals, zig-zag paths, tight turns, navigation around large obstacles, and goal configurations requiring global exploration. An additional test with suddenly appearing dynamic obstacles evaluates robustness to pedestrians or vehicles in unstructured settings. The evaluation is composed of the following phases:

- **Phase 0 – Baseline:** MPC parameters are hand-tuned through a limited set of simulation trials and held fixed, which serves as the reference for all comparisons.
- **Phase 1 – Offline BO:** 120 BO trials are executed on E1 and E2 using the TPE method described in Section III-E, yielding the tuned vector θ^*_{BO}.
- **Phase 2 – Generalization:** Both configurations are evaluated on the unseen environment E3 to quantify out-of-distribution transfer of BO tuned parameters.

- **Phase 3 – Sim-to-Real:** We performed a limited test in a real environment where the robot had to reach predefined set of navigation tasks in the environment E4.

As evaluation metrics for the navigation experiments, we report success rate (SR, %), time to goal (TTG, s), path length (PL, m), and average MPC solve time (TSM, ms). Success rate (SR) is defined as the percentage of episodes in which the robot reaches the goal region within a predefined tolerance, without collision, and within a 5-minute time limit. Time to goal (TTG) measures the total elapsed time from the initial pose to goal attainment. Path length (PL) denotes the cumulative distance travelled along the executed trajectory. Finally, average MPC solve time (TSM) quantifies computational cost as the mean solver execution time per MPC iteration.

V. Results

A. Navigation Performance

As shown in Table I, the hand-tuned baseline performs adequately in the open environment (E1: PL = 9.64 m, TSM = 40.85 ms) but degrades in confined settings (E2: PL = 17.92 m, TSM = 46.78 ms), reflecting the sensitivity of fixed parameters to obstacle density. Offline BO on E1 and E2 consistently improves all metrics across every evaluated scenario. In the indoor office environment (E2), path length is reduced by 32.3% (17.92 m → 12.13 m) and average MPC solve time decreases by 49.2% (46.78 s → 23.77 s), indicating that BO-tuned parameters yield shorter, more computationally efficient trajectories. Critically, The BO-tuned configuration generalizes to the unseen warehouse environment (E3): path length drops by 44.5% (73.97 m → 41.07 m), despite E3 never being used during tuning, demonstrating strong out-of-distribution transfer. Figure 3 illustrates the MPC objective function (11) comparison, which is consistently lower.

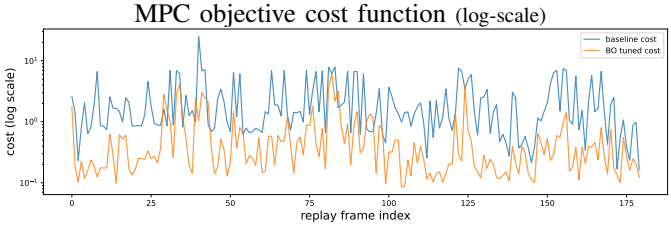
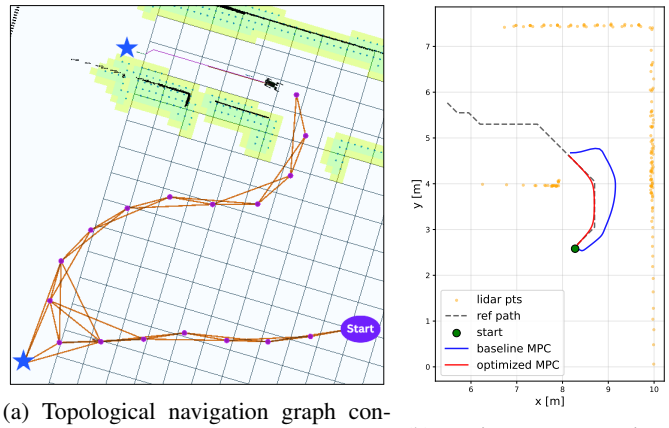


Fig. 3: Comparison of the MPC objective function (11) over the planning horizon between baseline and BO-tuned configurations in E3. Lower values indicate tighter path tracking and reduced obstacle cost, so overall better task performance.

Aggregated over E1–E3, BO reduces average path length by 38.7%, average MPC solve time by 25.3%, and time to goal by 53.0% (200.37 s → 94.25 s), while improving the overall success rate from 13/20 to 18/20. These gains confirm that optimizing across environments of contrasting difficulty produces robust, generalizing parameter configurations.

Figure 4 illustrates a representative trajectory comparison in E3, where the BO-tuned MPC follows a shorter, smoother path relative to the baseline.



(a) Topological navigation graph constructed online during the robot's exploration phase, where visited locations are nodes and edges encode regions connectivity. The purple circle marks the starting point, the blue stars are the first reached goal and the current target. The robotic agent is maneuvering around obstacles toward the second goal.

(b) Trajectory comparison in E3 showing A* reference (grey dotted), baseline MPC (blue), and BO-optimized MPC (red). Current robot pose is green; LiDAR points of obstacles are shown in yellow.

Fig. 4: Navigation graph and MPC path planner comparison in the held out-of-distribution warehouse environment (E3).

Hardware deployment in the real laboratory room (E4) validates sim-to-real transfer: success rate improves from 5/10 to 8/10 using θ^*_{BO} without any additional on-robot or online tuning, confirming that simulation-tuned parameters transfer effectively to physical hardware and real-world obstacles.

B. MPC Parameter Sensitivity

We analyze MPC parameter importance via GP-based sensitivity estimation (Section III-E), using baseline and BO-tuned configurations. The BO convergence is shown in Fig. 5.

The highest sensitivity is observed for control jerk R_{jerk} and control effort R_v, indicating that dynamic regularization primarily governs solution stability. Obstacle weight W and solver-time effects follow, while tracking terms (Q_{xy}, Q_ψ) and failure penalty show low sensitivity. Overall, variability is driven by dynamic and constraint-related parameters, with tracking having limited influence. Compared to the baseline, BO reduces average sensitivity by 42.3%, yielding a smoother landscape that improves robustness and generalization.

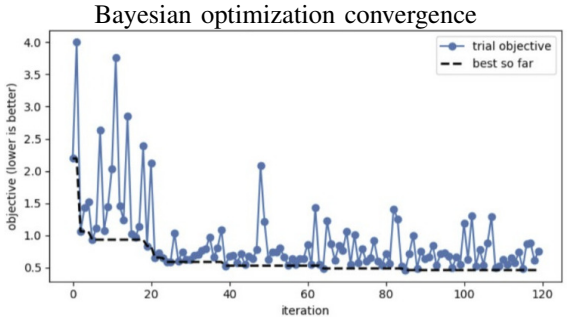


Fig. 5: Bayesian optimization (BO) convergence, showing the objective score $\mathcal{J}(\theta)$ versus trial index over 120 trials.

TABLE I: As navigation evaluation metrics, we report success rate (SR, % or success/trials, ↑), time to goal (TTG, s, ↓), path length (PL, m, ↓) and average time to solve MPC (TSM, ms, ↑). Note: ↑ denotes higher is better, ↓ lower is better.

	E1 (Open)		**E2 (Indoor Office)**		**E3*(Warehouse)**		**$E_{avg(1,2,3)}$ (Overall)**				**E4*(Real Room)**
Method	PL	TSM	PL	TSM	PL	TSM	PL	TSM	TTG	SR	SR
Baseline parameters θ_0	9.64	40.85	17.92	46.78	73.97	60.78	33.84	49.47	200.37	65.0%	5/10
BO tuned parameters θ^*_{BO}	9.03	38.68	12.13	23.77	41.07	48.41	20.74	36.95	94.25	90.0%	8/10
Percentage change $\Delta_{\%}$	-6.3	-5.3	-32.3	-49.2	-44.5	-20.4	-38.7	-25.3	-53.0	+38.5	+60.0

* Indicates held-out out-of-distribution environment.

VI. Discussion and Limitations

Limitations. The kinematic model treats the robot as a planar holonomic agent; significant slopes or steps require extension to 3-D dynamics. The BO search is bounded by the predefined space Θ, which may not include all the possible MPC parameters that could be tuned to find the optimal configuration, an a priori selection was necessary to avoid an extremely high dimensional optimization space.

Future Work. Three primary extensions are planned. First, an online BO loop warm-started from θ^*_{BO} will adapt MPC parameters incrementally during deployment via GP posterior updates and Expected Improvement acquisition [13], enabling environment-specific refinement for out-of-distribution settings without offline retraining. Second, a self-supervised Dynamic Graph CNN (DGCNN) encoder will provide scan-level, annotation-free MPC parameter conditioning from LiDAR embeddings. Third, the stack will be ported to the Unitree G1 bipedal humanoid and interfaced with Visual Language Action (VLA) [17] models for natural-language goal decomposition, with lifelong continual online learning to accumulate knowledge from the real world environment data.

VII. Conclusion

We have presented a map-free autonomous navigation framework combining rolling-horizon A* planning with nonlinear MPC tuned via offline Bayesian Optimization. The system integrates a Gaussian occupancy grid, a CasADi/IPOPT nonlinear MPC with differentiable sigmoid obstacle barriers, and a TPE-based BO pipeline that identifies near-optimal MPC weight configurations from a composite performance score.

Offline BO conducted on two training environments of different difficulty yields consistent improvements across all evaluated scenarios. The average length path decreases by 38.7% and the MPC average total solving time decreases by 25.2% relative to the hand-tuned baseline. The BO-tuned configuration generalizes to the held-out warehouse environment without additional tuning.

The modular architecture is platform-agnostic and designed for extension. Our ongoing research work targets online Bayesian Optimization for deployment-time adaptation, DGCNN for map-adaptive MPC parameter conditioning, and integration with Visual Language Action models for language-conditioned navigation across legged and wheeled platforms.

References

[1] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-body mpc and online gait sequence generation for wheeled-legged robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8388–8395.

[2] T. P. Nascimento, C. E. T. Dórea, and L. M. G. Gonçalves, "Nonholonomic mobile robots' trajectory tracking model predictive control: a survey," *Robotica*, vol. 36, no. 5, p. 676–696, 2018.

[3] A. G. Puigjaner, M. Prajapat, A. Carron, A. Krause, and M. N. Zeilinger, "Performance-driven constrained optimal auto-tuner for mpc," *IEEE Robotics and Automation Letters*, vol. 10, pp. 4698–4705, May 2025.

[4] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3019–3026, 2018.

[5] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[6] L. Technology, "Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control : implementation on the mit cheetah robot," *MIT Libraries (DSpace@MIT)*, 03 2014.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, 2011.

[8] T. F. Banfi, F. Dorati, N. Manzoni, and J. Martínez-Gómez, "Optimizing initial path finding in informed-rrt* with a novel map-adaptive sampling technique," in *2024 7th Iberian Robotics Conference (ROBOT)*, Nov 2024, pp. 1–6.

[9] D.-N. Bui, T. H. Khuat, M. D. Phung, T.-H. Tran, and D. L. Tran, "Model predictive control for optimal motion planning of unmanned aerial vehicles," in *2024 International Conference on Control, Robotics and Informatics (ICCRI)*, July 2024, pp. 1–6.

[10] J. Andersson, J. Gillis, G. Horn, J. Rawlings, and M. Diehl, "Casadi—a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2018.

[11] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[12] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. Ieee, 2019, pp. 3420–3431.

[13] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1, pp. 5–23, 2016.

[14] S. Watanabe, "Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance."

[15] H. Shah, E. Tevere, D. Atha, M. Kaufmann, S. Khattak, M. Patel, M. Hutter, J. Frey, and P. Spieler, "Wildos: Open-vocabulary object search in the wild," 2026.

[16] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. Ieee, 2004, pp. 2149–2154.

[17] B. e. a. Zitkovich, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds. PMLR, 2023, pp. 2165–2183.