

Real-Time Mapping and Planning Intelligent Paths using Optical LiDAR and Quadruped Robot

Wen-Yang Chang^{1,2*}, Chen-Hsiang Hung¹, Zheng-Xun Huang³, Hsuan-Jui Chang⁴

¹Department of Mechanical and Computer-Aided Engineering, National Formosa University, Yunlin, Taiwan.

²Smart Machine and Intelligent Manufacturing Research Center, National Formosa University, Yunlin, Taiwan.

³Doctoral Degree Program in Smart Industry Technology Research and Development, National Formosa University, Yunlin, Taiwan.

⁴Department of Bioscience and Biotechnology, National Taiwan Ocean University, Keelung, Taiwan

*E-mail: wenyang@nfu.edu.tw

Keywords: Real-time Mapping, Path Planning, Field Detection, Quadruped Robot, Obstacle Avoidance

Abstract

In general, the obstacle detection systems mainly rely on depth cameras or AI-based vision approaches; however, these methods are often constrained by limited fields of view and the need for continuous model retraining to adapt to complex and dynamic industrial scenes. To overcome these limitations, this study proposes a LiDAR-based obstacle detection and field monitoring system integrated with a quadruped robot. The proposed system focuses on three main components: real-time field mapping, intelligent path planning with obstacle avoidance, and field change detection. LiDAR point cloud data are pre-processed using pass-through and voxel grid filters, followed by coordinate transformation into the robot reference frame. The Cartographer simultaneous localization and mapping (SLAM) algorithm are employed to generate high-resolution occupancy grid maps for navigation. For autonomous operation, erosion processing and connected component labelling are used to define safe regions, while the A* algorithm enables efficient path planning and adaptive obstacle avoidance in complex environments. To detect unknown obstacles and environmental changes, Gaussian filtering and map differencing are applied, and map similarity is evaluated using histogram analysis and SIFT-based feature matching. Experimental results demonstrated that the system achieves a mapping resolution of 0.05 m and satisfies the Taiwan Association of Information and Communication Standards (TAICS) requirements, including 0.2 m planimetric accuracy and less than 0.1 m positional error. The proposed approach effectively identifies unknown obstacles and visually highlights risk areas, providing a reliable solution for intelligent workplace safety monitoring.

1. Introduction

With the rapid advancement of intelligent robotics and artificial intelligence, the quadruped robots have gained increasing attention due to their superior mobility and adaptability in complex environments. By adjusting gait parameters such as walking height and gait cycle, quadruped robots can maintain stable locomotion over diverse terrains, making them suitable platforms for autonomous field mapping and environment inspection. Compared with wheeled robots or automated guided vehicles (AGVs), the quadruped robots are less constrained by terrain and wheel-ground contact limitations, motivating extensive research on quadruped locomotion and obstacle avoidance.

Joseph Norby et al. proposed the Quad-SDK, an open-source software framework based on the Robot Operating System (ROS), designed to enhance the autonomous locomotion and obstacle avoidance capabilities of quadruped robots. The framework integrates global motion planning with nonlinear model predictive control (NMPC). The Unitree A1 quadruped robot was used as the experimental platform, incorporating multiple sensing devices, including an inertial measurement unit (IMU), rotary encoders, servo motor drivers, and foot-end force sensors. These sensors enable the construction of an accurate motion model to predict future state trajectories for stable and adaptive locomotion.

Siddhant Gangapurwala et al. proposed a real-time corrective reinforcement learning (RL) approach for the quadruped robot ANYmal, enabling online trajectory adjustment when encountering obstacles. The study integrated multiple sensors on the ANYmal platform, including LiDAR, depth cameras, gyroscopes, rotary encoders, and servo motor drivers. Data collected from these sensors facilitate accurate estimation of the robot's posture and joint states during trajectory optimization, providing essential reference information for the trajectory solver within the reinforcement learning framework.

This study proposes a solution that combines LiDAR technology with the agility of quadruped robots for real-time

field mapping and path planning capabilities. The system can adapt to dynamic conditions, following the mapping process, effectively identifies unmarked obstacles and visually displays them on a user interface, thereby facilitating the identification and mitigation of safety risks within workspaces. In addition, the Cartographer algorithm is employed to transform the acquired point cloud data into an accurate occupancy grid map, which serves as environmental input for subsequent intelligent inspection and obstacle avoidance modules. Based on the constructed occupancy grid map, the intelligent inspection path planning and obstacle avoidance framework integrates erosion processing, connected component labelling, and the A* search algorithm to develop an effective navigation and collision avoidance mechanism. This design ensures that the quadruped robot can move safely and efficiently within the environment while dynamically adapting to environmental changes and obstacles.

2. Methodology

2.1 Experimental setup

In this study, a quadruped robot is employed as the mobile platform for field mapping, autonomous inspection, and obstacle avoidance tasks. The robot, Mini Pupper, has dimensions of 210 mm × 110 mm × 165 mm (length × width × maximum ground clearance) and a mass of 560 g without payload. To support wireless operation, it is powered by a 1000 mAh micro-USB battery. The internal hardware configuration includes a Raspberry Pi controller, 240 × 320 pixels of ISP LCD display, and twelve servo motors. Each leg comprises three servo motors and three linkage mechanisms, which actuate the hip and knee joints to enable stable quadruped locomotion. Figure 1 is the schematic diagram for environment inspection and obstacle avoidance.

For environmental sensing, a Light Detection and Ranging (LiDAR) sensor is mounted on the robot to provide accurate distance measurements within the workspace. This sensor enables the quadruped to construct occupancy maps and perform

autonomous navigation and obstacle avoidance. The LiDAR module measures $38.59 \text{ mm} \times 38.59 \text{ mm} \times 33.5 \text{ mm}$ (length $\times$ width $\times$ height) and operates based on direct Time of Flight (dToF) ranging technology using an infrared laser with a wavelength of 905 nm. It provides a measurement range from 0.02 m to 12 m with a distance resolution of 0.015 m, offering precise spatial information for intelligent path planning and real-time navigation.

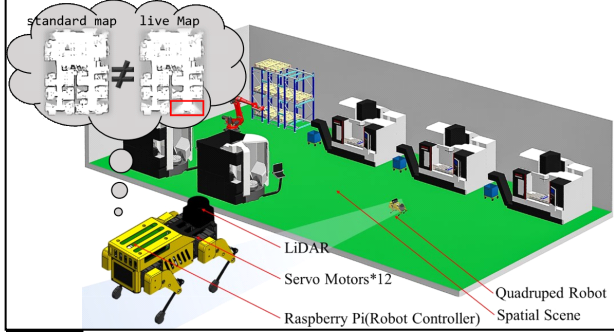


Figure 1. Schematic diagram illustrating the system architecture of the quadruped robot and its field mapping process for environment inspection and obstacle avoidance.

2.2 Robot and LiDAR coordinate system transformation

In general, the LiDAR scanning for point cloud data, inherently in polar coordinates within the radar's own system, require precise transformation to the robot's coordinate system for effective operational integration is demonstrated in Figure 2. This critical transformation, executed using a conversion matrix, is essential for the robot's accurate data interpretation and subsequent processing. A precise coordinate transformation based on their relative spatial orientation is required due to positional and angular offsets between the optical LiDAR and the robot coordinate system. The obstacle positions detected by the LiDAR, which are initially represented in polar coordinates (r, θ) , are systematically converted into Cartesian coordinates. This comprehensive process involves compensating for the LiDAR's rotation relative to the robot coordinate system and applying the translation matrix to accurately transform point coordinates into the robot reference frame. The coordinate transformation between the optical LiDAR and the robot coordinate system is shown in Eq. (1), where p_{LC} is the Cartesian coordinate and T_d is the translation matrix

$$p_R = p_{LC} + T_d = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & T_x \\ \sin\alpha & \cos\alpha & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{LP} * \cos(\theta_{LP}) \\ r_{LP} * \sin(\theta_{LP}) \\ 0 \\ 1 \end{bmatrix} \quad (1)$$

$$\text{where } x_{LC} = r_{LP} * \cos(\theta_{LP}), y_{LC} = r_{LP} * \sin(\theta_{LP})$$

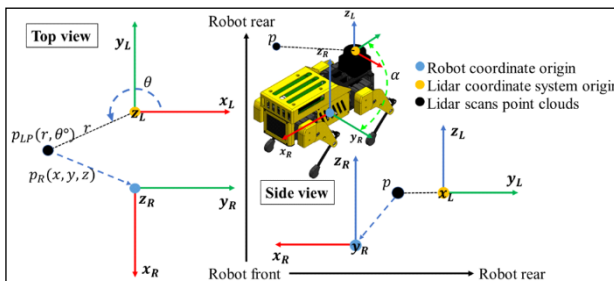


Figure 2. Geometry relationship diagram of robot and LiDAR coordinate systems

2.3 Cartographer SLAM algorithm

Cartographer SLAM algorithm, introduced by W. Hess et al., employed the optical LiDAR to capture environmental point

clouds for real-time robot localization and mapping. It consists of sensor data input with pass-through and voxel grid filtering, local map modelling through scan matching for frame alignment, and global map construction by merging processed submaps. There have five steps for real-time mapping and planning intelligent paths as follows :

(1) Submap creation : The first submap is created and initiated by the robot's first scan point cloud as shown in Figure 3(a). The multiple occupancy grids are determined by voxel size with initial gray areas, indicating unexplored spaces, as Eq. 2. Generally, the submap's origin aligns with the robot's position during initial data capture. The Lidar scanning points are converted to grid indices for storage, representing the captured environment at creation. The state of the submap after incorporating the scan data is depicted in Fig. 3(b).

$$\text{occupancy Grid Map} = \{m_i\} \text{ with index ID} = [i, j] \quad (2)$$

$$\text{where } i = \text{floor}\left(\frac{p_{ix}}{\text{voxel}_{size}}\right) \text{ \& } j = \text{floor}\left(\frac{p_{iy}}{\text{voxel}_{size}}\right)$$

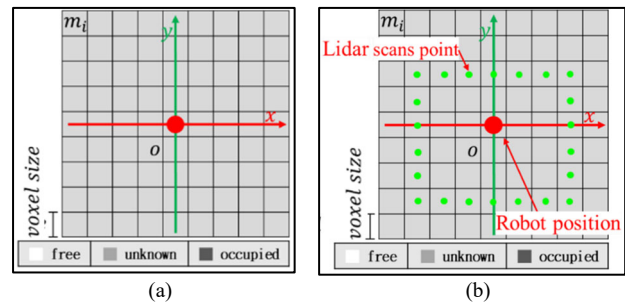


Figure 3. Submap data structure (a) Initial state (b) State after receiving Scan point cloud data

(2) Scan Match : During quadruped robot field exploration, robot motion causes shifts in the acquired point clouds, making scan frame alignment essential for constructing a continuous and accurate map. As illustrated in Figure 4, the scan matching enables the estimation of the robot's position and orientation, providing precise localization within the environment. The matching process begins with submap initialization using the first scan data, which are converted into grid indices, to ensure accurate map overlay. Using transformation matrices and least-squares optimization, as described in Eqs. (3) and (4), the scan data are adjusted within predefined translation and rotation constraints to achieve optimal alignment with the submap. The resulting optimal alignment ξ , given in Eq. (5), is then used to update the robot's position and orientation within the submap, thereby maintaining mapping accuracy and mitigating motion-induced displacement.

$$T_{\xi} p = \begin{pmatrix} \cos\xi_{\theta} & -\sin\xi_{\theta} \\ \sin\xi_{\theta} & \cos\xi_{\theta} \end{pmatrix} p + \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \quad (3)$$

$$\argmin_{\xi} \sum_{i=1}^l \left(1 - M_{\text{smooth}}(T_{\xi}, p_i)\right)^2 \quad (4)$$

$$\xi = (\xi_x, \xi_y, \xi_{\theta}) \quad (5)$$

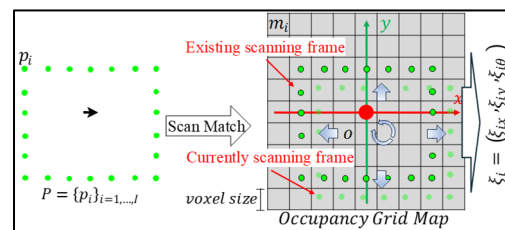


Figure 4. Schematic diagram of the scan match process for occupancy grid map of scanning matching.

(3) Submap Modeling : Radar scan point clouds return measurements only when the beam intersects an obstacle surface and do not directly indicate free space. To determine the state (occupied, free, or unknown) of each occupancy grid cell m_i within a submap, the Cartographer algorithm employs Bresenham's line algorithm. This approach iteratively computes the grid cells traversed by the sensor beam along the positive x-axis. The process begins by determining the relative start and end points of the LiDAR beam, excluding purely vertical or horizontal paths. The differences in the x- and y-coordinates (Δx and Δy) are computed as defined in Eq. (6). If an obstacle is located to the left of the LiDAR, i.e., its x-coordinate is smaller than that of the sensor, the positions of the LiDAR and the scan point are swapped to ensure that the beam propagates along the positive x-axis.

$$sub\ p_{begin} = (x_0, y_0) \ \& \ sub\ p_{end} = (x_1, y_1) \quad (6)$$

As illustrated in Figure 5, the scanning beam path simulates the grid traversal process for a single LiDAR scan point. The red dot represents the LiDAR position p_{begin} , the green dot denotes the obstacle position p_{end} , and the red dashed line indicates the beam path. The variable y represents the beam's height on the y-axis at each grid position. As grid cells are updated sequentially, the parameter m denotes the incremental change in the beam height along the y-axis for each unit advancement along the x-axis.

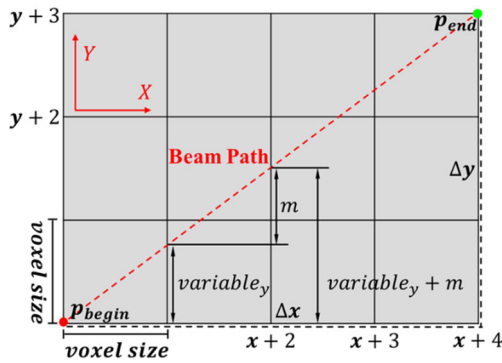


Figure 5. Schematic diagram of beam path calculation for optical LiDAR scan points

(4) Occupancy probability update in Submaps : As the optical LiDAR continuously captures point cloud data of environmental features, potential accuracy issues and inherent noise in LiDAR scans must be considered, particularly when the robot operates in dynamic environments. These factors may introduce errors into the scan data. To address this issue, the algorithm quantifies the confidence of each grid cell's occupancy state by accumulating results from multiple scans, thereby improving map stability through repeated probability updates.

The probabilistic representation is shown in Eqs. (7) & (8), where $Odds(s = hit)$ denotes the occurrence ratio of the occupied state of a grid cell within the submap. $P(s = hit)$ represents the probability that a grid cell is occupied, which can be interpreted as the likelihood of scan points falling within that cell, while $P(s = free)$ denotes the probability that the grid cell is classified as free space based on Bresenham's line algorithm. Both probabilities are constrained between 0.1 and 0.9, and their sum equals 1. During the subsequent probability update process within the submap, each time a new scan frame and its corresponding pose are inserted, a series of recalculations and updates of the occupancy grid probabilities is triggered. This iterative update mechanism enables the map to progressively converge toward a consistent and reliable representation of the environment.

$$Odds(s = hit) = \frac{P(s=hit)}{1-P(s=hit)} \quad (7)$$

$$Odds(s = free) = \frac{1-P(s=hit)}{P(s=hit)} \quad (8)$$

(5) Global Map Construction : The final submaps are generated by iteratively applying the aforementioned mapping process and are completed once the maximum number of scan frames is reached. Subsequently, a new submap is initialized, and this cycle repeats until the robot completes the field exploration. The completed submaps are then aligned through the global map construction module (Global SLAM) to form a consistent global map, as shown in Figure 6. Each occupied grid cell is converted into a single pixel, and the corresponding occupancy odds ratio (from 0 to 100) is mapped to grayscale values from 0 to 255.

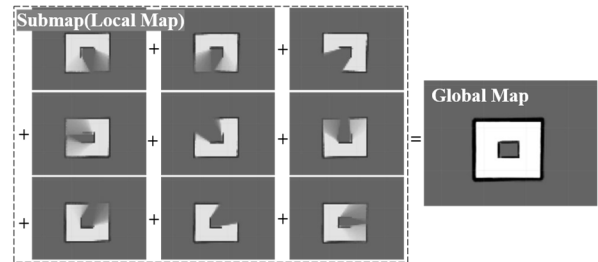


Figure 6. The construction process of the global map and visualization in grayscale

2.4 A* search algorithm

The path planning process was used A* search algorithm, involves storing the considered pixel node positions, total path distance values, and their parent nodes in an open set according to the exploration order. Each candidate pixel is then evaluated to determine whether it lies outside the boundary, corresponds to an obstacle, or is already in the closed set. Pixels meeting any of these conditions are excluded from further processing. The algorithm subsequently searches the open set in reverse order to identify the pixel with the smallest total path distance, thereby selecting nodes that are likely to lie on the shortest path. When multiple pixels have equal total path distances, priority is given to the pixel with the smaller predicted distance $h_c(n)$, indicating a closer proximity to the destination and refining the selection of potential shortest-path nodes.

To efficiently evaluate the robot's path and avoid excessive searching, this study employs the total path distance $f_d(n)$ to assess the robot's traversal. The total path distance is composed of two components: the distance already traveled $a_d(n)$ and the remaining path distance $h_d(n)$, as shown in Eq. (9):

$$f_d(n) = a_d(n) + h_d(n) \quad (9)$$

Where

$$a_d \text{ or } h_d = \sqrt{(Min(\Delta(x, y)) \cdot vSz)^2 + (Min(\Delta(x, y)) \cdot vSz)^2 + (|\Delta x| - |\Delta y|) \cdot vSz}$$

$$\Delta(x, y) = (x_{ref} - x_{start}, y_{ref} - y_{start})$$

$$\Delta(x, y) = (x_{ref} - x_{goal}, y_{ref} - y_{goal})$$

Here, the traveled distance $a_d(n)$ reflects the exact distance from the starting point to the current node, while the remaining distance $h_d(n)$ estimates the shortest distance from the current node to the destination. As illustrated in Figure 7, red pixels represent the starting point, green pixels represent the destination, and orange pixels denote the current node under evaluation. The

total path distance at each node is obtained by summing its traveled distance and remaining distance. The traveled distance is calculated as the x- and y-axis deviations between the current node and the starting point, whereas the remaining distance is calculated the x- and y-axis deviations between the current node and the destination.

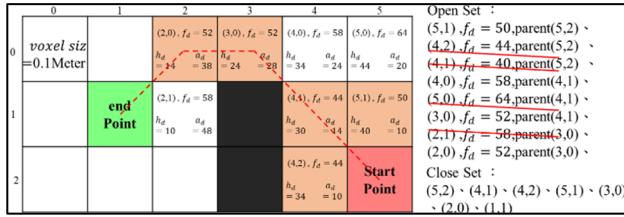


Figure 7. Iterative pathfinding and planning with the A* search algorithm

2.6 Scale-Invariant Feature Transform

This study employs the Scale-Invariant Feature Transform (SIFT) [23] algorithm for obstacle detection by comparing standard and real-time maps. The algorithm identifies stable feature points in both maps and performs feature matching to detect obstacles. The SIFT algorithm consists of four main stages: scale-space extremum detection, feature point localization, feature orientation assignment, and feature descriptor construction. During the scale-space extremum detection stage, Gaussian blur [24] and the Difference of Gaussian (DoG) are applied to process images at multiple scales. Equation (10) defines the Gaussian blur operation, where L denotes the blurred image, G represents the Gaussian convolution kernel, I is the original image, σ indicates the scale, and $*$ denotes the convolution operation.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (9)$$

$$\text{where } G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

To construct the scale space, the algorithm applies a series of Gaussian blurs to the original image, forming image sets at different scales (σ) to extract latent feature points (stable feature points) in different scale spaces, as shown in Figure 8. For each octave, σ is increased by multiplying by a blur coefficient $k = 2^{1/5}$, thus progressively increasing the blur level. After completing all scales in a set, the algorithm doubles the scale σ , downsamples the image, and halves the size of the next set of images. This process is repeated multiple times, forming a pyramid-like hierarchical structure. Next, the Gaussian blurred image difference between adjacent scales is calculated within each group. The Gaussian difference is shown in Equation (10). In the DoG scale space, a point is considered a feature point if it is the maximum or minimum value in its own layer and in the 26 neighborhoods of the layers above and below it. Furthermore, the optimization step for the feature point locations involves further removing points located at edges and with low contrast by optimizing these feature points using a three-dimensional quadratic function with Taylor expansion.

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (10)$$

3. Results and Discussion

This elaborates on the construction of standard maps and their precision analysis. Initially, by setting up a custom experimental field, the map construction dimensions are verified within a controlled environment to facilitate the validation of map construction outcomes. Emphasis is placed on optimizing parameters such as LiDAR scanning range, grid voxel filtering, and map resolution settings, directly affecting map quality and precision. Precision analysis involves using a laser rangefinder to

quantify map accuracy, analyzing discrepancies and deviations between the constructed map and actual terrain.

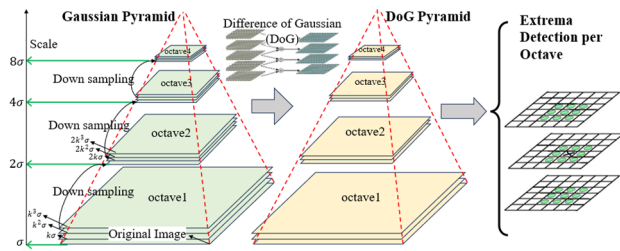


Figure 8. Gaussian blur scale space construction and extrema detection process

3.1 Map Construction Precision Analysis

This study first designed and constructed a dedicated rectangular test field measuring 2.0 m × 0.93 m, which contains a rectangular obstacle with dimensions of 0.46 m × 0.36 m. Experiments were conducted in this environment to evaluate the robot's mapping accuracy, as well as its path planning and obstacle avoidance performance. After setting the map resolution to 0.05 m, this study further analyzed its effect on the representation of obstacle dimensions within the test field. The original square obstacle placed in the test field measured 0.46 m in length and 0.36 m in width, as shown in Figure 9. After applying the map construction algorithm, the obstacle was represented on the map with dimensions of 0.45 m in length and 0.40 m in width. This change, induced by the resolution setting, resulted in adjustments to the measured free-space areas surrounding the obstacle.

Specifically, the free spaces on either side of the obstacle, originally measuring 0.77 m, were represented as 0.75 m on the map. This adjustment occurred because the LiDAR point cloud data fell within grid cells spanning from 0.75 m to 0.80 m, causing those cells to be classified as occupied. Similarly, the free distances above and below the obstacle were reduced from 0.295 m to 0.25 m. These results clearly demonstrate the direct influence of map resolution on the representation of both obstacle dimensions and surrounding free space. After accounting for the resolution-induced adjustments, the overall dimensions of the test field were determined to be 1.95 m in length and 0.90 m in width.

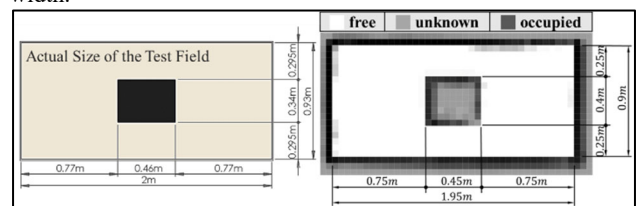


Figure 9. Comparison chart of actual field dimensions and map construction results

3.2 Defining Safe Exploration Zones and Target Selection

This section presents the establishment of safe exploration regions and target selection following the configuration of map construction parameters. Based on real-time submap data, the robot's current position is first determined, and the surrounding environment is analyzed. A binarization process is then applied to clearly separate free space from occupied regions. This process enables the navigation system to effectively distinguish navigable areas—represented in white for path planning—from non-navigable regions. To account for the robot's physical dimensions (210 × 110 mm) and kinematic constraints, including its turning radius, a 5 × 5 erosion convolution kernel is applied to the binary map. This operation reduces the free space by approximately 0.1 m around obstacles, thereby narrowing

feasible paths and preventing the robot from navigating too close to hazardous boundaries. As a result, the generated navigation corridors better reflect the robot's actual maneuvering capabilities and enhance operational safety. Figure 10 illustrates a comparison between the map before erosion and after erosion, demonstrating the effective contraction of regions considered safe for robot movement.

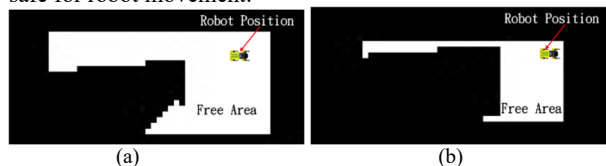


Figure 10. Comparison of maps before and after erosion processing (a) unprocessed, (b) result of free areas after erosion

3.3 Integration of Path Planning System

This research incorporates the A* search algorithm for path planning. Figure 11(a) illustrates the actual configuration of the test field along with the corresponding path planning results based on the robot's current position. In Figure 11(b), the red dot denotes the robot's location, while the red and green arrows indicate the exploration direction and target point, respectively. The planned path is represented by a solid red line, and the grid cells traversed along the path are marked with blue dots. Upon completion of the A* path planning process, the system performs backtracking from the target node, traversing each parent node along the computed path to generate a sequence of motion commands. This process enables the robot to explore the test field by following the planned trajectory. For example, when moving from grid A(x+1, y) to grid B(x, y), the system computes the required displacement and commands the robot to move a corresponding distance along the x-axis, equal to one grid unit with a resolution of 0.05 m. By sequentially publishing the planned path data to the robot controller, the robot executes the exploration tasks in accordance with the generated motion commands until all predefined actions are completed.

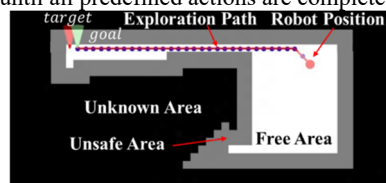


Figure 11. Path planning results under the current position using the A* search algorithm.

After analyzing the similarity between two maps, the SIFT algorithm is used to identify feature points in both maps and perform matching on these feature points. The matching criteria are set based on experimental data and the complexity of the environment. Once the criteria are met, the system declares the map building process complete and stops the map building task. If the feature matching threshold is not met, the system waits for the next detection. By continuously evaluating the map building status, unnecessary exploration activities are avoided after the task is completed. The matching results of feature points corresponding to the similarity matching between global maps are then calculated as shown Figure 12.

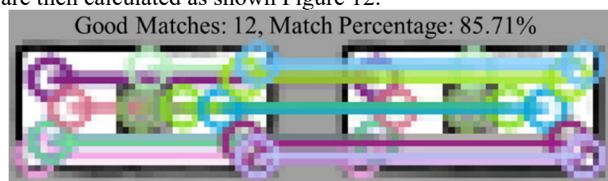


Figure 12. Matching results of feature points corresponding to the similarity matching between global maps are then calculated.

3.4 Field Detection System Construction

A map construction status detection system based on the SIFT algorithm was developed to automatically assess the completeness of robot-driven map construction and to determine an appropriate timing for declaring the mapping process complete. The system operates by comparing a predefined standard map with a real-time map in order to detect environmental changes and previously unidentified obstacles.

The standard map, shown in Fig. 13(a), was generated in a controlled environment without obstacles and was used as a stable reference. In contrast, the real-time map, illustrated in Fig. 13(b), was autonomously constructed by the robot during operation in an actual environment. Environmental variations, particularly the appearance of new obstacles, were detected through a combination of dimensional consistency checks, difference map analysis, and SIFT-based feature point matching.

Non-corresponding regions between the standard and real-time maps were highlighted in the difference map, as shown in Fig. 13(c). The detected obstacle locations were subsequently visualized on the final map, as illustrated in Fig. 13(d). By performing this comparison after each map construction cycle, the system enables the identification of newly appeared or modified obstacles, thereby providing timely and critical safety information for the robot or the operator.

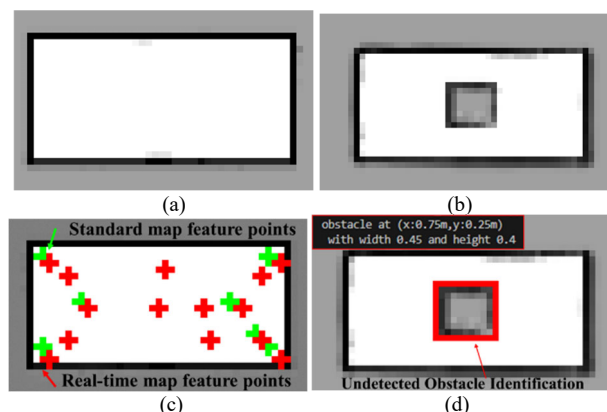


Figure 13. Obstacle detection in maps (a) Standard no-obstacle map (b) Real-time map showing obstacles (c) Identifying non-similar regions (d) Marked obstacles with details.

4. Conclusion

This study draws two main conclusions. First, real-time standard maps were constructed and analyzed with high precision using optical LiDAR data, gait measurements, and point cloud preprocessing, resulting in distortion-free maps with a resolution of 0.05 m, planimetric accuracy of 0.2 m, and precision under 0.1 m, meeting TAICS standards for high-precision mapping. Second, an intelligent path planning and detection system was developed by integrating erosion algorithms, Connected Component Labeling, Euclidean distance, the A* search algorithm, and SIFT-based obstacle detection. This system enables autonomous map exploration, identifies unspecified obstacles by comparing real-time and standard maps, and provides critical safety information, thereby enhancing robot operation in complex environments.

Acknowledgements

This work was partially supported by the Ministry of Science and Technology, under Grant No. NSTC 113-2221-E-150-010 and No. NSTC 114-2221-E-150-050, and Smart Machine and Intelligent Manufacturing Research Center.

References

- [1] Wolfgang, H., Damon, K., Holger, R., et al., 2016, "Real-time loop closure in 2D LIDAR SLAM", 2016 IEEE International Conference on Robotics and Automation (ICRA), pp. 1271-1278, Stockholm, Sweden, May.
- [2] Agarwal, S., Mierle, K., & The Ceres Solver Team. (2023). Ceres Solver (Version 2.2) [Computer software]. <https://github.com/ceres-solver/ceres-solver>
- [3] Manasi, M., Zichao, Z., Davide, S., 2020, "Voxel Map for Visual SLAM", IEEE Conference on Robotics and Automation (ICRA), Paris, Mar.
- [4] Lokesh, M., Kislay, A., Bharat, B., 2014, "BRESENHAM'S LINES ALGORITHM", International Journal of Research in Science and Technology (IJRST), Jul-Sep.
- [5] Joyce, J., 2021, "Bayes Theorem", The Stanford Encyclopedia of Philosophy, Fall 2021 Edition, California.
- [6] K A M, S., A B, J., 2021, "Analysis of Image Processing Using Morphological Erosion and Dilation", International Conference on Biomedical Engineering (ICoBE 2021), UniMAP, Malaysia (Virtual), September.
- [7] L. Di Stefano, A. Bulgarelli, 1999, "A simple and efficient connected components labeling algorithm", 10th International Conference on Image Analysis and Processing, pp. 322-327, Venice, Italy, September.
- [8] Xiang, L., Daoxiong, G., 2011, "A comparative study of A-star algorithms for search and rescue in perfect maze", 2011 International Conference on Electric Information and Control Engineering, pp. 24-27, Wuhan, April.
- [9] David, L., 2004, "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision, Canada, January.
- [10] Moi, Y., Michel, B., Ewe, H., 2003, "Gaussian Blurring-Deblurring for Improved Image Compression", Seventh International Conference on Digital Image Computing, Australia, December.
- [11] Gerardo, B., Matthew, P., Benjamin K., et al., 2018, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot", 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2245-2252, IEEE, Madrid, Spain, October.
- [12] Siddhant, G., Mathieu, G., Romeo, O., et al., 2021, "Real-Time Trajectory Adaptation for Quadrupedal Locomotion using Deep Reinforcement Learning", 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 5973-5979, June.
- [13] Alexander, W., Dario, B., Marco, H., et al., 2018, "Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization", IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 1560-1567, July.
- [14] Nathan, K., 2022, "Stanford Pupper: A Low-Cost Agile Quadruped Robot for Benchmarking and Education", arXiv, February.
- [15] Emanuel, T., Tom, E., Yuval, T., 2012, "MuJoCo: A physics engine for model-based control", 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026-5033, Vilamoura-Algarve, Portugal, October.
- [16] Joseph, N., 2022, "Quad-SDK: Full Stack Software Framework for Agile Quadrupedal Locomotion", ICRA Workshop on Legged Robots, August.