3L-Planner: Lightweight LiDAR mapping and real-time local planning for ground robot autonomous navigation

Wenlei Fan[1], Zhenqi Zheng[1], Mingyang Zhou[1], You Li[1]

[1] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China - (wenlifan,zhengzhenqi,zhoumy,liyou)@whu.edu.cn

Keywords: Ground robotics, Autonomous navigation, SLAM, Lightweight mapping, Local planning.

Abstract

Mobile robots are widely used in unmanned surveying, warehouse logistics, and emergency response. However, achieving safe, reliable, and efficient autonomous navigation in unknown environments remains challenging, where accurate environment representation and feasible trajectory planning are crucial. This paper presents an autonomous navigation method integrating lightweight LiDAR mapping with real-time local planning for ground robots. At the perception level, an incremental single-frame point cloud update is used to accumulate and project locally traversable space, producing a lightweight obstacle map that preserves geometric accuracy while reducing planning computation. At the planning level, A* is employed to generate reference control points, and uniform B-spline curves are used to optimize the trajectory while enforcing kinematic feasibility and smoothness. At the control level, nonlinear model predictive control (NMPC) ensures accurate trajectory tracking by producing control commands that satisfy velocity and acceleration constraints. The framework also supports low-cost evaluation in simulation. Experiments in simulated forests, simulated indoor corridors, and real-world gardens and hallways show average navigation speeds of 2.24 m/s, 0.76 m/s, 0.43 m/s, and 0.38 m/s, respectively. Results demonstrate that the proposed method generates smooth, feasible, and safe trajectories and completes autonomous navigation and mapping tasks across diverse environments.

1. Introduction

Autonomous mobile robots, owing to their operational flexibility, have become key enabling technologies in unmanned surveying, intelligent warehousing, emergency response, and autonomous driving. In autonomous navigation tasks, a robot should jointly accomplish three core functions,namely environment perception, real-time planning, and precise control, to determine a sequence of control commands that safely guide the robot from its current state to the target state (Lynch and Park, 2017).

The efficiency and success rate of navigation depend on the time required to compute feasible trajectories, which is proportional to the number of obstacles (spatial factor) and the length of the predicted trajectory (temporal factor) (Xia et al., 2022). As application scenarios expand into dense-obstacle and confined environments, higher demands are placed on the lightweight design, real-time performance, and robustness of navigation systems. Specifically, a robot should maintain accurate environment representation to avoid collisions, control computational cost to satisfy real-time constraints, and generate smooth trajectories that comply with kinematic limits.

Since ground robots typically do not perform active motion in the vertical direction, using a 2D map for environment representation is a common choice (Wang, 2013). Accordingly, some ground robots employ single-line LiDAR sensors for autonomous navigation (Zhang et al., 2020a, Abdalmanan et al., 2023, Sun et al., 2023), aiming to maintain perception accuracy while reducing hardware costs. Related 2D LiDAR SLAM methods (Kohlbrecher et al., 2011, Vincent et al., 2010, Grisetti et al., 2007) offer advantages such as simple structure and low computational overhead, but the lack of vertical perception can lead to incomplete environment representation, thereby compromising navigation safety. In contrast, algorithms such as LOAM (Zhang et al., 2014), FAST-LIO (Xu and Zhang, 2021), FAST-LIO2 (Xu et al., 2022), and LIO-SAM (Shan et al., 2020) generate 3D maps to compensate for this limitation. However, for ground robots, a large portion of the 3D scene lies in unreachable regions, and such redundant data significantly increases storage and computational cost. To reduce redundancy, several studies perform preprocessing on dense ground point clouds. Some methods identify and down-project ground points using elevation information (Lin et al., 2020, Steinke et al., 2023), while others extract foreground objects through point cloud segmentation (Himmelsbach et al., 2010, Steinhauser et al., 2008), thereby improving obstacle representation. These highlight the need to extract effective scene information while maintaining online performance to better support environment representation for ground-robot motion planning.

After perceiving the surrounding environment and estimating its own state, the autonomous mobile robot can plan a trajectory for the upcoming time horizon. The core idea is to compute a feasible trajectory that drives the system from the current state to a desired state, expressed as a sequence of control inputs or intermediate states. However, strict time constraints and limited onboard computational resources make real-time motion planning challenging. According to their technical characteristics, motion planners can be categorized into three classes (González et al., 2015): graph search–based, sampling-based, and optimization-based.

Graph search–based methods typically construct a graph to represent the environment and then perform search within it to obtain a path (Dolgov et al., 2008). Several approaches improve trajectory quality by refining graph structures and search strategies (Alshammrei et al., 2022, Zhu and Sun, 2021). These methods are generally applicable to various kinematic models but struggle to generate high-quality trajectories with limited discrete samples. Sampling-based planners (Kingston et al.,

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-1-2026-245-2026

2018) explore feasible paths by randomly sampling the continuous space and constructing trees or graphs. They can operate in high-dimensional spaces, yet their computational cost becomes substantial when solving for optimal paths in real time.

When kinematic and dynamic constraints are considered, both search-based and sampling-based approaches should explore a much larger state space, whereupon the problem of combinatorial explosion is magnified, significantly reducing planning efficiency. To address this, optimization-based methods take trajectories generated by search or sampling as initialization, integrate higher-order state information, and compute optimal trajectories efficiently in continuous space using numerical solvers. However, traditional optimization methods, including path following (Wang et al., 2022) and spatial cognition techniques (Liu et al., 2022), tend to be overly conservative, potentially causing robots to become trapped in complex environments. Emerging optimization-based planners alleviate this issue by explicitly modeling obstacle avoidance as distance constraints. Among them, Model Predictive Control (MPC) is a widely used optimal control method (Zhang et al., 2020b, Ammour et al., 2022, Han et al., 2023). Its core idea is to predict future system states within a discrete prediction horizon N_p using a dynamic model, while optimizing control inputs over a control horizon N_c, thereby anticipating system behavior (Wang, 2009, Besselmann, 2010). Despite these advantages, MPC-based planners typically compute trajectories only within short horizons to meet real-time requirements, and global optimality cannot be guaranteed. Polynomial trajectory optimization methods (Zhou et al., 2019, Zhou et al., 2020, Kingston et al., 2018) minimize predefined cost functions by adjusting polynomial coefficients and are typically employed for quadrotors without nonholonomic dynamic constraints. Some studies extend such methods to ground robots with differential flatness (Zhang et al., 2023, Zhang et al., 2025). However, ground robots are often subject to nonholonomic constraints, such as Ackermann-steered vehicles being unable to move laterally. Although polynomial optimization produces smooth trajectories, it is difficult to encode such discrete constraints directly. To address this, B-spline-based optimization is employed due to its local control property, which facilitates efficient re-planning (Usenko et al., 2017). Furthermore, its convex hull property simplifies collision avoidance by constraining control points (Zhou et al., 2019). The inherent C^2 continuity of B-splines also ensures smooth, dynamically feasible trajectories for nonholonomic ground robots.

This paper proposes an autonomous navigation method that integrates lightweight LiDAR mapping with real-time local planning for ground robots, aiming to enhance the perception–planning–control pipeline in complex environments. The main contributions are as follows:

1. A lightweight obstacle map construction strategy that accumulates and projects single-frame registered point clouds within the traversable region, preserving environmental fidelity while minimizing storage and computation.
2. A three-layer planning framework that includes A* control point generation, uniform B-spline trajectory optimization, and NMPC tracking, enabling fast generation of smooth trajectories and control inputs under kinematic and safety constraints.
3. A cross-scenario evaluation pipeline that supports simulation testing in Gazebo, providing an efficient and reproducible way to assess planning performance across diverse environments.

2. Method

This section presents our autonomous navigation method, 3L-planner, which integrates lightweight LiDAR mapping with real-time local planning. As shown in Figure 1, the system takes IMU and LiDAR measurements as perception inputs and performs pose estimation and 3D mapping in parallel. Based on these estimates, the planning module generates desired linear and angular velocity commands, which are executed in real time to control the robot.

2.1 Lightweight LiDAR Mapping

To reduce perception blind spots and improve planning efficiency, we obtain single-frame point clouds using FAST-LIO2 (Xu et al., 2022), and incrementally accumulate and project them into a 2D occupancy grid map. This provides a lightweight and efficient representation for trajectory planning. FAST-LIO2 fuses raw LiDAR point clouds and IMU measurements through a tightly coupled iterated Kalman filter (iEKF), enabling state estimation and mapping on resource-limited platforms.

State estimation is formulated by minimizing the residuals between observed points and the map. The system state vector is defined as

$$\mathbf{x}_k = [\mathbf{R}_k,\ \mathbf{p}_k,\ \mathbf{v}_k,\ \mathbf{b}_a,\ \mathbf{b}_g]^T, \tag{1}$$

where $\mathbf{R}_k \in SO(3)$ denotes the rotation matrix, $\mathbf{p}_k \in \mathbb{R}^3$ the position, $\mathbf{v}_k \in \mathbb{R}^3$ the velocity, and $\mathbf{b}_a, \mathbf{b}_g \in \mathbb{R}^3$ the accelerometer and gyroscope biases, respectively. During propagation, IMU integration provides motion prediction, while LiDAR points supply measurement constraints.

For each LiDAR point, the geometric residual is defined as

$$\mathbf{z}_j^k = \mathbf{n}_j^T \left({}^{G}\mathbf{T}_{I_k}\, {}^{I}\mathbf{T}_{L_k}\, {}^{L}\mathbf{p}_j - {}^{G}\mathbf{q}_j \right), \tag{2}$$

where ${}^{L}\mathbf{p}_j$ is the point coordinate in the LiDAR frame, $\mathbf{n}_j$ and ${}^{G}\mathbf{q}_j$ denote the normal vector and a point on the corresponding plane, ${}^{G}\mathbf{T}_{I_k} = \left({}^{G}\mathbf{R}_{I_k},\ {}^{G}\mathbf{p}_{I_k}\right)$ is the LiDAR pose, and ${}^{I}\mathbf{T}_{L_k} = \left({}^{I}\mathbf{R}_{L_k},\ {}^{I}\mathbf{p}_{L_k}\right)$ represents the extrinsic calibration. The iEKF iteratively updates the state by optimizing $\Delta\mathbf{x}$ to minimize $\sum \left(\mathbf{z}_j^k\right)^2$, thereby refining the state estimate and covariance.

To achieve temporally accumulated perception, FAST-LIO2 aligns each LiDAR scan to the global map using an incremental kd-tree. By minimizing the residuals between observed points and the global map, the system obtains both high-frequency pose estimates and motion-compensated 3D point clouds. We denote the set of de-skewed LiDAR points from the k-th scan in the global coordinate frame as

$$\mathcal{P}_k = \left\{ {}^{G}\bar{\mathbf{p}}_i^{\,k} \right\}_{i=1}^{N_k}. \tag{3}$$

Due to range noise, a single-frame point cloud $\mathcal{P}_k$ may miss portions of the surrounding obstacles. In addition, a single scan is relatively sparse and cannot fully describe the environment.

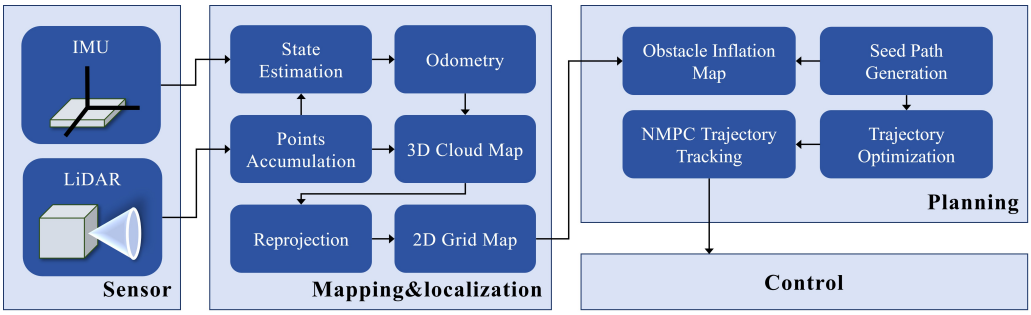

Figure 1. Overall architecture of the proposed 3L-planner, consisting of perception, planning, and control modules.

Therefore, we accumulate consecutive point clouds in a unified global frame to obtain a more complete obstacle observation set:

$$\mathcal{M} = \bigcup_{k=1}^{K} \mathcal{P}_k, \tag{4}$$

where K denotes the number of accumulated scans.

Based on the accumulated obstacle set $\mathcal{M}$, we construct a 3D voxel map that discretizes the continuous space into a voxel set $\mathcal{V} = \{v_i\}$. Each voxel maintains an occupancy probability $p(v_i)$, binarized according to the presence of obstacle points. Free space along LiDAR rays is marked as traversable, while voxels near the ray endpoints are marked as occupied.

Since ground robots operate primarily on a nearly planar surface, the 3D voxel map inevitably contains redundant vertical information irrelevant to navigation. To preserve essential obstacle geometry while reducing dimensionality, we extract an effective height range H that depends on the robot's feasible traversal band. Voxels within this height range are then projected column-wise onto a 2D plane to form a 2D occupancy grid for the local planning module. Let $c_{x,y}$ denote a 2D grid cell whose occupancy state is determined by the set of corresponding voxels

$$\{v_{x,y,z} \mid z \in H\}. \tag{5}$$

Through this projection rule, the 3D environmental data is compressed into a 2D representation in real time. Each 2D grid cell stores an occupancy probability, providing a lightweight yet reliable map for local planning.

2.2 2D Local Trajectory Planning with Lightweight Map

For local planning, the proposed method uses the lightweight 2D occupancy grid as the environmental representation and adapts the 3D Ego-Planner framework (Zhou et al., 2020) into a 2D form to match the motion characteristics of ground robots.

On the constructed 2D grid map, we first generate a collision-free initial path using the A* algorithm (Hart et al., 1968). A* finds a feasible path to the goal by minimizing the sum of the accumulated cost and an optimistic estimate of the remaining cost. For each grid node n, A* employs the heuristic function

$$f(n) = g(n) + h(n), \tag{6}$$

where $g(n)$ denotes the accumulated cost from the start to node n, and $h(n)$ is a heuristic based on the Euclidean distance. By prioritizing nodes with the smallest $f(n)$, A* efficiently searches for a collision-free sequence of 2D control points.

Since A* produces only a discrete path, further processing is required to obtain a temporally and spatially continuous trajectory. We therefore apply a uniform B-spline (De Boor, 1978) to smooth and parameterize the initial path. A 2D B-spline curve is defined by a set of control points and basis functions:

$$\mathbf{p}(u) = \sum_{i=0}^{N} \mathbf{Q}_i \, B_i^k(u), \tag{7}$$

where $\mathbf{Q}_i$ are control points and $B_i^k(\cdot)$ are k-th order B-spline basis functions. Adjacent knot intervals are uniformly spaced. When obstacles form convex regions in the 2D plane, the convex hull property of B-splines ensures that if all control points lie in free space, the resulting curve will also lie entirely in free space, thereby ensuring trajectory safety.

To further enhance trajectory quality, we formulate a multi-objective optimization problem on the initial B-spline curve, jointly considering smoothness, collision cost, dynamic feasibility, and goal-tracking capability:

$$\min J = \lambda_s J_s + \lambda_c J_c + \lambda_f J_f + \lambda_t J_t, \tag{8}$$

where J_s is the smoothness cost, J_c the collision cost, J_f the feasibility penalty, and J_t the goal-tracking cost. The coefficients λ_s, λ_c, λ_f, and λ_t denote the corresponding weights, which are selected through empirical tuning based on task requirements and system performance. As the overall objective J changes with the robot's real-time motion state, the solver should operate efficiently.

We employ L-BFGS (Nocedal, 1980) to solve this optimization problem. As a limited-memory quasi-Newton method, L-BFGS stores only the most recent m gradient vectors to approximate the Hessian, substantially reducing memory usage. Through iterative refinement, L-BFGS guides the trajectory toward a near-optimal solution while satisfying the above constraints. This enables efficient, smooth, and computationally tractable real-time planning on resource-limited onboard platforms, ensuring both safety and feasibility.

2.3 NMPC Trajectory Tracking

Optimization-based trajectory generation inherently satisfies the robot's kinematic model. Building upon this property, we introduce a nonlinear model predictive controller (NMPC) to improve trajectory tracking performance.

We adopt the unicycle kinematic model to describe the planar motion of the robot. The system state vector is defined as

$$\mathbf{x} = [x,\, y,\, \theta]^T \in \mathbb{R}^3,$$

where x and y denote the robot's global position, and θ is the heading angle. The control input is

$$\mathbf{u} = [v,\, \omega]^T \in \mathbb{R}^2,$$

where v is the linear velocity and ω is the angular velocity.

The continuous-time dynamics are given by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix}. \tag{9}$$

Using forward Euler discretization with sampling time T, we obtain

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T\,\mathbf{f}(\mathbf{x}_k,\, \mathbf{u}_k). \tag{10}$$

To handle nonlinear dynamics, we linearize the model at each prediction step using a first-order Taylor expansion around the reference state $\mathbf{x}_r$ and reference input $\mathbf{u}_r$. The Jacobians are

$$\begin{aligned} \mathbf{A}_k &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & -v_r\sin\theta_r \\ 0 & 0 & v_r\cos\theta_r \\ 0 & 0 & 0 \end{bmatrix}, \\ \mathbf{B}_k &= \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned} \tag{11}$$

The discrete-time linearized model becomes

$$\mathbf{x}_{k+1} \approx (\mathbf{I} + T\mathbf{A}_k)\mathbf{x}_k + T\mathbf{B}_k\mathbf{u}_k + \mathbf{O}_k, \tag{12}$$

where the compensation term is $\mathbf{O}_k = -T\mathbf{A}_k\mathbf{x}_r$.

Given the current state $\mathbf{x}_k$ and the reference trajectory $\{\mathbf{x}_r^{(i)},\, \mathbf{u}_r^{(i)}\}_{i=0}^{N-1}$, NMPC solves the following optimization problem at each control step:

$$\min_{\mathbf{U}} J = \sum_{i=1}^{N} \left(\|\tilde{\mathbf{x}}^{(i)}\|_{\mathbf{Q}}^2 + \|\mathbf{u}^{(i)}\|_{\mathbf{R}}^2 \right), \tag{13}$$

where $\mathbf{U} = [\mathbf{u}^{(1)T}, \ldots, \mathbf{u}^{(N)T}]^T \in \mathbb{R}^{2N}$ is the optimization variable, $\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} - \mathbf{x}_r^{(i)}$ is the predicted state error, and $\mathbf{Q} = \omega_0\mathbf{I}_{3N}$, $\mathbf{R} = \omega_1\mathbf{I}_{2N}$ are the weighting matrices.

The control inputs are subject to box constraints:

$$\mathbf{u}_{\min} \le \mathbf{u}^{(i)} \le \mathbf{u}_{\max}, \qquad i = 1, \ldots, N. \tag{14}$$

To solve the NMPC efficiently, we convert it into the standard quadratic programming (QP) form:

$$\min_{\mathbf{U}} \ \frac{1}{2}\mathbf{U}^T\mathbf{H}\mathbf{U} + \mathbf{g}^T\mathbf{U}, \tag{15}$$

where $\mathbf{H} = 2(\mathbf{B}^T\mathbf{Q}\mathbf{B} + \mathbf{R})$ and $\mathbf{g} = 2\mathbf{B}^T\mathbf{Q}\mathbf{E}$.

The QP is solved online using the qpOASES solver (Ferreau et al., 2014), and only the first optimal control input in the sequence is applied to the robot in a receding-horizon manner.

Through time-varying linearization and QP-based optimization, the proposed controller outputs smooth and feasible tracking commands at real-time frequency while strictly satisfying input constraints.

3. Results and Analysis

Experiments are conducted in a real-world environment using the wheeled robotic platform shown in Figure 2. The platform is equipped with a Livox Mid-360 LiDAR and an onboard mini PC powered by an Intel Core i9-12900H processor (14 cores, 20 threads). Our algorithm runs on Ubuntu 20.04 and is developed and integrated under the ROS1 Noetic framework. In the simulation, An Ouster OS1-128 LiDAR sensor is employed for perception in the simulation. LiDAR data is used solely for environmental sensing during planning, while localization relies on ground truth to exclude sensor noise. This setup is specifically designed for the validation and benchmarking of planning algorithms, and it may not fully capture the physical characteristics of sensors in complex real-world environments.

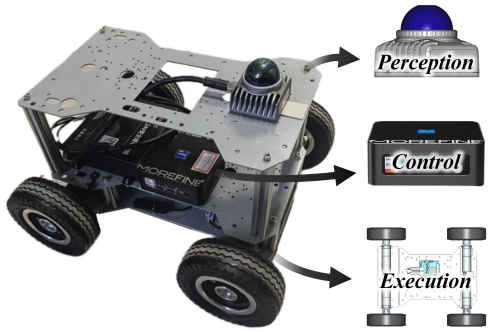

Figure 2. Mobile robot platform used for real-world experiments.

In our method, the local planning range is maintained as a $10\,\text{m} \times 10\,\text{m}$ horizontal area centered on the robot, with a vertical height integration range of $[0.2, 2.0]\,\text{m}$ to encompass the robot's physical dimensions. The spatial resolution of the grid map is set to $0.1\,\text{m}$, and the lightweight map update rate is set to $5\,\text{Hz}$. Local map parameters must align with the robot's physical dimensions, velocity, environmental complexity, and available computing power. To ensure reliable obstacle avoidance, the map resolution should not fall below the sensor's minimum precision. In highly complex environments, increasing spatial resolution while reducing maximum speed can provide finer environmental representation and sufficient planning response time. For safety considerations in real-world experiments, the maximum velocity is limited to $1\,\text{m/s}$. In contrast, the maximum velocity in simulation experiments is increased to $5\,\text{m/s}$ to fully evaluate the algorithm's performance.

This section evaluates the proposed method in four distinct scenarios, including two real-world environments and two simulated environments. Tests 1 and 2 are conducted in real campus scenes, corresponding to an indoor corridor and an outdoor garden, respectively. Tests 3 and 4 take place in simulated environments, corresponding to an indoor corridor and an outdoor forest. The simulated scenarios are constructed based on the CMU exploration environment (Cao et al., 2022).

Test 1 is performed in a real indoor environment (Figure 3). The site is a typical building interior over $45\,\text{m}$ in length and

more than 15 m at its widest part, with several offices connected by long straight corridors. The repetitive wall structure and monotonous textures tend to degrade LiDAR scan matching, increasing the difficulty of localization and mapping. For clearer visualization, the ceiling points in Figure 3 have been removed based on height filtering.

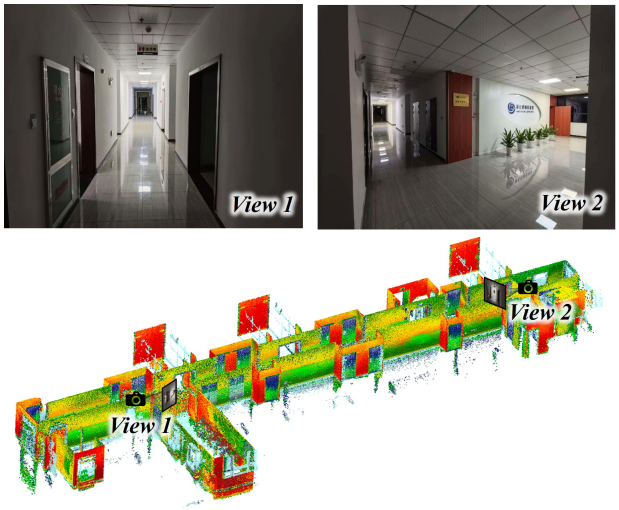

Figure 3. Real indoor corridor environment and corresponding point cloud.

The full trajectory executed by the robot in Test 1 is shown in Figure 4, where the color gradient illustrates the temporal evolution of the motion. The trajectory is continuous, collision-free, and contains no reversals. From the distribution of timestamps along the path, it can be observed that the robot significantly slows down near corners. This behavior results from the combined effects of limited local visibility, reduced obstacle clearance, and the constraints imposed by the NMPC controller, indicating that the system can proactively decrease velocity in response to environmental uncertainty to enhance safety. In contrast, the timestamp distribution becomes sparser along long straight corridors, reflecting that the robot can travel at higher speeds in predictable open segments. Overall, this experiment demonstrates that the proposed navigation method ensures safety and adaptively regulates motion in constrained indoor spaces.

The key process of autonomous navigation and mapping at a corridor corner is shown in Figure 5. As the robot approaches the turn, the lightweight local map captures the geometry of nearby walls and corners in real time (black occupied cells), which is immediately reflected in the planning module as changes in the feasible region. As more of the environment becomes visible, the locally traversable space gradually expands, and the optimized B-spline trajectory consistently remains within safe boundaries.

Test 2 is conducted in a real outdoor garden environment, as shown in Figure 6. This open outdoor space contains irregularly distributed flower beds, shrubs, tables, chairs, and various obstacles with significant height variations. The ground surface exhibits rich texture changes, and illumination conditions are complex. Compared with indoor environments, outdoor scenes feature more diverse obstacle geometries, and LiDAR returns are more easily affected by vegetation structure and occlusions, making local environment representation and obstacle-boundary extraction more challenging.

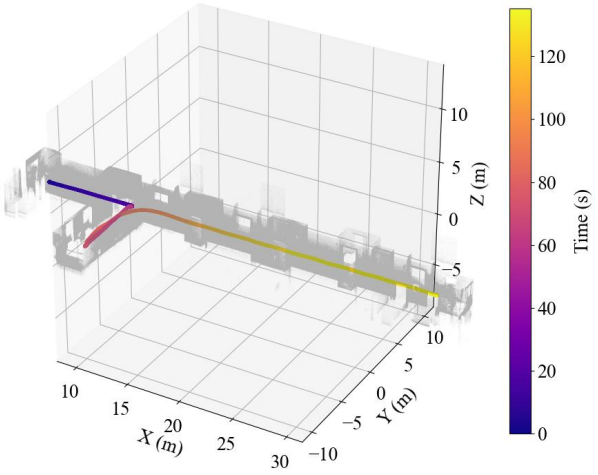

Figure 4. Temporal evolution of the robot trajectory in the real indoor corridor environment.

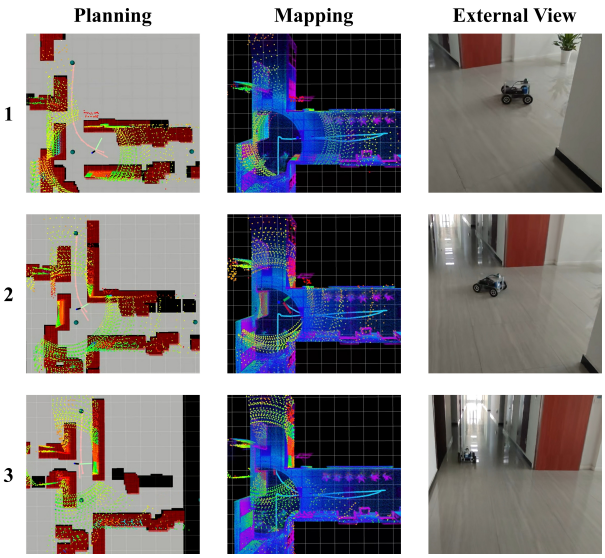

Figure 5. Autonomous navigation and mapping process at a corridor corner.

The complete trajectory executed by the robot in Test 2 is shown in Figure 7, where the color gradient represents temporal evolution. The robot maintains a stable trajectory despite the presence of irregular obstacles such as flower beds, shrubs, and ground undulations. The path remains continuous, without noticeable reversals or stops. Although vegetation often leads to unstable LiDAR returns and non-uniform point density, which make obstacle-boundary extraction more difficult, the final trajectory still adheres smoothly to free-space boundaries. This demonstrates that the proposed method can effectively handle outdoor environments, maintaining reliable perception and planning performance.

The key process of obstacle avoidance in the outdoor garden environment is illustrated in Figure 8. When the robot approaches local obstacles such as the edges of flower beds or low vegetation, the lightweight map promptly updates obstacle boundaries and reflects them in the planning module as a dynamic contrac-

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-1-2026-245-2026 | © Author(s) 2026. CC BY 4.0 License.

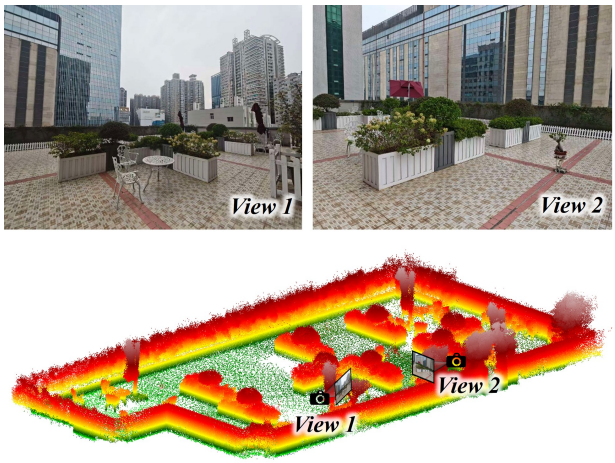

Figure 6. Real outdoor garden environment and corresponding point cloud.

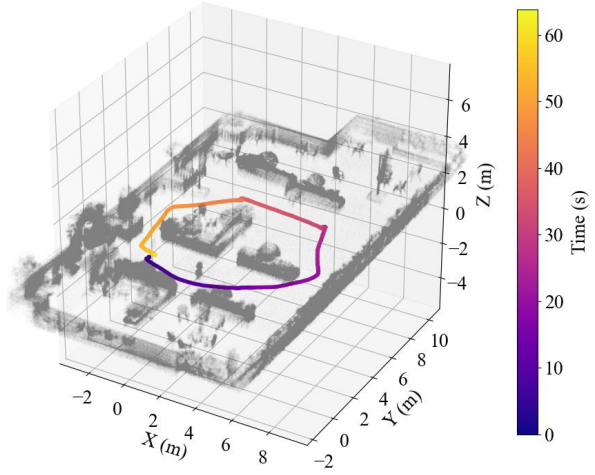

Figure 7. Temporal evolution of the robot trajectory in the real outdoor garden environment.

tion of the traversable region. As additional local point clouds accumulate, the map gradually stabilizes, enabling the planner to generate smooth detour paths while maintaining a safe clearance to nearby obstacles. During this process, the robot first decelerates to perform small-scale path adjustments and then accelerates again once the avoidance maneuver is completed. The curvature variation of the local trajectory shows that the B-spline path exhibits smooth and natural lateral deviations during obstacle avoidance, without sharp turns or discontinuous control commands, demonstrating the proposed planner’s ability to regulate motion under velocity, acceleration, and steering constraints.

Tests 1 and 2 are conducted in real-world environments. We required the robot to plan a return to the vicinity of the starting point, and evaluated its performance using positional consistency at loop closure by measuring the SLAM loop error (the norm of the translational residual). In the garden and corridor scenarios, the loop errors are 0.21 m and 0.29 m, respectively. In both experiments, the robot completed the planning task stably, with no path failures caused by localization drift.

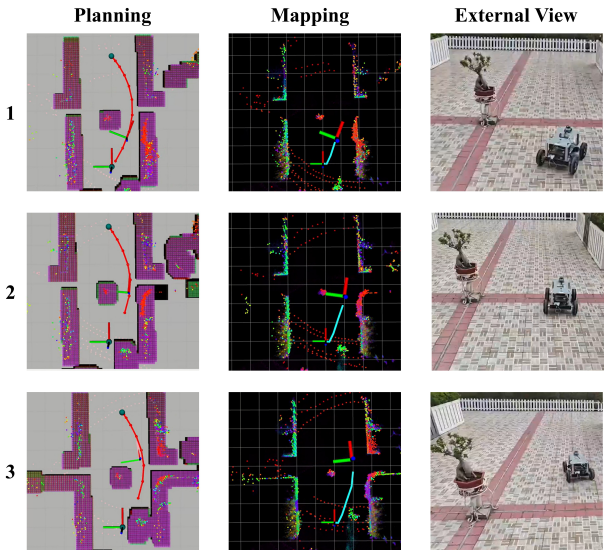

Figure 8. Autonomous navigation and mapping during local obstacle avoidance in the outdoor garden environment.

Test 3 is conducted in a simulated indoor corridor environment, as shown in Figure 9. The scene covers an area of approximately 130 m $\times$ 100 m and consists of multiple long, narrow corridors connected to several hall areas, forming a typical multi-branch indoor passage layout. Obstacles such as tables, chairs, and pillars are placed throughout the environment, and thin structures such as railings are added in certain regions, increasing the complexity of the traversable space.

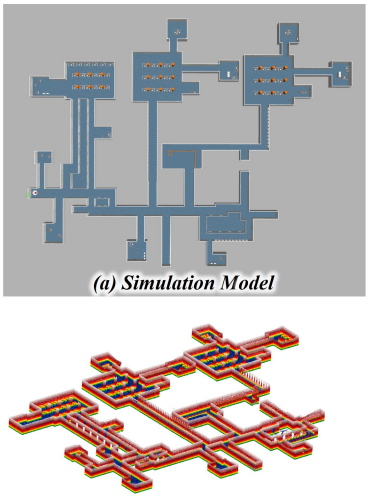

Figure 9. Simulated indoor corridor and its corresponding point cloud.

The complete autonomous navigation trajectory in Test 3 is shown in Figure 10. The robot maintains continuous and well-controlled motion, repeatedly traversing long corridors, entering and exiting halls with pillar obstacles, and completing path switching at intersections. The temporal distribution of the trajectory remains uniform, and no collisions occur throughout the entire process. These results indicate that the proposed method can reliably perform perception, planning, and control in large-scale, multi-corridor, and repetitive indoor environments.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-1-2026-245-2026 | © Author(s) 2026. CC BY 4.0 License.

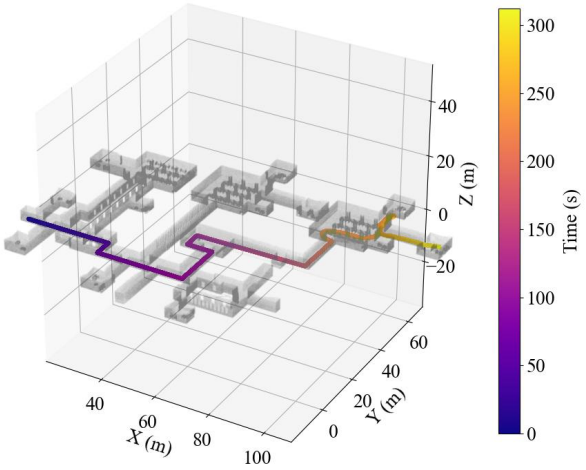

Figure 10. Temporal evolution of the robot trajectory in the simulated indoor corridor environment.

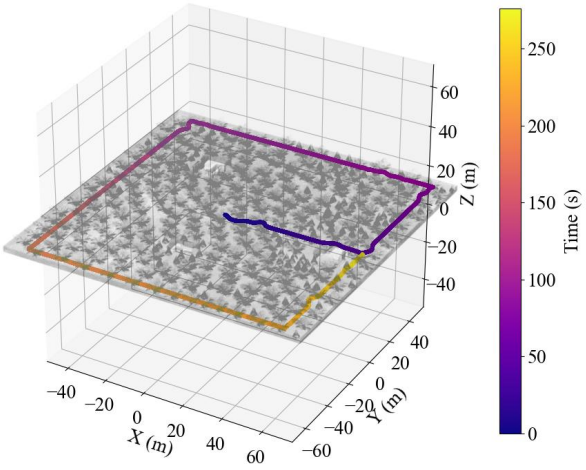

Figure 12. Temporal evolution of the robot trajectory in the simulated outdoor forest environment.

Test 4 is carried out in a simulated outdoor forest environment, as shown in Figure 11. The scene spans 150 m × 150 m and contains a large number of densely distributed trees as well as several scattered houses, resulting in a cluttered and irregular layout.

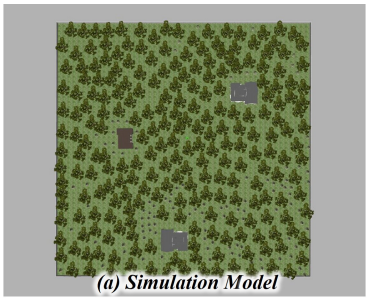

Figure 11. Simulated outdoor forest environment and its corresponding point cloud.

The complete trajectory executed in Test 4 is presented in Figure 12. The robot follows a generally smooth path, and in the central region, where tree density is highest, it performs multiple fine-grained curve adjustments without sharp turns or excessive detours. This demonstrates that the proposed method maintains strong robustness even in environments with dense and irregular obstacle distributions.

Quantitative metrics and trajectory performance across the four scenarios are summarized in Table 1. In real-world experiments, the maximum linear velocity remains below 1 m/s, constrained primarily by the smaller spatial scale and stricter safety requirements. In the garden scene, frequent turning results in a noticeably higher maximum angular velocity compared with the indoor corridor. In contrast, to fully assess the algorithmic capability, a higher velocity limit is configured in the simulation environments. Owing to the larger spatial scale and more flexible maneuvering space, the robot achieves linear velocities exceeding 4 m/s and maintains a maximum angular velocity of 1.5 rad/s, enabling efficient execution of long-range trajectories.

Table 1. Trajectory performance in different scenarios.

Scenario		Area (m^2)	Length (m)	Time (s)	Max.v (m/s)	Max.w (rad/s)
Real-world	Corridor	49.8 × 17.8	51.21	135.40	0.92	0.66
	Garden	12.9 × 25.7	27.43	63.80	0.88	1.22
Simulation	Corridor	130.0 × 100.0	236.61	312.07	4.33	1.50
	Forest	150.0 × 150.0	617.61	276.06	4.40	1.50

Under the same hardware configuration, we evaluated the resource overhead of our method in the indoor scene (Fig 9) and the forest scene (Fig 11), including the average planning time, CPU usage, and memory usage. We compared these results with the 3D baseline without lightweight mapping, as summarized in Table 2. In addition, we further compared the planning time of our method with that of a baseline method based on 2D SLAM and the TEB local planner in the same scenarios, reporting both the average and maximum values, as shown in Table 3.

Simulation results in Forest and Corridor scenarios quantitatively demonstrate the advantages of our lightweight mapping method for real-time planning. As shown in Table 2, by employing 3D-to-2D projection compression, our method reduces average memory usage by nearly half, decreases CPU overhead by approximately 7%, and cuts average planning time by over 75% compared to the 3D baseline. Furthermore, Table 3 compares our approach with a 2D SLAM and TEB local planner baseline. In the Forest scenario, our method achieves an average planning time of 0.29 ms with a maximum of 2.03 ms. In the Corridor scenario, the average time is 0.53 ms, with a worst-case latency of only 6.35 ms, which significantly outperforms the baseline’s 19.32 ms. These quantitative results confirm that our method enhances planning real-time performance and stability by reducing map data scale and collision query overhead.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-1-2026-245-2026 | © Author(s) 2026. CC BY 4.0 License.

Table 2. Average efficiency comparison between the lightweight 2D map and the 3D baseline in simulation.

Scenario	Method	Plan Time (ms)	CPU Usage (%)	Memory Usage (MB)
Forest	Ours	0.29	20.62	245.69
	Baseline	1.63	27.85	414.79
Corridor	Ours	0.53	29.31	246.95
	Baseline	2.19	31.84	417.45

Table 3. Planning time comparison in simulation.

Scenario	Method	Plan Time (ms)	
		Avg	Max
Forest	Ours	0.29	2.03
	Baseline	1.06	2.47
Corridor	Ours	0.53	6.35
	Baseline	1.68	19.32

In summary, the proposed method generates smooth, collision-free, and dynamically adjustable trajectories across environments with varying spatial scales and obstacle distributions. In real-world scenarios, the system adopts a safety-first velocity strategy, while in large-scale simulation settings, it demonstrates efficient long-range trajectory execution. Both the trajectory patterns and quantitative metrics confirm the robustness of the proposed navigation framework across diverse conditions.

4. Conclusion

This paper presents a lightweight LiDAR-based perception and real-time local planning framework for autonomous navigation of ground robots. The proposed method constructs a lightweight local environment representation through incremental point cloud accumulation and projection. By integrating A*-guided B-spline trajectory optimization with NMPC-based tracking, the system achieves smooth, safe, and efficient navigation. Experiments conducted in real indoor corridors, real outdoor gardens, simulated corridors, and simulated forests demonstrate that the method consistently produces stable and collision-free trajectories, while exhibiting strong velocity regulation and environmental adaptability.

While the proposed method achieves lightweight mapping via height-band projection, it is limited by the simplification of 3D environments into 2D occupancy spaces. This dimensionality reduction may lead to insufficient geometric detail for overhanging obstacles, negative obstacles, and partially traversable structures. Furthermore, the current approach primarily targets static scenes and lacks an integrated prediction mechanism for dynamic objects. Consequently, its obstacle avoidance robustness in highly dynamic or geometrically complex environments requires further enhancement.

Nonetheless, compared with data-driven approaches such as reinforcement learning, the proposed framework offers greater interpretability, with clearly defined planning and control logic. To address the aforementioned limitations, future work will focus on incorporating dynamic object tracking and extending the mapping module to more complex environments to further enhance the system’s overall stability and robustness.

References

Abdalmanan, N., Kamarudin, K., Bakar, M. A. A., Rahiman, M. H. F., Zakaria, A., Mamduh, S. M., Kamarudin, L. M., 2023. 2D lidar based reinforcement learning for multi-target path planning in unknown environment. *IEEE Access*, 11, 35541–35555.

Alshammrei, S., Boubaker, S., Kolsi, L., 2022. Improved Dijkstra algorithm for mobile robot path planning and obstacle avoidance. *Comput. Mater. Contin*, 72(3), 5939–5954.

Ammour, M., Orjuela, R., Basset, M., 2022. A MPC combined decision making and trajectory planning for autonomous vehicle collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 23(12), 24805–24817.

Besselmann, T., 2010. *Constrained optimal control: piecewise affine and linear parameter-varying systems*. ETH Zurich.

Cao, C., Zhu, H., Yang, F., Xia, Y., Choset, H., Oh, J., Zhang, J., 2022. Autonomous exploration development environment and the planning algorithms. *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 8921–8928.

De Boor, C., 1978. *A practical guide to splines*. 27, springer New York.

Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J., 2008. Practical search techniques in path planning for autonomous driving. *ann arbor*, 1001(48105), 18–80.

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., Diehl, M., 2014. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

González, D., Pérez, J., Milanés, V., Nashashibi, F., 2015. A review of motion planning techniques for automated vehicles. *IEEE Transactions on intelligent transportation systems*, 17(4), 1135–1145.

Grisetti, G., Stachniss, C., Burgard, W., 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1), 34–46.

Han, R., Wang, S., Wang, S., Zhang, Z., Zhang, Q., Eldar, Y. C., Hao, Q., Pan, J., 2023. RDA: An accelerated collision free motion planner for autonomous navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 8(3), 1715–1722.

Hart, P. E., Nilsson, N. J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Himmelsbach, M., Hundelshausen, F. V., Wuensche, H.-J., 2010. Fast segmentation of 3d point clouds for ground vehicles. *2010 IEEE intelligent vehicles symposium*, IEEE, 560–565.

Kingston, Z., Moll, M., Kavraki, L. E., 2018. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems*, 1(1), 159–185.

Kohlbrecher, S., Von Stryk, O., Meyer, J., Klingauf, U., 2011. A flexible and scalable slam system with full 3d motion estimation. *2011 IEEE international symposium on safety, security, and rescue robotics*, IEEE, 155–160.

Lin, C.-C., Mao, W.-L., Chang, T.-W., Chang, C.-Y. et al., 2020. Fast Obstacle Detection Using 3D-to-2D LiDAR Point Cloud Segmentation for Collision-free Path Planning. *Sensors & Materials*, 32.

Liu, D., Lyu, Z., Zou, Q., Bian, X., Cong, M., Du, Y., 2022. Robotic navigation based on experiences and predictive map inspired by spatial cognition. *IEEE/ASME Transactions on Mechatronics*, 27(6), 4316–4326.

Lynch, K. M., Park, F. C., 2017. *Modern robotics*. Cambridge University Press.

Nocedal, J., 1980. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151), 773–782.

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., Rus, D., 2020. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 5135–5142.

Steinhauser, D., Ruepp, O., Burschka, D., 2008. Motion segmentation and scene classification from 3d lidar data. *2008 IEEE intelligent vehicles symposium*, IEEE, 398–403.

Steinke, N., Goehring, D., Rojas, R., 2023. Groundgrid: Lidar point cloud ground segmentation and terrain estimation. *IEEE Robotics and Automation Letters*, 9(1), 420–426.

Sun, J., Zhao, J., Hu, X., Gao, H., Yu, J., 2023. Autonomous navigation system of indoor mobile robots using 2D lidar. *Mathematics*, 11(6), 1455.

Usenko, V., Von Stumberg, L., Pangercic, A., Cremers, D., 2017. Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 215–222.

Vincent, R., Limketkai, B., Eriksen, M., 2010. Comparison of indoor robot localization techniques in the absence of gps. *Detection and sensing of mines, explosive objects, and obscured targets XV*, 7664, SPIE, 606–610.

Wang, L., 2009. *Model predictive control system design and implementation using MATLAB®*. Springer.

Wang, X., 2013. 2d mapping solutions for low cost mobile robot.

Wang, Z., Zhou, X., Wang, J., 2022. Extremum-seeking-based adaptive model-free control and its application to automated vehicle path tracking. *IEEE/ASME Transactions on Mechatronics*, 27(5), 3874–3884.

Xia, W., Wang, W., Gao, C., 2022. Trajectory optimization with obstacles avoidance via strong duality equivalent and hp-pseudospectral sequential convex programming. *Optimal Control Applications and Methods*, 43(2), 566–587.

Xu, W., Cai, Y., He, D., Lin, J., Zhang, F., 2022. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4), 2053–2073.

Xu, W., Zhang, F., 2021. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2), 3317–3324.

Zhang, J., Singh, S. et al., 2014. Loam: Lidar odometry and mapping in real-time. *Robotics: Science and systems*, 2number 9, Berkeley, CA, 1–9.

Zhang, M., Chen, N., Wang, H., Qiu, J., Han, Z., Ren, Q., Xu, C., Gao, F., Cao, Y., 2025. Universal Trajectory Optimization Framework for Differential Drive Robot Class. *IEEE Transactions on Automation Science and Engineering*, 22, 13030-13045.

Zhang, M., Xu, C., Gao, F., Cao, Y., 2023. Trajectory optimization for 3d shape-changing robots with differential mobile base. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 10104–10110.

Zhang, X., Lai, J., Xu, D., Li, H., Fu, M., 2020a. 2D Lidar-based SLAM and path planning for indoor rescue using mobile robots. *Journal of Advanced Transportation*, 2020(1), 8867937.

Zhang, X., Liniger, A., Borrelli, F., 2020b. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3), 972–983.

Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S., 2019. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4), 3529–3536.

Zhou, X., Wang, Z., Ye, H., Xu, C., Gao, F., 2020. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2), 478–485.

Zhu, D., Sun, J., 2021. A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute. *IEEE access*, 9, 19761–19775.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-XI-1-2026-245-2026 | © Author(s) 2026. CC BY 4.0 License.