ORIGINAL ARTICLE

International Journal of Applied Mathematics in Control Engineering

Journal homepage: http://www.ijamce.com

Multi-Sensor Obstacle Recognition and Fusion Method Based on Costmap

Qiulin Yu[1]* | Meng Tang[1] | Weiliang Wang[1] | Bailan Huang[1] | Xiao Han[1]

[1]Mechanical Engineering School, Southwest Jiaotong University, Chengdu, Sichuan, 610031, China

Correspondence
Qiulin Yu, Mechanical Engineering School, Southwest Jiaotong University, Chengdu, Sichuan, 610031, China
Email: yuqiulin1998@foxmail.com

Article Info
Article history:
Received 2 January 2025
Accepted 11 March 2025
Available online 12 March 2025

Abstract

With advancements in robotics, mobile robots are increasingly deployed in diverse applications. However, complex indoor environments pose challenges for perception and obstacle recognition. This paper proposes a costmap-based multi-sensor fusion method that integrates visual point clouds with two 2D LiDARs for obstacle detection. A downward-facing LiDAR enables ground obstacle detection, with systematic functional design and parameter optimization. Experimental results demonstrate effective 3D obstacle perception and improved obstacle avoidance, achieving an average response time of 1.09 s for static obstacles and 1.30 s for dynamic obstacles.

KEYWORDS

Multi-Sensor, Costmap, Obstacle Recognition, Obstacle Avoidance

1 | INTRODUCTION

With the advancement of technology, mobile robots are being increasingly applied across various fields, including commercial services, transportation and logistics, healthcare, and professional cleaning[1, 2]. In complex indoor environments, environmental perception and obstacle recognition face multiple challenges. Sensors serve as the primary data source for environmental perception. However, single-sensor systems have significant limitations: cameras are highly sensitive to lighting conditions, while LiDAR is constrained by its field of view and susceptibility to occlusions. Moreover, indoor environments contain both static and dynamic obstacles with varying heights, further increasing the complexity of perception.

This paper proposes a multi-sensor-based approach for environmental perception and obstacle recognition using a costmap. Obstacle layers are generated separately from visual point clouds and two 2D LiDARs, and then fused through the costmap. Additionally, a ground obstacle detection method based on a downward-facing LiDAR is designed, with functional implementation and parameter configuration. The proposed method is validated within a unified path planning framework. Experimental evaluations on a real mobile robot platform demonstrate the effectiveness of the approach.

Digital Object Identifiers: 10.62953/IJAMCE.544194

2 | RELATED WORK

2.1 | Obstacle Recognition

Obstacle recognition algorithms have evolved from traditional geometric methods to learning-based methods, with each stage of technological advancement laying the foundation for subsequent improvements.

Early obstacle recognition algorithms primarily relied on geometric models to identify obstacles in the environment by processing point cloud data, such as filtering, segmentation, and clustering. Among them, the grid map-based obstacle recognition method became the most common form. For example, the grid-based method proposed by Thrun et al.[3] and others stores obstacle information in a gridded form and uses simple geometric operations for obstacle detection. This method heavily relies on the source of the obstacles, including their accuracy and coverage.

With the continuous development of computing power and machine learning technologies, obstacle recognition began to incorporate classification and regression algorithms, transitioning into the machine learning-based phase. Support Vector Machine (SVM) proposed by Cortes et al.[4] and the Random Forest (RF) algorithm proposed by Breiman [5] have been widely used for obstacle classification tasks. These algorithms can train on feature data obtained from sensors, enabling more accurate object recognition in complex environments. Compared to traditional geometric methods, machine learning significantly enhances performance in diverse environments, particularly when identifying multiple types of obstacles or handling dynamic obstacles, showing stronger adaptability.

With the rapid development of deep learning technologies, the performance of obstacle recognition algorithms has improved significantly. Specifically, convolutional neural networks (CNNs) proposed by LeCun et al.[6] have made it possible to extract high-level features from complex image and point cloud data, greatly enhancing recognition capabilities. Deep learning-based algorithms can not only perform traditional obstacle detection but also carry out semantic segmentation, enabling precise identification of both the type and location of obstacles. For example, the YOLO model proposed by Redmon et al.[7] and the Mask R-CNN model proposed by He et al.[8] can perform real-time object detection, while the PointNet model proposed by Qi et al.[9] for point cloud data can directly recognize obstacles in three-dimensional space, providing stronger perception capabilities for autonomous navigation in complex environments.

However, single-sensor obstacle recognition often faces issues such as noise interference and misidentification. To improve accuracy and robustness, sensor fusion technology has gradually become an important tool in obstacle recognition. By integrating data from multiple sensors, such as LiDAR, RGB-D cameras, and IMUs, robots can better perceive obstacles in complex environments.[10] In dynamic scenes, sensor fusion can effectively reduce errors and uncertainties caused by the limitations of a single sensor. For example, Cai et al.[11] proposed a low-cost and robust multi-sensor data fusion solution for heterogeneous multi-robot cooperative navigation, which integrates data from multiple sensors. Based on the principles of discrete Kalman filtering and extended Kalman filtering, a three-step joint filtering model is used to improve state estimation, and after processing visual data with the YOLO deep learning object detection algorithm, the fusion filter is updated.

2.2 | Obstacle Avoidance Technology

Obstacle avoidance technology is closely related to path planning algorithms and mainly includes methods based on graph search, sampling, and optimization. It has continuously developed with the improvement of computational capabilities.

Starting with Dijkstra's algorithm [12], this method guarantees global optimality but has lower computational efficiency. The A* algorithm [13] speeds up the search process by incorporating a heuristic function but still faces high computational complexity in large-scale or obstacle-dense environments. The D* Lite algorithm [14] improves real-time performance and adaptability in dynamic environments by updating paths locally.

A representative method is the Rapidly-exploring Random Tree algorithm [15], which rapidly explores the target area through random sampling, though it cannot guarantee the optimal path. The Timed Elastic Band algorithm [16], on the other hand, provides smooth paths through trajectory optimization, making it suitable for dynamic obstacle avoidance scenarios and able to respond quickly to environmental changes.

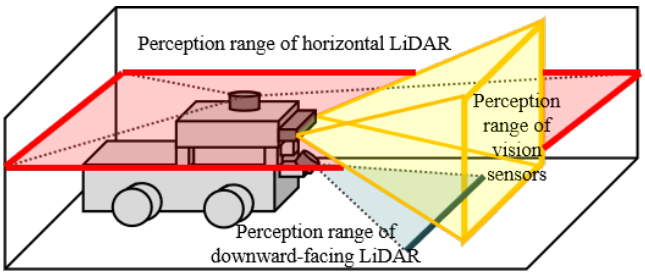


FIGURE 1 Schematic diagram of sensor perception range.

The Dynamic Window Approach [17] and Model Predictive Control [18] are representative of optimization-based algorithms. DWA samples within the velocity space and optimizes the motion trajectory by combining multiple cost functions, enabling real-time obstacle avoidance. MPC predicts future states based on the robot's dynamic model, making it suitable for complex environments with strong adaptability and real-time capabilities.

3 | OBSTACLE DETECTION AND OBSTACLE AVOIDANCE DESIGN

This paper uses an RGB-D camera and two LiDARs for environmental perception. The RGB-D camera is employed to detect and recognize 3D obstacles in front of the robot, while one LiDAR is placed horizontally to detect global obstacles, and the other is tilted downward to detect ground obstacles. The multi-source sensor fusion enables 3D obstacle recognition, describing the distribution of obstacles in the overall environment. The perception range is shown in Fig. 1.

Ground obstacle detection is implemented using the downward-facing LiDAR. The design of the downward-facing LiDAR involves several key factors, including scanning range, obstacle recognition, and geometric correction. To accurately identify ground obstacles and reduce the influence of the ground, the functional scheme of the downward-facing LiDAR needs to be tailored to the specific application scenario.

The horizontally placed LiDAR is primarily used for global obstacle detection. Unlike the downward-facing LiDAR, the horizontal LiDAR application is relatively straightforward and does not require complex custom designs. It can directly input point cloud data into the perception fusion algorithm. Its main task is to continuously scan the surrounding environment and provide global obstacle information.

The three sensors are fused through a costmap. The depth image data from the RGB-D camera, the global obstacle information from the horizontal LiDAR, and the ground obstacle information from the downward-facing LiDAR are integrated through various layers of costmaps to generate a unified grid map. This grid map precisely describes the obstacles in the environment, providing spatial distribution, size, and relative position of obstacles, thus offering high-precision environmental perception data for the robot's navigation and obstacle avoidance.

3.1 | Design of Downward-Facing LiDAR

Traditionally, a single horizontally placed LiDAR has limitations in detecting ground and low-level obstacles. Since its main scanning plane is horizontal, it is challenging to effectively cover low-lying objects or obstacles near the ground. This can lead to blind spots in environments with complex terrain or numerous low-level obstacles. To address this issue, this paper adds an additional tilted LiDAR to the existing horizontal LiDAR, enhancing the detection capability for ground and low-level obstacles by adjusting the scanning angle, thus improving the accuracy and comprehensiveness of environmental perception.

When designing the downward-facing LiDAR, the following key factors were considered: tilt angle α , laser scanning angle β, obstacle recognition, and geometric correction.

The tilt angle α determines the ground obstacle detection distance D. A larger tilt angle brings detected obstacles closer to the robot, reducing the available space for subsequent path planning, which may negatively impact the planning process. Conversely,

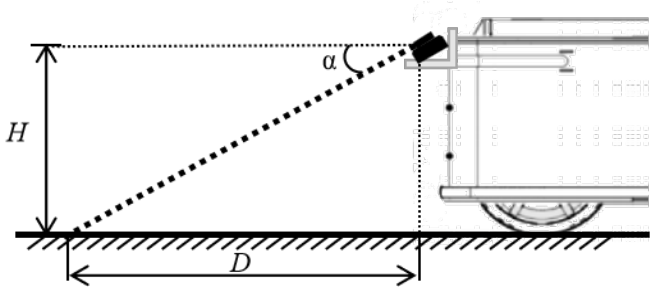


FIGURE 2 Design of tilt angle.

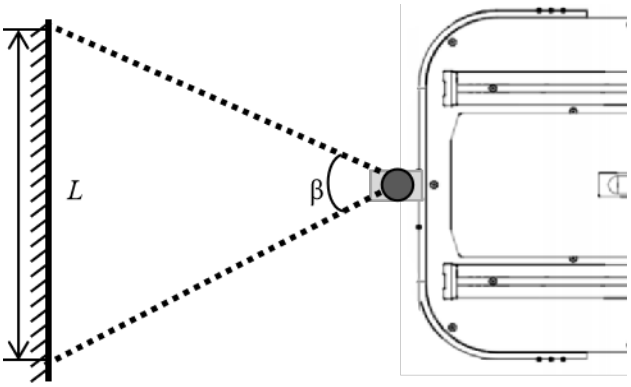


FIGURE 3 Design of scanning angle.

a smaller tilt angle allows obstacles to be detected farther from the robot, providing more space for path planning. However, an excessively small tilt angle may deviate from the intended purpose of ground obstacle detection, making it ineffective in capturing ground-level obstacles.

Therefore, the optimal tilt angle must balance sufficient path planning space and effective ground obstacle detection. The ideal detection distance D should be 1.5 to 2 times the robot's body length. Based on this detection distance and the LiDAR installation height H , the appropriate tilt angle can be calculated as Eq. (1):

$$\alpha = \arctan(H/D) \tag{1}$$

The laser scanning angle β determines the detection range L. When the laser scanning angle increases, the horizontal detection range in front of the LiDAR also expands. Conversely, when the scanning angle decreases, the horizontal detection range becomes narrower.

Theoretically, the laser detection range should be at least equal to the robot's body width. However, in practical applications, to meet the requirements of path planning, the detection range L should cover 1.5 to 2 times the robot's width.

By leveraging geometric relationships, the required laser scanning angle can be derived from the desired detection range, as expressed in Eq. (2):

$$\beta = 2 \cdot \arcsin\left(\frac{L}{2\sqrt{H^2 + D^2}}\right) \tag{2}$$

The primary goal of obstacle recognition is to distinguish between the ground and obstacles. Typically, LiDAR detects obstacles by analyzing laser reflections and calculating obstacle distances based on the reflection time. For the downward-facing LiDAR, it is necessary to compare the measured laser distance with the expected ground distance in the absence of obstacles.

Assuming that, in an obstacle-free scenario, the laser signal at point i reaches the ground at a distance of $l_{i,frist}$, the presence

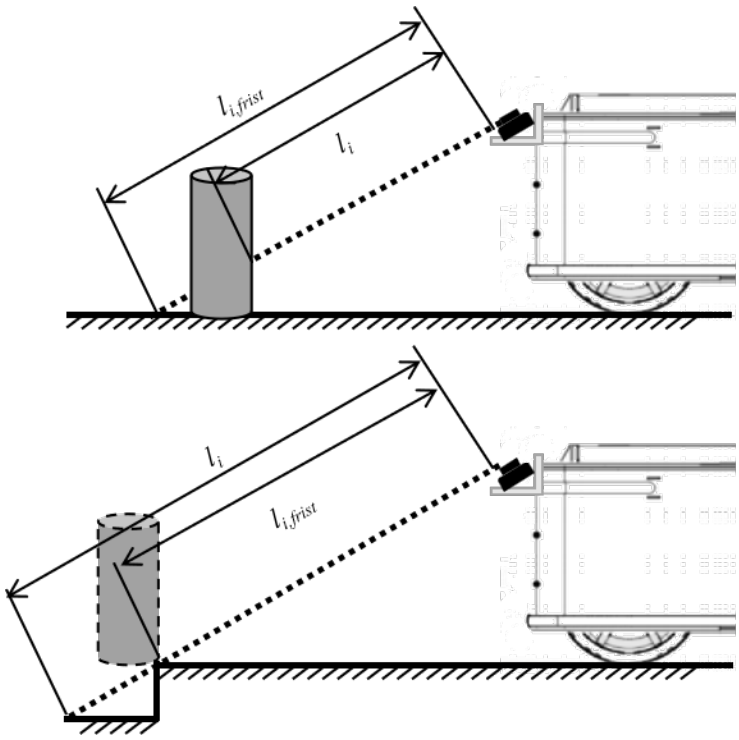


FIGURE 4 Diagram of obstacle recognition.

of an obstacle would result in a detected distance l_i smaller than $l_{i,frist}$. Conversely, if there is a depression or gap, the detected distance would be greater than $l_{i,frist}$, but it should still be classified as a ground obstacle. Therefore, the recorded obstacle distance should be maintained as $l_{i,frist}$.

To account for measurement errors, a threshold thr is introduced to differentiate between obstacles and ground. Following this logic, all point cloud data within $l_{i,frist}$ and its threshold thr are filtered out, while points with distances greater than $l_{i,frist}$ are retained, as described in Eq. (3).

By applying this process to each laser data point within the scanning angle, the ground obstacle recognition for the current position is completed.

$$l_i = \begin{cases} l_i & ,l_i < l_{i,frist} - thr \\ 0 & ,\left|l_{i,frist} - l_i\right| < thr \\ l_{i,frist} & ,l_i > l_{i,first} + thr \end{cases} \tag{3}$$

The purpose of geometric correction is to accurately determine the horizontal distance d_i from an obstacle to the robot. Since the LiDAR is installed at an inclined angle, the coordinate system of the raw data is affected, meaning that the directly obtained obstacle distance is typically the straight-line distance l_i from the obstacle to the LiDAR.

However, for mapping or navigation purposes, the required measurement is the horizontal distance from the obstacle to the robot, which is necessarily shorter than the straight-line distance. To precisely determine the obstacle's position, the LiDAR data must be geometrically corrected based on the tilt angle and installation height, converting the straight-line distance l_i into the horizontal distance d_i, as expressed in Eq. (4):

$$d_i = l_i \cdot \frac{\sqrt{l_{i,frist}^2 - H^2}}{l_{i,frist}} = l_i \cdot \sqrt{1 - \frac{H^2}{l_{i,frist}^2}} \tag{4}$$

Thus, the actual distance of each laser measurement di is given by:

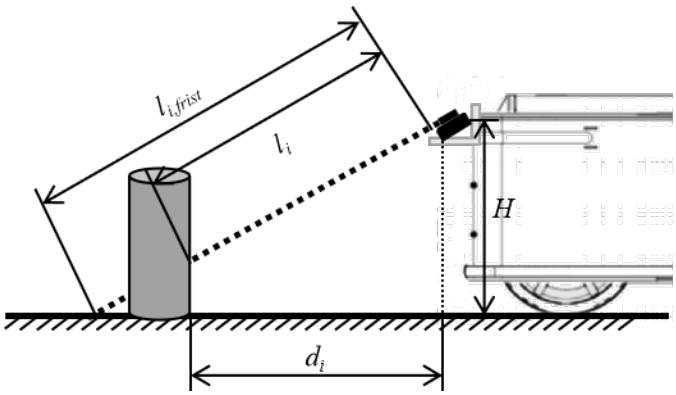


FIGURE 5 Diagram of geometric correction.

$$d_i = \begin{cases} l_i \cdot \sqrt{1 - \frac{H^2}{l_{i,frist}^2}} & , l_i < l_{i,frist} - thr \\ 0 & , \left| l_{i,frist} - l_i \right| < thr \\ \sqrt{l_{i,frist}^2 - H^2} & , l_i > l_{i,frist} + thr \end{cases} \tag{5}$$

3.2 | Perception Fusion

The global laser point cloud data from the horizontal LiDAR, the ground obstacle point cloud data from the downward-facing LiDAR, and the front-facing depth point cloud data from the RGB-D camera are integrated to generate a costmap.

The costmap is a fundamental concept in robotic navigation, playing a crucial role in path planning, obstacle avoidance, and environmental perception. It maps different regions of the environment (such as obstacles and free space) onto a 2D grid and assigns a cost value to each grid cell, representing the "difficulty" or "cost" of traversing that area. During path planning, the robot prioritizes regions with lower cost values to optimize its trajectory.

The cost values represent the distribution of obstacles and free space within the costmap, where each grid cell holds a specific value: Lethal obstacles (254): Represented as static global map obstacles or dynamic obstacles detected by sensors. Obstacle space (128-253): Areas where collisions may occur, especially for robots with complex contours and postures. Irregular robot shapes are more prominent in this category. Inflation space (1-127): Ensures that the robot maintains a minimum safe distance from obstacles during path planning, especially useful in narrow spaces to facilitate the generation of a safe, low-cost path. Free space (0): Areas where the robot can navigate freely.

In obstacle and inflation spaces, cost values can either be assigned directly from predefined thresholds or mapped using a decay coefficient. The cost values are computed based on the robot's geometric dimensions and sensor data, leading to the construction of the costmap.

As shown in Fig. 6, different sensor data sources contribute to the costmap.

Single-line LiDAR scans the environment and generates a 2D point cloud in LaserScan format, originally in polar coordinates. This data is transformed into the global coordinate frame (x,y,z) and stored in PointCloud2 format. Multi-line LiDAR and depth cameras generate depth maps in real-time, which are converted into 3D point clouds, typically in PointCloud or PointCloud2 format. While PointCloud is an earlier ROS format, PointCloud2 is more efficient for handling 3D data.

After converting and storing all sensor data in PointCloud2 format, the points are added to an obstacle list. Each point (x,y,z) undergoes height and distance filtering, removing points exceeding the maximum height or Euclidean distance threshold. The remaining points are projected into a 2D coordinate system (x,y) and marked as obstacles (cost = 254).

Through this process, all sensor-generated point cloud data is continuously pushed into the obstacle list, which is then used to update the obstacle layer. In addition to the obstacle layer, the system also maintains static and inflation layers, forming a layered

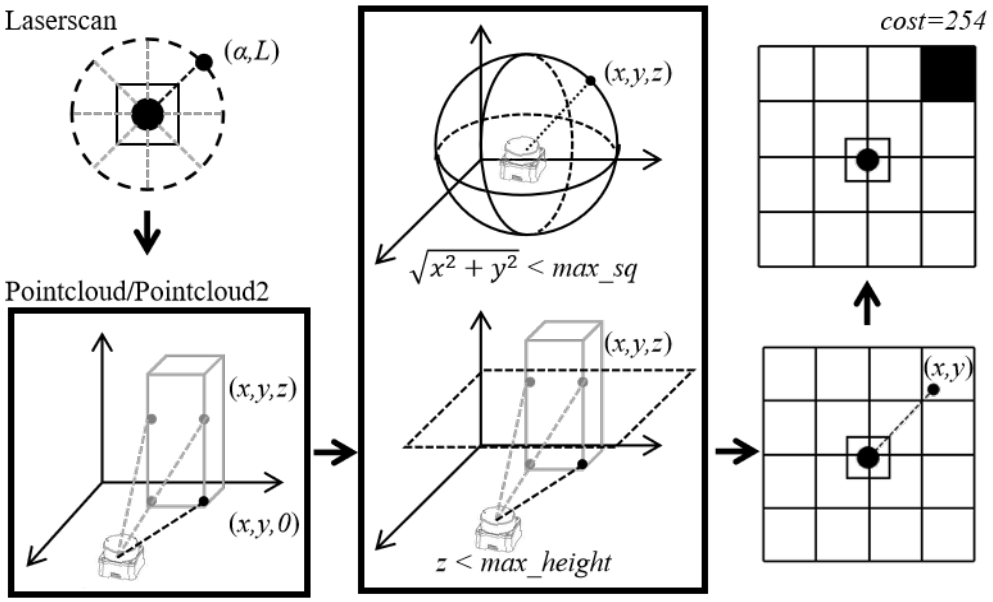


FIGURE 6 Diagram of geometric correction.

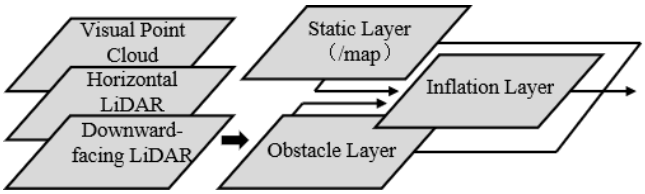


FIGURE 7 Diagram of layer design.

costmap structure, as shown in Fig. 7. The static layer is generated from a global map topic, while the inflation layer is created based on predefined obstacle inflation radii. The final costmap is obtained by overlaying all layers together.

When multiple sensors are used for obstacle detection, two primary update methods are available for the main obstacle layer, as illustrated in Fig. 8.

Overwrite Update: The new obstacle layer directly replaces the old values in the main layer, discarding historical data. This method is suitable for highly dynamic environments but may result in the loss of past obstacle information.

Maximum Value Update: The new obstacle value is compared with the existing value, and the maximum is retained. This approach prevents frequent updates caused by dynamic obstacles and more accurately reflects the actual environment. It is particularly useful in complex or dynamic settings where preserving historical data is beneficial.

The overwrite method is aggressive, best suited for dynamic environments but prone to data loss, whereas the maximum value method is more conservative, retaining past obstacle information unless explicitly cleared. In multi-sensor processing, the maximum value update is the preferred approach.

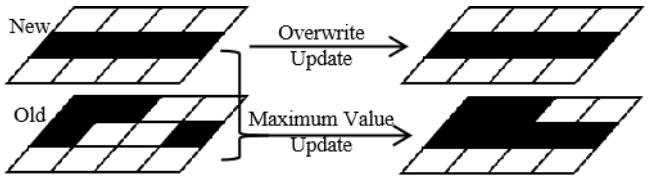


FIGURE 8 Diagram of map update.

3.3 | Obstacle Avoidance Implementation

This paper adopts the A* algorithm for global path planning and the DWA algorithm for local path planning to achieve autonomous navigation and obstacle avoidance.

The A* algorithm is a heuristic search algorithm that introduces a heuristic estimation function into the traditional Dijkstra algorithm, guiding the search direction and significantly improving search efficiency. A* utilizes two cost functions, as shown in Eq. (6):

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

Here, $g(n)$ is the cumulative cost function, representing the actual movement cost from the start node to the current node; while $h(n)$ is the heuristic function, used to estimate the minimum expected cost from the current node to the target node. The heuristic function can use different distance metrics, such as Euclidean distance, Manhattan distance, or Chebyshev distance, with the specific choice depending on the characteristics of the environment.

DWA (Dynamic Window Approach) is a local path planning method based on velocity obstacles. It primarily generates feasible motion trajectories for the robot by considering the robot's speed limitations and the obstacles in the surrounding environment. The DWA algorithm relies on the robot's velocity space (the range of velocities the robot can achieve within a certain time period) and samples different speed combinations to evaluate the trajectories at each speed. The optimal trajectory is then selected as the robot's motion direction.

Velocity space sampling is based on the robot's physical parameters, such as linear velocity, angular velocity, acceleration, and safety distance. Based on these physical parameters, velocity space sampling can be divided into three types of constraints: velocity boundary constraints, acceleration constraints, and environmental obstacle constraints.

After obtaining the velocity sampling space, the DWA algorithm needs to perform sampling at a certain frequency. Assuming the sampling intervals are denoted as v and w for linear velocity and angular velocity respectively, the number of sampled velocity sets can be expressed by the Eq. (7):

$$n = \left[\left(v_{high} - v_{low}\right)/v_{res}\right] \cdot \left[\left(\omega_{high} - \omega_{low}\right)/\omega_{res}\right] \tag{7}$$

where: v_{high} and v_{low} are the maximum and minimum linear velocities; ω_{high} and ω_{low} are the maximum and minimum angular velocities.

Among the n sampled trajectories, an evaluation function is used to assess and select the optimal trajectory. The trajectory evaluation function is given by Eq. (8):

$$G(v_c, \omega_c) = a \cdot \text{heading}(v_c, \omega_c) + b \cdot \text{dist}(v_c, \omega_c) + c \cdot \text{vel}(v_c, \omega_c) \cdot \rightarrow \tag{8}$$

Where: a, b, c are the weighting coefficients of the evaluation function; $\text{heading}(v_c, \omega_c)$ is the heading evaluation function, which describes the geometric relationship between the heading angle and the line connecting the robot center to the target point; $\text{dist}(v_c, \omega_c)$ is the distance evaluation function, which measures the distance between the robot's current velocity trajectory and the nearest obstacle; $\text{vel}(v_c, \omega_c)$ is the velocity evaluation function, representing the magnitude of the robot's current velocity in the trajectory.

4 | EXPERIMENTAL RESULTS AND ANALYSIS

4.1 | Physical Mobile Robot

This paper constructs a two-wheeled differential-drive mobile robot equipped with a stereo depth camera and two LiDAR sensors for multi-sensor environmental perception and obstacle recognition. A general path planning scheme is employed for

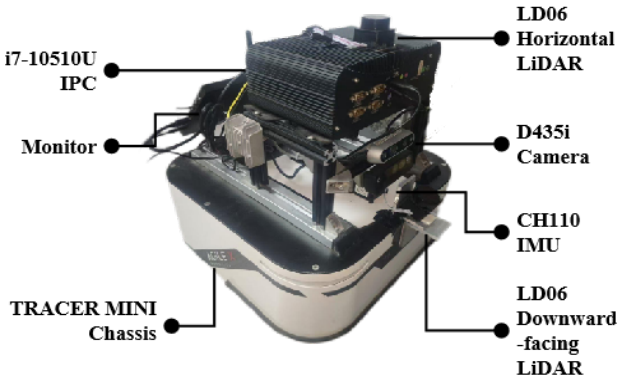


FIGURE 9 Physical mobile robot.

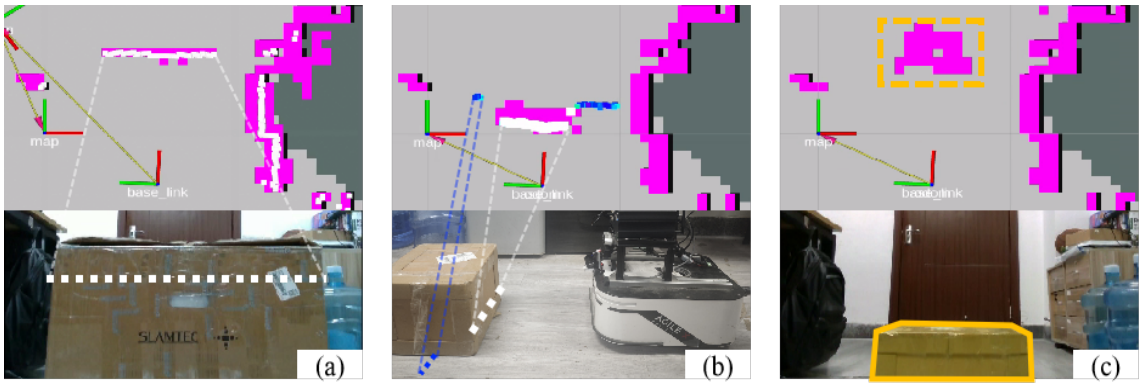


FIGURE 10 Detection results of individual sensors.

navigation and obstacle avoidance. The system configuration is provided in Tab. 1, and the overall setup is shown in Fig. 9.

4.2 | Obstacle Detection

This study employs visual point clouds, horizontal LiDAR, and ground LiDAR to achieve multi-dimensional obstacle detection. In an indoor environment, individual validation is conducted by testing different obstacle heights for each detection method separately. Finally, the functionality of multi-dimensional obstacle detection is verified by simultaneously placing obstacles of varying heights.

Fig. 10 illustrates the obstacle detection results when different sensors are used individually:

- An obstacle taller than the horizontal LiDAR height is placed. The white lines represent the detection results of the horizontal LiDAR, capturing not only the obstacle itself but also environmental information.
- An obstacle shorter than the horizontal LiDAR height is placed. When the robot approaches the obstacle, it successfully detects it and provides the correct position. The white lines represent the corrected obstacle data from the ground LiDAR, while the blue lines represent the raw LiDAR data without obstacle identification or geometric correction.
- An obstacle shorter than the horizontal LiDAR height is placed outside the ground LiDAR detection range. The obstacle is detected using visual point clouds. Due to the nature of depth point clouds, only the side facing the robot has a more complete edge representation.

Fig. 11 presents the obstacle detection results when multiple sensors work together. The red lines represent the horizontal LiDAR data, the white lines represent the ground LiDAR obstacle data, the blue lines represent the raw ground LiDAR data, and

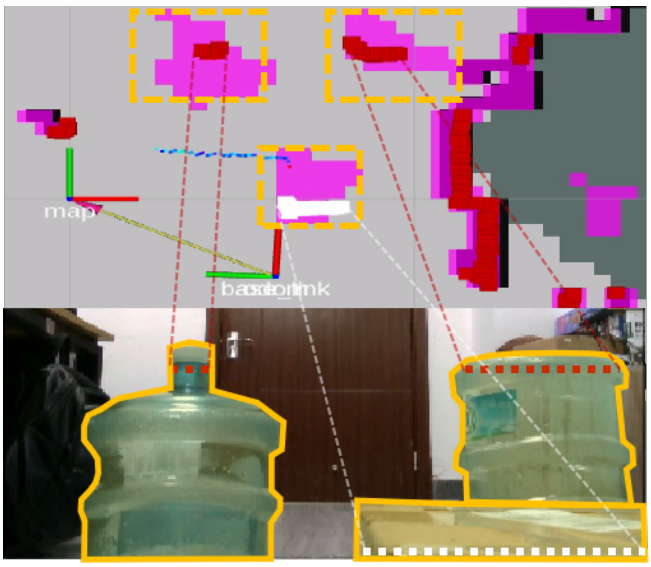


FIGURE 11 Detection results of multiple sensors.

FIGURE 12 Experimental site and the distribution of static obstacles.

the orange lines indicate the visual detection results. By placing two buckets in different positions, the complementary effect of horizontal LiDAR and visual detection can be observed. The visual detection helps to complete the correct obstacle region for the left bucket.

4.3 | Obstacle Avoidance Experiment

After completing obstacle detection, the feasibility of path planning and the timeliness of obstacle avoidance are validated. The experiment consists of static obstacle tests and dynamic obstacle tests. The experimental site and the distribution of static obstacles are shown in Fig. 12, while the dynamic obstacle experiment simulates moving obstacles using pedestrians in the test environment.

The results of the static obstacle experiment are shown in Fig. 14. At 0 s, the robot platform issued a target point and planned a global path directly to the goal. At 3.30 s, the robot detected the first obstacle (a stool), and at 4.45 s, it updated the global path and bypassed the obstacle via the local path. At 7.00 s, the robot detected the second obstacle (a foam box), and at 7.32 s, it updated the global path to avoid it. At 11.15 s, the robot detected the third obstacle (a cardboard box). Due to the influence of the inflation zone, even though the robot was physically able to pass through, it still planned a global path that avoided the obstacle at 12.55 s. Finally, the robot successfully reached the goal at 26.35 s, completing the navigation task.

To verify the stability of the system, four additional experiments were conducted. In five experiments, the robot responded

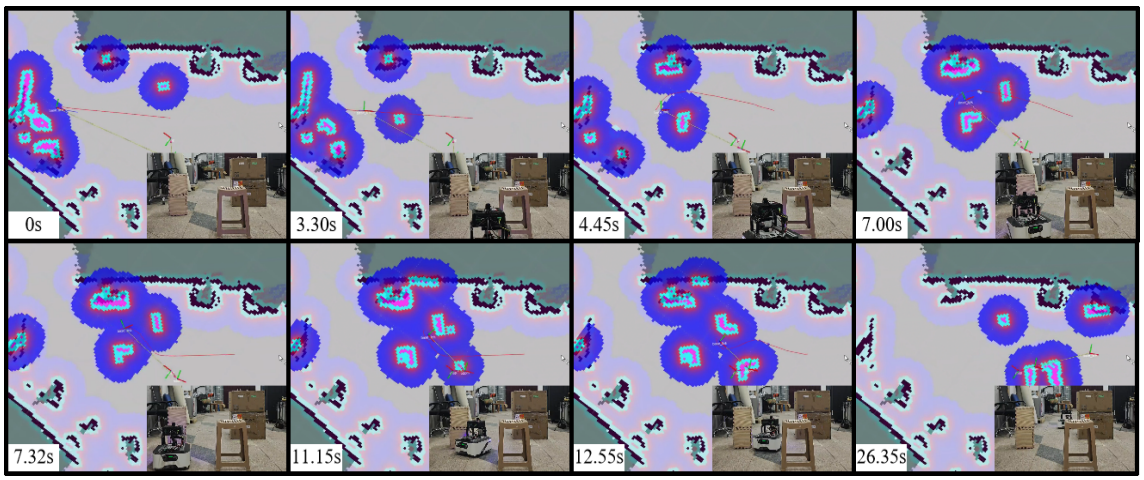


FIGURE 13 Results of the static obstacle experiment.

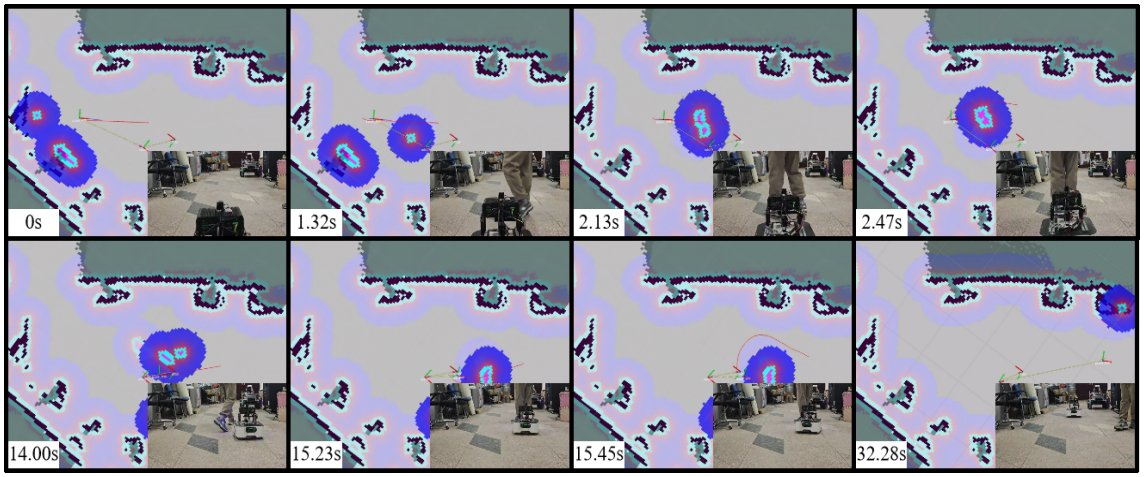


FIGURE 14 Results of the dynamic obstacle experiment.

to 15 static obstacles with an average response time of 1.09 seconds. Timely response indicates that the robot exhibits reliable obstacle avoidance performance in static environments.

The results of the dynamic obstacle experiment are shown in Fig. 15. At 0 s, the robot platform planned a global path directly to the goal. At 1.32 s, a pedestrian began to move and was detected as an obstacle. At 2.13 s, the pedestrian appeared in front of the robot, but the global path was not immediately updated. At 2.47 s, the robot replanned the global path to avoid the obstacle. At 14.00 s, the pedestrian moved again and was detected; at 15.23 s, the pedestrian appeared in front of the robot, but the global path still remained unchanged. At 15.45 s, the robot updated the global path to perform an avoidance maneuver. Ultimately, the robot successfully reached the goal at 32.28 s, completing the navigation task.

In addition, four dynamic obstacle avoidance experiments were conducted. In five experiments, a total of 10 dynamic interruptions were recorded, with an average dynamic obstacle response time of 1.30 seconds.

5 | EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results demonstrate that the proposed costmap-based obstacle detection and avoidance method effectively handles environmental perception and obstacle detection in complex environments. Compared to traditional single-sensor approaches, the complementary use of vision and LiDAR enhances the robustness of environmental perception. For ground

obstacle detection, the designed ground-facing LiDAR successfully identifies and enriches the detection of obstacles in front of the robot.

During real-world experiments, the proposed method accurately identifies various obstacles and generates a costmap. Using a unified path planning framework, both static and dynamic obstacle avoidance were validated. The method achieved an average response time of 1.09 s in static obstacle experiments and 1.30 s in dynamic obstacle experiments, confirming its effectiveness in real-world scenarios.

CONFLICT OF INTEREST

The authors do not have any possible conflicts of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, Q.L.; methodology, Q.L.; validation, W.L., B.L.; resources, X.H.; data curation, X.H.; writing—original draft preparation, Q.L.; writing—review and editing, W.L.; project administration, M.T. All authors have read and agreed to the published version of the manuscript.

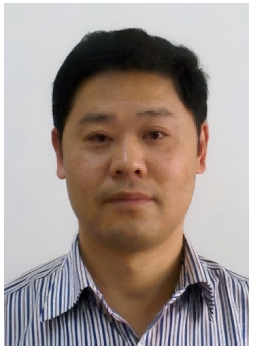
FUNDING

This research received no external funding

REFERENCES

[1] Schepers J, Belanche D, Casaló LV, Flavián C. How smart should a service robot be? Journal of Service Research 2022;25(4):565–582.
[2] Gonzalez-Aguirre JA, Osorio-Oliveros R, Rodríguez-Hernández KL, Lizárraga-Iturralde J, Morales Menendez R, Ramirez-Mendoza RA, et al. Service robots: Trends and technology. Applied Sciences 2021;11(22):10702.
[3] Thrun S. Probabilistic robotics. Communications of the ACM 2002;45(3):52–57.
[4] Cortes C, Vapnik V. Support-vector networks. Machine learning 1995;20:273–297.
[5] Breiman L. Random forests. Machine learning 2001;45:5–32.
[6] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998;86(11):2278–2324.
[7] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 779–788.
[8] He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 2961–2969.
[9] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 652–660.
[10] Wang Z, Wu Y, Niu Q. Multi-sensor fusion in automated driving: A survey. Ieee Access 2019;8:2847–2868.
[11] Cai Z, Liu J, Chi W, Zhang B. A low-cost and robust multi-sensor data fusion scheme for heterogeneous multi-robot cooperative positioning in indoor environments. Remote Sensing 2023;15(23):5584.
[12] Dijkstra EW. A note on two problems in connexion with graphs. In: Edsger Wybe Dijkstra: his life, work, and legacy; 2022.p. 287–290.
[13] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics 1968;4(2):100–107.
[14] Koenig S, Likhachev M. D* lite. In: Eighteenth national conference on Artificial intelligence; 2002. p. 476–483.
[15] LaValle S. Rapidly-exploring random trees: A new tool for path planning. Research Report 9811 1998;.
[16] Rösmann C, Feiten W, Wösch T, Hoffmann F, Bertram T. Trajectory modification considering dynamic constraints of autonomous robots. In: ROBOTIK 2012; 7th German Conference on Robotics VDE; 2012. p. 1–6.
[17] Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine 1997;4(1):23–33.
[18] Rawlings J, Mayne D. Postface to model predictive control: Theory and design. Nob Hill Pub 2012;5:155–158.

Qiulin Yu is currently pursuing his Master's degree study at the School of Mechanical Engineering, Southwest Jiaotong University, Chengdu, China. He obtained his BS degree from Southwest Jiaotong University, China, in 2022. His main research interests are in the areas of SLAM and autonomous navigation technology.



Meng Tang , Ph.D., received his M.E. from Southwest Jiaotong University in 1998 and his Ph.D. from the Norwegian University of Science and Technology(NTNU) in 2005. He is currently with the School of Mechanical Engineering, Southwest Jiaotong University. His research interests include mechatronic-hydraulic control, system modeling, electric vehicle drive control, and intelligent systems.

Weiliang Wang , received his bachelor's degree from Southwest Jiaotong University in 2023. He is currently pursuing a master's degree at the School of Mechanical Engineering, Southwest Jiaotong University. His current research interests include ROS, robotics, and reinforcement learning.

Bailan Huang , received his bachelor's degree from Southwest Jiaotong University in 2023. He is currently pursuing a master's degree at the School of Mechanical Engineering, Southwest Jiaotong University. His current research interests include ROS, robotics, and LiDAR point cloud processing.

Xiao Han , received her bachelor's degree from Southwest Jiaotong University in 2022. She is currently pursuing a master's degree at the School of Mechanical Engineering at Southwest Jiaotong University. Her current research interests include deep learning, gesture recognition, and semantic segmentation.