

# Collision-Free Navigation of Mobile Robots via Quadtree-Based Model Predictive Control

Osama Al Sheikh Ali, Sotiris Koutsoftas, Ze Zhang, Knut Åkesson, Emmanuel Dean

**Abstract**—This paper presents an integrated navigation framework for Autonomous Mobile Robots (AMRs) that unifies environment representation, trajectory generation, and Model Predictive Control (MPC). The proposed approach incorporates a quadtree-based method to generate structured, axis-aligned collision-free regions from occupancy maps. These regions serve as both a basis for developing safe corridors and as linear constraints within the MPC formulation, enabling efficient and reliable navigation without requiring direct obstacle encoding. The complete pipeline combines safe-area extraction, connectivity graph construction, trajectory generation, and B-spline smoothing into one coherent system. Experimental results demonstrate consistent success and superior performance compared to baseline approaches across complex environments.

## I. INTRODUCTION

Autonomous Mobile Robots (AMRs) are becoming pivotal across industrial domains, requiring precision, flexibility, and adaptability [1]. Safe and efficient navigation in such environments typically mandates integrating three core components: global path planning, trajectory generation, and control for trajectory tracking [2], which are usually decoupled. A global planner, such as A\* or RRT, generates a geometric path without accounting for robot kinematics and dynamics, while a local controller attempts to track it [3]. This separation can lead to violations of safety or dynamic feasibility constraints [4], [5], as the geometric path may require maneuvers exceeding the robot's acceleration, steering, or obstacle avoidance capabilities when executed in real environments.

To overcome this decoupling between planning and control, recent work has adopted Model Predictive Control (MPC), which unifies both processes through receding-horizon optimization under dynamic and safety constraints [6], [7]. However, representing obstacles as hard constraints may yield non-convex conditions that hinder real-time performance and robustness [5], [8]. Conventional MPC explicitly encodes convex polygonal obstacles as constraints, causing computational spikes or infeasibility in cluttered environments. Alternative variants mitigate this issue using penalty methods or by tracking precomputed paths from global planners [3], [9], but remain sensitive to parameter tuning and prone to local minima. Deep Reinforcement Learning (DRL) offers a data-driven alternative [10], [11],

This work is financed by the European Union's Horizon European grant euRobin (GA no. 101070596), and was partially supported by the Chalmers Production Area of Advance (AoA) project.

All authors are with Electrical Engineering, Chalmers University of Technology, 41296 Gothenburg, Sweden. Contact: zhze@chalmers.se

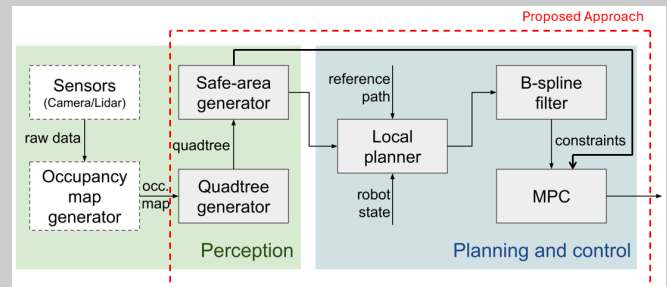


Fig. 1: The proposed framework integrates quadtree-based safe-area generation from occupancy grid map perception with local planning and control.

yet its black-box nature limits interpretability and trustworthiness in safety-critical applications.

Several studies leverage structured environment representations to improve trajectory planning efficiency. In [12], safe flight corridors are constructed using octree decomposition and generate trajectories via quadratic programming, while requiring iterative extrema checking to satisfy corridor constraints. It is extended to multi-agent coordination through relative corridors based on Bernstein polynomial properties [13]. In [14], a grid-based stochastic MPC is formulated that thresholds probabilistic occupancy maps and computes convex hulls for constraint enforcement.

In this work, the proposed quadtree-based approach enables one-shot convex constraint generation through hierarchical spatial decomposition and efficient region merging, which doesn't require iterative constraint refinement [12], or relies on probabilistic thresholding [14]. This design directly addresses the persistent challenge of achieving feasible navigation with formal safety guarantees in environments containing non-convex obstacles. In general, the framework integrates perception, mapping, trajectory generation, and control within an MPC formulation as demonstrated in Fig. 1. The occupancy map is decomposed into axis-aligned safe regions, which are subsequently merged into larger convex areas that serve a dual purpose: they define a connectivity-aware reference trajectory and are simultaneously encoded as linear inequality constraints in the MPC problem. This unified use of precomputed safe areas for both path generation and constraint enforcement eliminates the need to model obstacles directly, transforming the inherently non-convex obstacle-avoidance task into a convex optimization problem. As a result, the approach guarantees recursive feasibility, improves computational stability, and enhances robustness in complex, cluttered environments.

The main innovations of this work are threefold: (1) a safe-

area MPC framework for dynamic feasibility and safety by tightly coupling all navigation components; (2) a quadtree-derived convex constraint embedding that makes obstacle avoidance tractable; and (3) a dual-layer MPC cost structure that combines hard safety constraints with soft penalty margins to improve resilience against modeling uncertainties.

## II. PRELIMINARIES

For mobile robots, MPC utilizes their motion model to anticipate future states and achieve certain objectives while satisfying system and safety constraints. At each control cycle, given the horizon  $N$ , it solves:

$$\min_{\mathbf{u}_{0:N-1}} \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + \ell_N(\mathbf{x}_N), \quad (1)$$

$$\text{s.t. } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad (2)$$

$$\mathbf{x}_k \in \mathcal{X}, \quad (3)$$

$$\mathbf{u}_k \in \mathcal{U}, \quad (4)$$

where  $\mathbf{x}_k$  and  $\mathbf{u}_k$  are the robot state and input at time step  $k$ ,  $f(\cdot)$  is the robot's motion model,  $\ell(\mathbf{x}_k, \mathbf{u}_k)$  is the stage cost, and  $\ell_N(\mathbf{x}_N)$  is a terminal cost. The constraints  $\mathcal{X}$  and  $\mathcal{U}$  encode physical and safety limits. Only the first input  $\mathbf{u}_0$  is applied at each step (receding horizon), allowing MPC to adapt to changing conditions while optimising future behaviour.

## III. METHODOLOGY

### A. Quadtree-based Safe Area Generation

Occupancy grids divide the environment into uniform cells indicating free or occupied space [2], enabling reliable segmentation and collision checking. Although this uniform representation is effective for basic mapping and navigation, it becomes inefficient in large or unevenly structured environments due to redundant data in homogeneous regions. To overcome this limitation, quadtrees extend the occupancy grid concept by hierarchically subdividing non-uniform regions into quadrants while merging homogeneous areas, providing a more compact and flexible spatial representation. By recursively partitioning the occupancy map (Fig. 2), the quadtree clusters contiguous free spaces into coherent regions and continues this hierarchical subdivision until each region reaches homogeneity or the minimum resolution  $\delta_{\min}$ , yielding leaf nodes that represent free or occupied areas. This hierarchical representation preserves spatial relationships between navigable regions and inherently compresses map data while maintaining adjacency information between free regions. The overall quadtree-based safe area generation procedure is summarised in Algorithm 1.

### B. Safe Area Merging and Connectivity

While this fine-grained representation captures local spatial structure accurately, it introduces significant challenges for downstream planning. In particular, the large number of small and potentially disconnected safe regions leads to a dense and complex connectivity graph. This not only increases the computational cost of graph construction but

**Algorithm 1:** Quadtree-based safe area generation and merging procedure.

**Input:** Occupancy map of the environment, desired minimum cell size

**Output:** Quadtree representation with safe area connectivity graph

Initialize `QuadTreeNode` for the full map.

`build_quadtree(node)`:

- 1) If the region is free or occupied, mark as leaf.
- 2) If free, add to the connectivity graph.
- 3) Otherwise, split into quadrants, repeat for each child.
- 4) Then, call `update_connectivity(node)`.

`update_connectivity(node)`:

- 1) For each adjacent free region sharing a full edge, add an edge in the connectivity graph weighted by centre distance.

`merge_adjacent_nodes()`:

- 1) For each free region, check its neighbors.
- 2) If two regions can merge into a valid rectangle, combine them.
- 3) Update connectivity and repeat until no more merges are possible.

also results in longer planning times, as the search space becomes cluttered with redundant transitions.

For motion planners, this fragmentation means more region boundaries must be considered, each potentially introducing additional constraints during optimisation. As a result, the planner may produce conservative or suboptimal trajectories due to restricted manoeuvring flexibility. To address this, merging areas reduces data redundancy and enables efficient multi-scale and variable-resolution mapping [15], [16]. The `merge_adjacent_nodes` function is applied, which consolidates neighbouring rectangular free cells into larger convex regions. Two cells are merged if they share a complete edge and their union forms a valid rectangle. This reduces region fragmentation, resulting in fewer transitions and a more tractable planning space for MPC. Following the merging process, the `update_connectivity` function reconstructs the connectivity graph by linking adjacent free regions. This yields a simplified topological map where edges represent direct, obstacle-free traversals between safe areas. The effectiveness of this process is illustrated in Fig. 2, where the initial dense cell map is transformed into a more coherent and navigable environment.

### C. Trajectory Generation through collision-free areas

To generate a smooth, collision-free trajectory based on the quadtree-derived safe corridors, the process comprises three stages: (i) discrete waypoint selection, (ii) interpolation of waypoints into a dense reference path, and (iii) reference trajectory generation via B-spline smoothing.

**Waypoints generation:** The first stage constructs a sequence of waypoints that connect the start and goal through

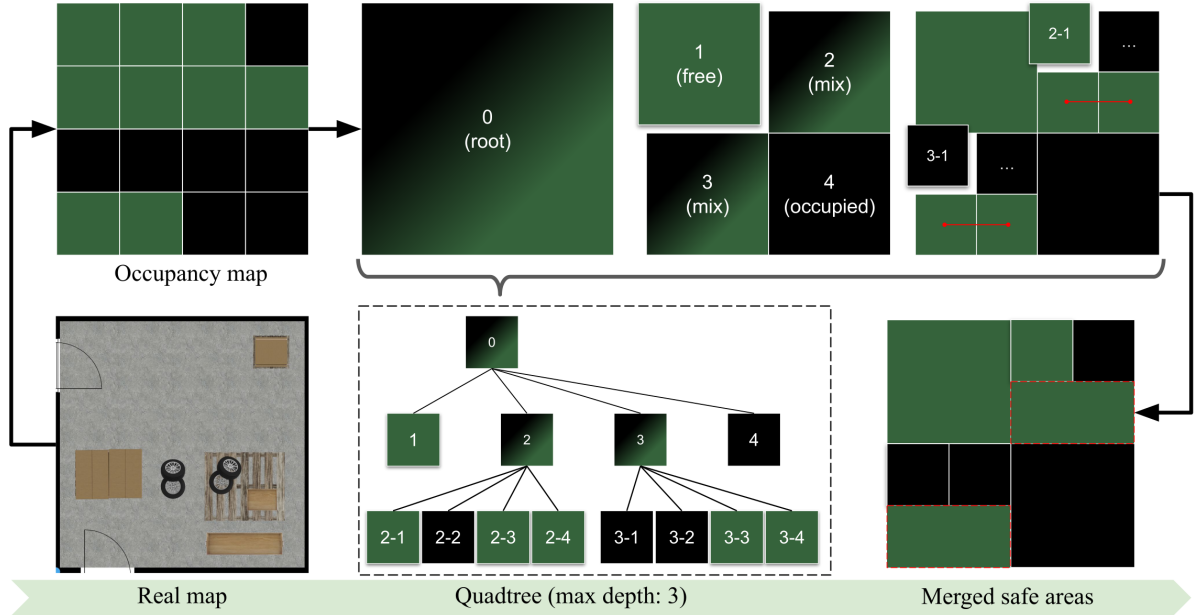


Fig. 2: Illustration of the safe area generation: From the real-world map to occupancy map generation, quadtree decomposition (nodes are indexed with different colors representing their states), and merged safe areas enabling efficient trajectory planning.

adjacent safe polygons. The sequence starts with the robot's current position  $p_0$  and ends at the goal position  $p_g$ . The system identifies the safe polygons containing the start and goal (or the nearest ones if outside). If these polygons are different, the goal point is projected to the safe area adjacent to the start polygon. Specifically, this procedure identifies the closest point on the polygon boundary to the input goal and computes a small inward offset along the estimated normal vector to ensure the projection lies strictly inside the safe region. Fig. 3 illustrates the waypoints generation procedure, with large purple-coloured points marking the start and projected goal waypoints.

**Waypoints interpolation:** Given a sequence of waypoints  $\{w_i\}_{i=0}^M$  starting at  $w_0 = p_0$  and ending at  $w_M = p_g$ , the path is parametrised by its cumulative arc-length. Starting with  $s_0 = 0$  and incrementally adding segment lengths  $s_i = s_{i-1} + \|w_i - w_{i-1}\|$ , the total length is  $s_M$ . For any  $s \in [0, s_M]$ , the segment index  $i$  is located such that  $s \in [s_i, s_{i+1})$ . Then, a normalised interpolation ratio can be defined, and the interpolated point can be computed:

$$w'(s) = (1 - \alpha(s)) w_i + \alpha(s) w_{i+1}, \quad (5)$$

$$\text{where } \alpha(s) = (s - s_i) / (s_{i+1} - s_i) \quad (6)$$

Note that  $w'(s)$  always lies inside the convex hull of  $w_i$  and  $w_{i+1}$ . Since each segment is contained in the corridor,  $w'(s) \in \text{conv}(w_i, w_{i+1}) \subseteq \mathcal{S}$ , guarantees collision-free interpolation. The arc-length parametrisation also provides a well-defined domain  $[0, s_M]$  for sampling, using uniform steps in  $s$  ensures that interpolated points are distributed evenly along the path regardless of waypoint spacing.

**B-spline smoothing:** To obtain a continuously differentiable trajectory with reduced curvature and to ensure compliance with the robot's dynamics, a cubic B-spline

$S(\mu)$  is fitted to the densified reference path. With  $\{p_j\}_{j=1}^M$  denoting the interpolated positions sampled uniformly along the arc-length parameter  $s$ . The spline control points are determined by minimising the regularised least-squares cost:

$$J = \sum_{j=1}^M \|S(\mu_j) - p_j\|^2 + \lambda \int_0^1 \|S''(\mu)\|^2 d\mu, \quad (7)$$

where  $\mu \in [0, 1]$  is the spline parameter and  $\lambda > 0$  weights the smoothness term. The first term enforces closeness to the reference path, while the second penalises high curvature. Once the B-spline is computed, its samples form the reference trajectory are used by the MPC.

#### D. Safe Area Constraints and Safety Guarantee

As a primary contribution, the incorporation of safety-area linear inequality constraints into the MPC formulation enforces navigation within the designated safe areas. Each rectangular safe region is represented as a set of four linear inequalities, which are imposed as hard constraints. These are expressed in Eq. (8), where  $A_{\text{safe},k}$  and  $b_{\text{safe},k}$  define the spatial boundaries at step  $k$ , and  $X(t)$  is the system state:

$$A_{\text{safe},k} X(t) \leq b_{\text{safe},k}. \quad (8)$$

To improve resilience against modelling errors or unexpected dynamics, a soft constraint in the form of a penalty term  $J_{SA}$  is introduced. This term quantifies deviations from the safe regions, encouraging trajectories that remain within convex boundaries defined as:

$$H_{n,m} = \{x \in \mathbb{R}^2 : b_{n,m} - a_{n,m}^\top x > 0\}, \quad (9)$$

where  $h_{n,m}(\mathbf{x}) = b_{n,m} - a_{n,m}^\top \mathbf{x}$  evaluates the robot's relative position to each boundary. Safety is guaranteed when:

$$\prod_{n=1}^N \sum_{m=1}^4 [h_{n,m}(\mathbf{x})]_-^2 = 0, \quad (10)$$

and is penalized when violated via the smooth cost function:

$$J_{SA}(\mathbf{x}) = \prod_{n=1}^N \sum_{m=1}^4 [q_{SA}(n) \cdot h_{n,m}(\mathbf{x})]_-^2, \quad (11)$$

where  $[h]_- = \min(0, h)$ , and  $q_{SA}(n)$  controls the priority of each safe area.

By embedding  $J_{SA}$  into the MPC cost, the planner not only avoids unsafe regions but also actively penalizes proximity to their boundaries. This dual-layer formulation of hard linear constraints and soft penalties represents a distinctive integration of safety considerations into MPC, ensuring that all planned actions remain within validated safe zones. Crucially, the inclusion of  $J_{SA}$  acts as a safety guarantee within the MPC optimisation. Even in uncertainty or dense environments, the generated control actions strictly adhere to safety boundaries, enabling the robot to operate with full assurance of collision-free behaviour at every step.

### E. Full MPC Formulation

For the robot's discrete state space, at time step  $k$ , the state vector  $\mathbf{x}_k = [x_k, y_k, \theta_k]^\top$  represents the robot's position  $(x_k, y_k)$  and its heading angle  $\theta_k$  in Cartesian space, while the control input  $\mathbf{u}_k = [v_k, \omega_k]^\top$  consists of the linear speed  $v_k$  and angular velocity  $\omega_k$ . The motion model is the unicycle model, assuming no wheel slip:

$$f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} x_k + v_k \cos(\theta_k) \Delta t \\ y_k + v_k \sin(\theta_k) \Delta t \\ \theta_k + \omega_k \Delta t \end{bmatrix}, \quad (12)$$

where  $\Delta t$  is the sampling time. The key contribution here is the incorporation of quadtree-derived safe areas directly into the MPC constraints. Each rectangular safe region from the quadtree decomposition is encoded as four linear inequalities, transforming the non-convex collision avoidance problem into a set of convex constraints. The complete MPC problem incorporating safe area navigation is formulated as:

$$\begin{aligned} \min_{\mathbf{u}_{0:N-1}} \quad & J_N + \sum_{k=0}^{N-1} [J_R(k) + J_{SA}(\mathbf{x}_k)] \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \dots, N-1, \\ & \mathbf{u}_k \in [\mathbf{u}_{\min}, \mathbf{u}_{\max}], \quad k = 0, 1, \dots, N-1, \\ & \frac{\Delta \mathbf{u}_k}{\Delta t} \in [\dot{\mathbf{u}}_{\min}, \dot{\mathbf{u}}_{\max}], \quad k = 0, 1, \dots, N-1, \\ & \underbrace{\mathbf{A}_{\text{safe},k} \mathbf{x}_k \leq \mathbf{b}_{\text{safe},k}}_{\text{Quadtree-derived constraints}}, \quad k = 0, 1, \dots, N, \end{aligned} \quad (13)$$

where  $\mathbf{x}_0$  is the current state, the terminal cost is  $J_N = \|\mathbf{x}_N - \tilde{\mathbf{x}}_N\|_{Q_N}^2$  with the penalty value  $Q_N$ . The quadtree information is embedded through the constraint,  $\mathbf{A}_{\text{safe},k} \mathbf{x}_k \leq \mathbf{b}_{\text{safe},k}$ , where  $\mathbf{A}_{\text{safe},k} \in \mathbb{R}^{4 \times 2}$  contains the normal vectors of the four edges of the rectangular safe area at prediction step

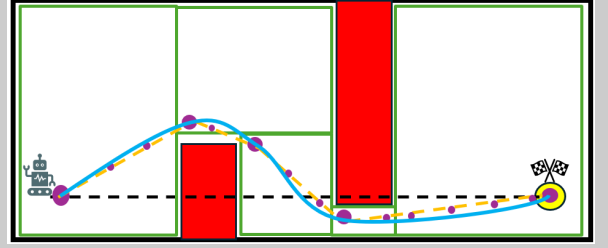


Fig. 3: Trajectory generation. Green rectangles are safe regions, and red blocks are obstacles. The dark dashed line is the original reference path. Large purple points are waypoints from greedy search, and small ones are interpolated. The yellow polyline forms the new reference path, and the blue curve is the B-spline-smoothed trajectory.

$k$ , and  $\mathbf{b}_{\text{safe},k} \in \mathbb{R}^4$  contains the corresponding offset values, which parameters are extracted from the quadtree leaf nodes identified as collision-free during the safe area generation phase. The running cost  $J_R(k)$  is given by:

$$J_R(k) = \|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|_{Q_x}^2 + \|\mathbf{u}_k - \tilde{\mathbf{u}}_k\|_{Q_u}^2 + \|\mathbf{u}_k - \mathbf{u}_{k-1}\|_{Q_a}^2, \quad (14)$$

which balances trajectory tracking, control accuracy, and smoothness, where  $\|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|_{Q_x}^2$  minimises the deviation from the desired state,  $\|\mathbf{u}_k - \tilde{\mathbf{u}}_k\|_{Q_u}^2$  penalises control signal deviations, and  $\|\mathbf{u}_k - \mathbf{u}_{k-1}\|_{Q_a}^2$  ensures smooth transitions between control inputs. The weighting matrices  $Q_x$ ,  $Q_u$ , and  $Q_a$  tune the importance of each term. The control input  $u_k$  is constrained within a specified range  $[\mathbf{u}_{\min}, \mathbf{u}_{\max}]$ , and its rate of change  $\frac{\Delta \mathbf{u}_k}{\Delta t}$  is also bounded by  $[\dot{\mathbf{u}}_{\min}, \dot{\mathbf{u}}_{\max}]$ . Finally, the penalty constraint,  $\mathbf{A}_{\text{safe},k} \mathbf{X}(t) \leq \mathbf{b}_{\text{safe},k}$ , keeps the robot within predefined safe limits.

## IV. IMPLEMENTATION AND RESULTS

The proposed safe-area MPC framework is evaluated under diverse scenarios. All experiments were conducted with an AMD Ryzen 7 5800H CPU, with randomized initial orientation to assess robustness. The MPC problem is solved by the PANOC solver [6] for real-time performance, and its weighting matrices were selected to prioritize adherence to the reference path while maintaining smooth movement. In this work, all experiments are in static environments, so the quadtree is computed offline in advance.

### A. Evaluation Against Explicit Obstacle Avoidance

To evaluate the effectiveness of the proposed method, we compare it with a conventional MPC baseline that represents established trajectory planners, such as in [4], [17], [18]. This baseline follows a nonlinear MPC formulation that plans motion over a finite horizon while minimizing a trajectory cost and satisfying collision avoidance constraints. The baseline uses the same kinematic model and tuning parameters as the proposed SA-MPC. Obstacles are explicitly modelled as convex polygons and, in this implementation, are incorporated into the MPC formulation through terms in the cost function rather than as hard feasibility constraints [19].

The framework was tested across five scenarios, as shown in Fig. 4, ranging from simple convex obstacles to complex

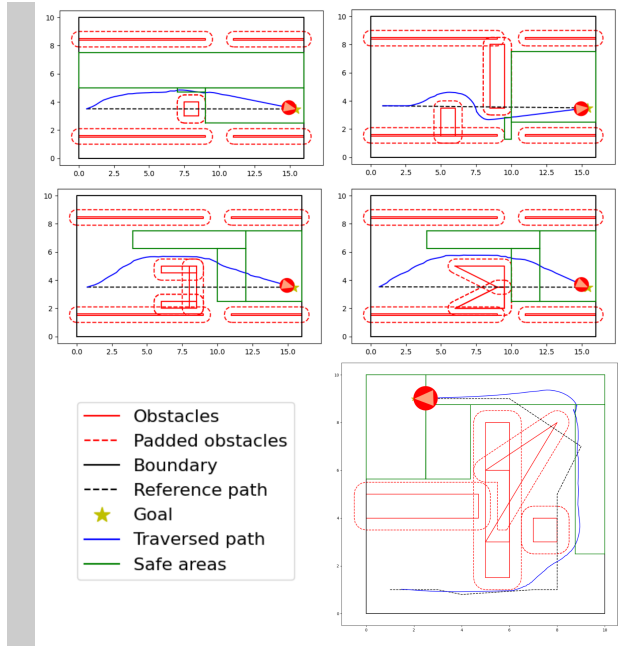


Fig. 4: SA-MPC navigation across test scenarios with quadtree-derived safe areas (green)

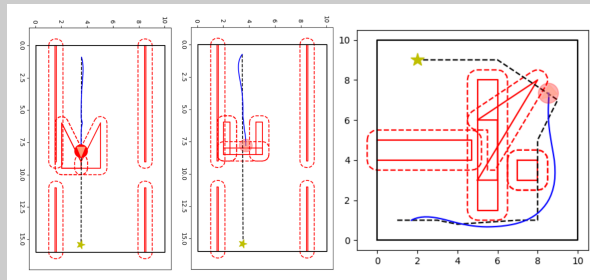


Fig. 5: Baseline MPC performance showing failures in non-convex and complex scenarios

non-convex geometries. In the narrow dual-obstacle case, the controller generates smooth trajectories and accurately navigates through narrow corridors. In U-shaped and V-shaped non-convex scenarios, the safe-area decomposition helps the trajectory avoid getting trapped in local minima, which is a common failure point for traditional approaches. It also ensures that necessary deviations remain smooth. Finally, in the cluttered scenario with multiple obstacles of varying shapes, the proposed method maintains successful navigation and computational efficiency. In contrast, Fig. 5 shows the performance of the reference controller in complex environments, where the robot gets stuck at local minima.

Table I provides a detailed comparison illustrating the performance advantages of the proposed approach. While baseline MPC succeeds only with convex obstacles, our method achieves 100% success across all scenarios. For non-convex obstacles, baseline MPC exhibits computational spikes and fails to find solutions, whereas SA-MPC maintains consistent performance with very low maximum computation times, including the full pipeline overhead.

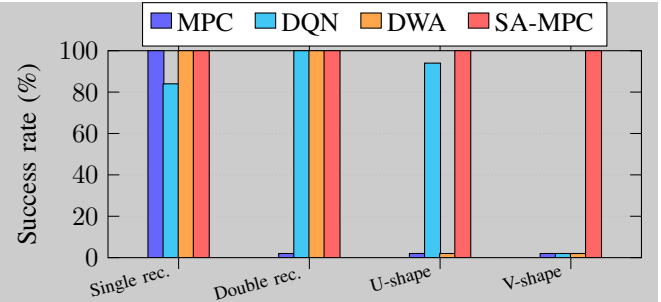


Fig. 6: Success rate across four scenarios. Zero-height bars indicate 0% success.

### B. Extensive Comparative Study

Apart from the baseline MPC method, a learning-based method and a sampling-based method are also adopted for comparison. As demonstrated in Fig. 6, success rates across four scenarios are compared against baseline MPC, learning-based Deep Q-Network (DQN) method, and sampling-based Dynamic Window Approach (DWA). SA-MPC achieves 100% success across all scenarios. Baseline MPC handles only simple convex cases, failing in complex geometries where the solver encounters infeasibility. DQN shows moderate performance but struggles with precise manoeuvring, which causes hesitation and collision when encountering obstacles with sharp corners [20]. DWA excels in simple scenarios but cannot handle non-convex obstacles, since it has limited flexibility for necessary detours. SA-MPC's superior performance stems from transforming non-convex collision avoidance into convex constraints via quadtree-derived safe areas, guaranteeing recursive feasibility.

## V. DISCUSSION

The experimental results demonstrate that the proposed SA-MPC framework outperforms the baseline MPC approach across all tested scenarios, in terms of both efficiency and safety, particularly in environments with complex non-convex obstacles. Its core innovation is using convex collision-free regions from a hierarchical quadtree decomposition both to generate trajectories and to impose soft constraints within the MPC. This transforms non-convex obstacle avoidance into a structured, efficient pipeline that avoids explicit obstacle modelling, unlike classical MPC methods that encode obstacles as non-linear constraints, signed-distance penalties, or discrete exclusion zones. The generation of collision-free areas guarantees recursive feasibility at each step, enabling MPC to compute valid control actions even in non-convex scenarios. As shown in Table I, our system consistently produces collision-free, feasible trajectories with low optimisation latency. While a baseline MPC comparison is provided, it lacks integrated trajectory generation and collision-free-area constraints. Therefore, these results underscore the enhanced capabilities of the proposed integration approach.

In the current development, the proposal framework has a structural limitation, which is the reliance on axis-aligned rectangular safe areas. This may introduce conservatism in

TABLE I: Performance comparison between Safe-Area (SA) MPC and baseline MPC approaches (averaged over 10 runs)

Scene	Obstacle Type	Method	MPC Computation / Full Pipeline* (ms/step)			Performance	
			Mean	Max	Std	Steps	Success (%)
Scene 1	Single rectangular obstacle	Baseline-MPC	23.7	273.5	38.8	66	100
		SA-MPC	<b>20.34</b> / 29.18	<b>39.56</b> / 52.67	<b>8.19</b> / 12.24	58	<b>100</b>
	Two rectangular obstacles	Baseline-MPC	24.5	75	17.9	71	100
		SA-MPC	<b>22.6</b> / 46.83	<b>87.04</b> / 108.4	<b>13.67</b> / 22.46	83	<b>100</b>
U-shape obstacle	Baseline-MPC	126.2	949.4	11.75	-	0	
	SA-MPC	<b>23.79</b> / 38.4	<b>59.77</b> / 88.76	<b>10.93</b> / 18.73	62	<b>100</b>	
V-shape obstacle	Baseline-MPC	97.8	694.5	196.2	-	0	
	SA-MPC	<b>22.8</b> / 38.4	<b>55.41</b> / 85.83	<b>11.6</b> / 21.85	64	<b>100</b>	
Scene 2	Complex map (mixed obstacles)	Baseline-MPC	19.6	58.1	6.9	-	0
		SA-MPC	<b>25.71</b> / 35.02	<b>48</b> / 57.3	<b>7</b> / 7.5	122	<b>100</b>

\*Full pipeline (only for SA-MPC) includes quadtree generation, safe area merging, and trajectory planning.

environments where free-space boundaries are highly irregular or diagonally oriented, and result in suboptimal paths or reduced manoeuvrability. Furthermore, the potential extension to dynamic settings will require efficient incremental updates of the quadtree and its connectivity graph, as well as providing real-time guarantees, which is a challenging task and an important future work.

## VI. CONCLUSION

This work presents an integrated navigation framework that unifies perception, environment representation, trajectory planning, and MPC through a quadtree-based safe-area decomposition. By converting obstacle avoidance into convex constraints, it ensures recursive feasibility, efficiency, and robustness without heuristic penalties. The direct integration of quadtree-derived safe areas into the MPC controller simplifies obstacle handling, enhances interpretability, and maintains strong performance across complex non-convex environments—offering a fast, reliable, and verifiable navigation pipeline for real-world deployment.

Future work will explore dynamic updates to the quadtree to handle moving obstacles, as well as extensions to multi-robot coordination. Real-world deployment will also be pursued to validate robustness under practical constraints.

## REFERENCES

- [1] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020.
- [2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots (2nd Edition)*. England: MIT Press, 2011.
- [3] J. Berlin, G. Hess, A. Karlsson, W. Ljungbergh, Z. Zhang, K. Åkesson, and P.-L. Götvald, "Trajectory generation for mobile robots in a dynamic environment using nonlinear model predictive control," in *CASE*. IEEE, 2021, pp. 942–947.
- [4] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, 2023.
- [5] A. Haffemayer, A. Jordana, M. Fourmy, K. Wojciechowski, G. Saurel, V. Petrik, F. Lamiroux, and N. Mansard, "Model predictive control under hard collision avoidance constraints for a robotic arm," in *UR*. IEEE, 2024, pp. 701–706.
- [6] L. Stella, A. Themelis, P. Sotasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *CDC*. IEEE, 2017, pp. 1939–1944.
- [7] S. V. Raković, S. Zhang, H. Sun, and Y. Xia, "Model predictive control for linear systems under relaxed constraints," *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 369–376, 2023.
- [8] Z. Zhang, G. Hess, J. Hu, E. Dean, L. Svensson, and K. Åkesson, "Future-oriented navigation: Dynamic obstacle avoidance with one-shot energy-based multimodal motion prediction," *IEEE Robotics and Automation Letters*, vol. 10, no. 8, pp. 8043–8050, 2025.
- [9] B. Hermans, P. Patrinos, and G. Pipeleers, "A penalty method based approach for autonomous navigation using nonlinear model predictive control," *IFAC Conference on Nonlinear Model Predictive Control*, vol. 51, no. 20, pp. 234–240, 2018.
- [10] C. Glanois, P. Weng, M. Zimmer, D. Li, T. Yang, J. Hao, and W. Liu, "A survey on interpretable reinforcement learning," *Machine Learning*, vol. 113, no. 8, p. 5847–5890, 2024.
- [11] G. A. Vouros, "Explainable deep reinforcement learning: State of the art and challenges," *ACM Comput. Surv.*, vol. 55, no. 5, Dec. 2022.
- [12] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *ICRA*. IEEE, 2016, pp. 1476–1483.
- [13] J. Park and H. J. Kim, "Fast trajectory planning for multiple quadrotors using relative safe flight corridor," in *IROS*. IEEE, 2019, pp. 596–603.
- [14] T. Brüdigam, F. D. Luzio, L. Pallottino, D. Wollherr, and M. Leibold, "Grid-based stochastic model predictive control for trajectory planning in uncertain environments," in *ITSC*. IEEE, 2020, pp. 1–8.
- [15] H. Senoussi and A. Saoudi, "A quadtree algorithm for template matching on a pyramid computer," *Theoretical Computer Science*, vol. 136, no. 2, pp. 387–417, 1994.
- [16] Y. Li and Y. Ruichek, "Building variable resolution occupancy grid map from stereoscopic system — a quadtree based approach," in *Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 744–749.
- [17] C. Katrakazas, M. Qaddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.
- [18] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.
- [19] Y. Hattori, E. Ono, and S. Hosoe, "Optimum vehicle trajectory control for obstacle avoidance problem," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 5, pp. 507–512, 2006.
- [20] Z. Zhang, Y. Cai, K. Ceder, A. Enliden, O. Eriksson, S. Kylander, R. Sridhara, and K. Åkesson, "Collision-free trajectory planning of mobile robots by integrating deep reinforcement learning and model predictive control," in *CASE*. IEEE, 2023, pp. 1–7.