electronics

MDPI

Article

Research on SLAM and Path Planning Method of Inspection Robot in Complex Scenarios

Xiaohui Wang, Xi Ma * and Zhaowei Li

School of Electrical and Information Engineering, Beijing University of Civil Engineering and Architecture, Beijing 102616, China; wangxiaohui@bucea.edu.cn (X.W.)
* Correspondence: 2108550021054@stu.bucea.edu.cn

Abstract: Factory safety inspections are crucial for maintaining a secure production environment. Currently, inspections are predominantly performed manually on a regular basis, leading to low efficiency and a high workload. Utilizing inspection robots can significantly improve the reliability and efficiency of these tasks. The development of robot localization and path planning technologies ensures that factory inspection robots can autonomously complete their missions in complex environments. In response to the application requirements of factory inspections, this paper investigates mapping, localization, and path planning methods for robots. Considering the limitations of cameras and laser sensors due to their inherent characteristics, as well as their varying applicability in different environments, this paper proposes SLAM application systems based on multi-line laser radar and visual perception for diverse scenarios. To address the issue of low efficiency in inspection tasks, a hybrid path planning algorithm that combines the A-star algorithm and time elastic band method is introduced. This approach effectively resolves the problem of path planning becoming trapped in local optima in complex environments, subsequently enhancing the inspection efficiency of robots. Experimental results demonstrate that the designed SLAM and path planning methods can satisfy the inspection requirements of robots in complex scenarios, exhibiting excellent reliability and stability.

Keywords: complex scenes; inspection robot; SLAM; path planning

Citation: Wang, X.; Ma, X.; Li, Z. Research on SLAM and Path Planning Method of Inspection Robot in Complex Scenarios. *Electronics* **2023**, *12*, 2178. https://doi.org/10.3390/electronics12102178

Academic Editor: Cecilio Angulo

Received: 10 April 2023
Revised: 5 May 2023
Accepted: 8 May 2023
Published: 10 May 2023

1. Introduction

With the introduction of "Industry 4.0", robotics technology has rapidly advanced. Among such technologies, inspection robots have been extensively used in aerospace, manufacturing, agriculture, service industries, and other fields due to their superior flexibility, mobility, and functionality [1,2]. Xu et al. proposed the pulsed eddy current testing (PECT) method, which is sensitive to defects in brazed foils. By using a robotic arm, the PECT probe is scanned above the panel specimen in motion. This method can effectively detect incomplete brazing defects in local areas of stainless steel core panels [3]. Foumani et al. proposed a method to minimize the partial cycle time of such cells for three different inspection scenarios: in-process, post-process, and in-line. This method was used to evaluate a basic two-machine robotic rework cell and determine whether it is technically profitable to replace an in-process (or post-process) inspection scenario with an in-line inspection scenario [4]. As inspection application scenarios become more diversified and complex, higher requirements are placed on the autonomous navigation performance of robots. Robot navigation technology mainly consists of SLAM technology and path planning technology. Simultaneous Localization and Mapping (SLAM) is the process by which a mobile robot determines its own position and creates a map through sensors carried in the surrounding environment. Path planning technology creates the optimal navigation path for the robot to reach the target location based on different task goals and requirements.

SLAM technology can be classified into two categories based on different sensors: vision-based and LiDAR-based. The MonoSLAM method proposed by Davison et al. estimates the camera pose by extracting sparse feature points frame by frame, which was

the first real-time visual SLAM system using a single camera [5]. Subsequently, Klein et al. proposed the PTAM method, which introduced nonlinear optimization and a keyframe mechanism [6], solving the problem of high computational complexity in MonoSLAM. Newcombe et al. proposed a dense per-pixel method based on an RGB-D camera, which can achieve real-time tracking and reconstruction [7]. As for LiDAR-based SLAM methods, Grisetti et al. proposed a 2D SLAM algorithm based on particle filtering, which solved the problem of particle dissipation caused by resampling by reducing the number of particles [8]. Konolige et al. proposed the first open-source algorithm based on graph optimization by using highly optimized and non-iterative square-root factorization to sparsify and decouple the system [9]. Kohlbrecher et al. designed the Hector SLAM algorithm, which matches the current frame's LiDAR data with the factor graph and optimizes the pose using the Gauss–Newton method to obtain the optimal solution and bias [10]. Sensor fusion is also crucial. Xia et al. proposed an autonomous vehicle sideslip angle estimation algorithm based on consensus and vehicle kinematics/dynamics synthesis, which solves the observability issue of the heading error by utilizing a multisensor fusion framework incorporating GNSS, IMU, and onboard sensors. This approach enhances the reliability and accuracy of sideslip angle estimation under various automated driving conditions [11]. Xia et al. proposed a method for the IMU and automotive onboard sensors to estimate the yaw misalignment autonomously by analyzing the observability of yaw misalignment through the piece-wise constant system (PWCS) and singular value decomposition (SVD) theory, addressing the difficulty of directly measuring yaw misalignment [12]. Liu et al. proposed a new YOLOv5-tassel algorithm, which introduced the attention mechanism of SimAM. This improves the accuracy of tassel detection in UAV-based RGB imagery [13]. Xia et al. proposed a data acquisition and processing platform for automated driving systems (ADSs) based on connected automated vehicle (CAV) cooperative perception, which is utilized for vehicle trajectory extraction, reconstruction, and evaluation, addressing challenges such as noise, outliers, and ID switches in object detection and tracking [14].

The aim of path planning is to plan an optimal collision-free path from the starting point to the destination point in the mapped environment. Path planning can be divided into global and local path planning according to whether the environmental information is known in advance. Dijkstra et al. proposed the shortest path planning algorithm, which uses a breadth-first search to search for paths [15]. Hart et al. proposed the A-star algorithm, which reduces the search nodes by using a heuristic evaluation function and improves the efficiency of path searching [16]. Fox et al. proposed DWA, which dynamically samples the velocity in the robot's sampling space according to the robot's kinematic model and current motion parameters and selects the best trajectory [17]. To address the insufficient evaluation function of DWA, Chang et al. proposed an improved DWA algorithm based on Q-learning, which modifies and extends the evaluation function and adds two evaluation functions to improve the navigation performance and achieve a higher navigation efficiency and success rate in complex unknown environments [18]. Rösmann et al. proposed the time elastic band algorithm based on multi-objective optimization, which ensures that the robot can output a smooth trajectory under the premise of satisfying its kinematic constraints [19].

Both LiDAR and camera sensors are essential for factory inspections, but their applicability depends on the specific application scenarios and factory environments. LiDAR is suitable for long-distance and high-precision measurements, such as inventory management in large warehouses and position control in robot operations. However, LiDAR has limitations in processing details and colors, making it unsuitable for scenes with high visual requirements. Camera sensors, with functions such as image recognition, detection, and tracking, are ideal for environments that require high-precision visual detection and recognition, such as detecting product dimensions, shapes, and colors and performing automated visual inspections during assembly processes on production lines. In addition, camera sensors have strong processing capabilities for details and colors, providing more detailed image information. However, camera sensors are highly dependent on ambient

light, which can limit their effectiveness in poor lighting conditions or when obstacles are present.

This article describes the navigation system of an inspection robot designed for complex real-world scenarios, as shown in Figure 1. We designed two sets of SLAM systems based on multi-line LiDAR and vision sensors to meet the inspection needs of robots in different environments. For the visual SLAM system, RGB-D images were first collected as external inputs, and then the features of the images were extracted, loop-closure detection was performed, and global mapping optimization was used to obtain a point cloud map. For the laser SLAM system, laser radar scans of the environment were collected first, and then features such as corners and edges were extracted from the scans. Finally, these features were used to construct a 3D point cloud map of the environment. After obtaining the point cloud map of the visual or laser SLAM system, the A-star algorithm was used for global path planning to improve navigation efficiency and help the robot quickly plan the optimal path. Then, the time elastic band algorithm was introduced to perform real-time path planning based on changes in the environment and obstacles. Ultimately, autonomous navigation and obstacle avoidance of the inspection robot were achieved in complex scenarios. The main contributions are listed as follows:

- We designed a SLAM application system based on multi-line laser radar and vision that can be applied to different scenarios.
- We propose a hybrid path planning algorithm that combines the A-star algorithm and time elastic band algorithm. It effectively solves the problem of local optima in path planning in complex environments, improving robot inspection efficiency.
- The two SLAM application systems share a set of hybrid path planning algorithms to achieve high-precision navigation inspection tasks.

The overall structure of this article is as follows. Section 2 presents the SLAM systems based on visual sensors and multi-line LiDAR sensors. Section 3 introduces a hybrid path planning algorithm that combines the A-star algorithm as the global path planning algorithm and the TEB algorithm as the local path planning algorithm. Section 4 describes the performance evaluation of visual SLAM, LiDAR SLAM, and hybrid path planning algorithms on the ROS platform. Finally, Section 5 concludes this work and provides future outlooks.

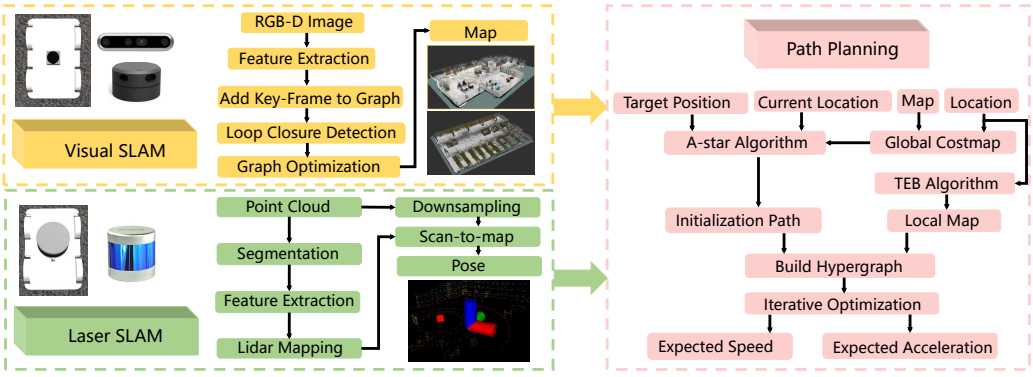


Figure 1. System framework.

2. Inspection Robot SLAM System

2.1. Visual SLAM Algorithm Design and Implementation

In the factory environment, the position and motion of inspection robots and equipment may undergo rapid changes, thus requiring the real-time acquisition and processing of sensor data for accurate localization and mapping. To achieve this, we chose an appearance-based localization and mapping method that is invariant to time and scale, as shown in the structural diagram in Figure 2.

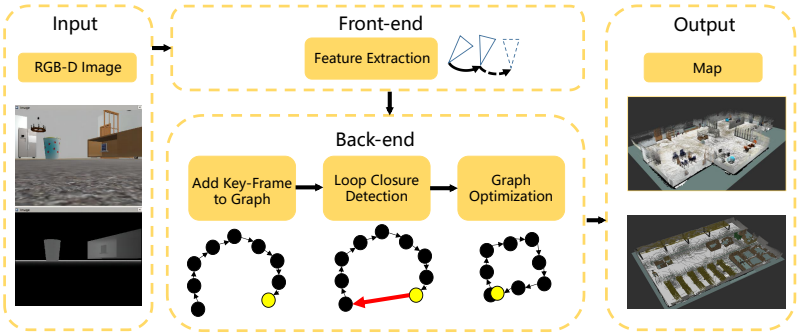


Figure 2. Vision-based SLAM system structure diagram.

In Figure 2, it can be seen that RGB-D images are used as external input, and the ORB algorithm is used to extract feature points from the RGB-D images [20]. Then, a Bag-of-Words-based image matching method is used to match feature points between adjacent frames. The loop-closure detection mechanism is introduced to eliminate drift. When the robot returns to an area previously visited, loop-closure detection can identify the area and match the newly observed data with the previous map data, thus addressing the cumulative drift problem. Next, graph optimization is performed, where the robot's poses are represented as nodes in the graph, and the observed data are represented as edges. Then, the least-squares method is used to optimize the positions of all nodes, minimizing the error between observed and predicted data. Finally, a dense map can be generated.

This vision-based mapping and localization algorithm utilizes global and local loop-closure detection techniques, which can identify and handle errors and drift in sensor data, improving the robustness and accuracy of localization and mapping and enhancing the efficiency and real-time performance of inspection robots.

2.2. Multi-Line LiDAR-Based SLAM Algorithm Design and Implementation

In indoor factory environments, lighting conditions can vary with time and location, which may lead to the misidentification of objects or inaccurate positioning by visual sensors. However, laser sensors do not require an external light source, as they emit their own laser beams and are not affected by lighting issues. Therefore, we adopted a lightweight and ground-optimized laser odometry and mapping method, whose structural diagram is shown in Figure 3. Firstly, the laser point cloud data are dimensionally reduced by projecting the 3D laser onto a 2D depth image, segmenting the ground according to the pitch angle, clustering non-ground point clouds, and obtaining labeled point cloud data. Then, feature extraction is performed based on smoothness, resulting in four sets of feature point clouds. Constraint relationships are established for the feature point cloud sets, and the 6-DOF pose transformation matrix is solved using the Levenberg–Marquardt (LM) optimization method. Subsequently, loop detection is conducted using the Iterative Closest Point (ICP) algorithm, and finally, the current point cloud is mapped to the global map based on graph optimization, completing the establishment of a high-precision map [21].

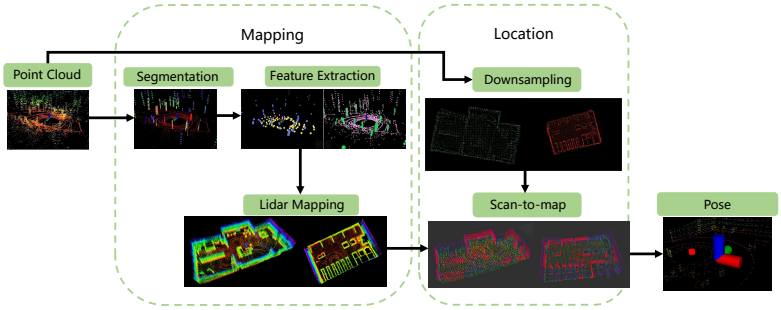


Figure 3. Multi-line LiDAR-based SLAM system structure diagram.

(1) Point cloud segmentation: Due to the complexity of the inspection environment and other factors, noise may exist in the laser point cloud data. We first use point cloud segmentation to filter out noise. By projecting a frame of the 3D point cloud onto a 2D depth image using a projection method, ground segmentation is performed to separate non-ground points [22]. Let $P_t = p_1, p_2, \ldots, p_n$ be the point cloud data obtained by the LiDAR at time t, where P_i is a point in P_t. These points are projected onto a depth image, and the 3D points in space become 2D pixels in space. After projection, the Euclidean distance r_i of point P_i to the sensor is obtained. Since the 3D point cloud contains a large amount of ground information, it is necessary to filter the point cloud to improve the efficiency and accuracy of feature extraction. Firstly, the ground points are labeled, and the labeled ground points will no longer be segmented in subsequent steps. After separating ground points and non-ground points, the non-ground points are processed by clustering. After this module, each point has its own segmentation label (ground or non-ground), row and column indices in the depth image, and the Euclidean distance r_i to the sensor.

(2) Feature extraction: According to the smoothness, the projected depth image is horizontally divided into several sub-images. For each sub-image, the following process is performed [23]: Let S be the set of continuous points in the same row in the depth image, and calculate the smoothness c of point P_i.

$$c = \frac{1}{|S| \cdot \|r_i\|} \left\| \sum_{j \in S, j \neq i} (r_j - r_i) \right\| \tag{1}$$

where r_i and r_j are the Euclidean distances from points P_i and P_j to the sensor. According to Equation (1), the smoothness of each point can be calculated, and then the smoothness is sorted. After sorting, feature points are selected. Different types of features are segmented based on the set threshold c_{th}. The edge points with a smoothness c greater than c_{th} are selected as set F_{me}, and the plane points with a smoothness c less than c_{th} are selected as set F_{mp}. The largest $n_{F_{me}}$ edge points with the maximum c value and the smallest $n_{F_{mp}}$ plane points with the minimum c value are selected from all sub-images to form the edge feature point set F_{me} and the plane feature point set F_{mp}. Then, n_{F_e} edge points not belonging to ground points with the maximum c value are selected from set F_{me} to form set F_e, and n_{F_p} plane points belonging to ground points with the minimum c value are selected from set F_{mp} to form set F_p. Obviously, $F_e \subset F_{me}$ and $F_p \subset F_{mp}$.

(3) Radar odometry: The odometry module estimates the robot's pose change between adjacent frames using a radar sensor. In the estimation process, tag matching is used to narrow down the matching range and improve accuracy, and a two-step LM optimization method is used to find the transformation relationship between two consecutive frames [24]. The first step uses ground feature points F_p to obtain $\left[t_z, t_{roll}, t_{pitch}\right]$, and the second step matches the edge features extracted from the segmented point cloud to obtain the transformation $\left[t_x, t_y, t_{yaw}\right]$. Finally, by fusing $\left[t_z, t_{roll}, t_{pitch}\right]$ and $\left[t_x, t_y, t_{yaw}\right]$, a 6-DOF pose transformation matrix $\left[t_x, t_y, t_z, t_{roll}, t_{pitch}, t_{yaw}\right]$ is obtained.

(4) Radar mapping: After obtaining the pose change between adjacent frames with radar odometry, the features in the feature set F_{me}^t, F_{mp}^t at time t are matched with the surrounding point cloud Q^{t-1} to further refine the pose transformation. Then, using LM optimization, the final transformed pose is obtained, and the pose graph is sent to GTSAM for map optimization to update the sensor-estimated pose and the current map [25].

In addition, noise may exist in the collected laser point cloud data. In order to achieve high-precision localization on the map, it is necessary to preprocess the high-precision map. We use a statistical-based robust filter to remove outliers, a pass-through filter to clip the point cloud within a specified coordinate range, and a voxel grid filter to downsample the point cloud. For an inspection task that requires high-precision real-time localization, we perform real-time localization on the constructed high-precision map through point cloud registration. First, the reference point cloud (i.e., high-precision map) is transformed into

a multivariate normal distribution [26]. If the transformation parameters can accurately match the reference point cloud and the current point cloud, the transformed points in the reference frame have a high probability density. Therefore, an optimization method can be used to calculate the transformation parameters that maximize the sum of the probability densities. In this case, the two sets of laser point cloud data match best. The specific algorithm steps are as follows:

(1) Given the current scanning point cloud S and the reference point cloud T, the space occupied by the T point cloud is divided into voxel grids of a specified size, and the expected vector $\vec{\mu}$ and covariance matrix Σ of N points in each voxel grid are calculated.

$$\vec{\mu} = \frac{1}{N}\sum_{k=1}^{N}\vec{x}_k \tag{2}$$

$$\Sigma = \frac{1}{N-1}\sum_{k=1}^{N}(\vec{x}_k - \vec{\mu})(\vec{x}_k - \vec{\mu})^T \tag{3}$$

Here, $\vec{x}_k$ represents the three-dimensional coordinates of the point cloud in the voxel grid.

(2) Initialize the transformation parameters to be solved, with zero values or odometry data. For each sample $\vec{x}_k$ in the S point cloud, transform it into the T point cloud according to the transformation parameters. Let $\vec{x}'_k$ be the coordinate of $\vec{x}_k$ in the T point cloud coordinate system. Find the grid where $\vec{x}'_k$ falls in the T point cloud, and combine the probability density function of each grid in the T point cloud to calculate the corresponding probability distribution function $p(\vec{x}'_k)$.

$$p(\vec{x}'_k) \sim \exp\left(-\frac{(\vec{x}'_k - \mu_k)^T \sum_k^{-1}(\vec{x}'_k - \mu_k)}{2}\right) \tag{4}$$

(3) Add up the probability densities calculated for each mapped point to obtain the registration score.

$$\text{score}(p) = \sum_k \exp\left(-\frac{(\vec{x}'_k - \mu_k)^T \sum_k^{-1}(\vec{x}'_k - \mu_k)}{2}\right) \tag{5}$$

Use the Newton optimization algorithm to optimize the objective function until the optimal transformation parameters are found that maximize the registration score, complete convergence, and solve the best rigid body transformation between the target and source point clouds to achieve accurate localization.

3. Inspection Robot Path Planning System

3.1. Sports Model

Currently, the chassis of inspection robots mainly consists of legged, tracked, and wheeled types, each with its advantages and disadvantages in different environments. Legged inspection robots have strong terrain adaptability, but their structure and control system are complex. Tracked inspection robots have high traction and strong applicability in complex terrains such as outdoor, sandy, and muddy areas, but their speed is relatively low, and they have high motion noise. Wheeled inspection robots have fast speed, high efficiency, and low motion noise and are widely used. In this paper, we focus on the complex indoor factory environment and adopt a two-wheel differential wheeled robot, whose kinematic model is shown in Figure 4 [27].

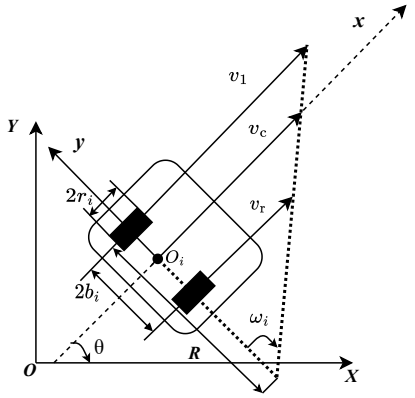


Figure 4. Two-wheel differential robot model.

In Figure 4, the motion of robot $R_i(i = 1, 2, \ldots, n)$ is completed by two independently driven wheels. Let the radius of the driving wheel be r_i, and define the midpoint of the two driving wheels as O_i. The distance between the two wheels is $2b_i$, where $\{O, X, Y\}$ is the inertial Cartesian coordinate system, and $\{O_i, X, Y\}$ is the local coordinate system of the robot. v_l is the speed of the left driving wheel, v_r is the speed of the right wheel, and v_c is the speed of the center of the robot. If $v_l \neq v_\mathrm{r}$, the angular velocity ω_i can be obtained. According to the robot model, the forward speed depends on the speed of the wheels.

$$v_\mathrm{c} = \frac{v_l + v_\mathrm{r}}{2} \tag{6}$$

The angular velocity ω is determined by the difference in speed between the left and right driving wheels and the distance between them.

$$\omega = \frac{v_l - v_\mathrm{r}}{2b_i} \tag{7}$$

In the ideal case, according to the principle of rigid motion, the trajectory of the robot is a circle, and the radius can be expressed as

$$R = \frac{v_\mathrm{c}}{\omega} = \frac{b_i(v_l + v_\mathrm{r})}{(v_l - v_\mathrm{r})} \tag{8}$$

The kinematic equation of the robot can be expressed as

$$q = \begin{bmatrix} x & y & \theta \end{bmatrix}^\mathrm{T} u \tag{9}$$

$$V = \begin{bmatrix} v_c & \omega \end{bmatrix}^\mathrm{T} \tag{10}$$

$$\begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^\mathrm{T} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix} \tag{11}$$

The above formula is the pose state matrix and motion state matrix, and the final formula can be written as follows:

$$\dot{q} = S(q)V = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_\mathrm{c} \\ \omega \end{bmatrix} \tag{12}$$

where $S(q)$ is a 3 $\times$ 2 smooth linearly independent matrix, and V is the motion matrix of the robot.

3.2. Path Planning

The path planning of inspection robots mainly relies on the constructed grid map to generate a safe and collision-free path by specifying the start and target locations. Path planning for robots can be divided into global path planning and local path planning.

(1) Global path planning

To ensure that the patrol robot can effectively avoid obstacles globally and locally, and considering that the grid map of the actual road scene is relatively simple, the A-star algorithm is used as the global path planning method to provide accurate obstacle avoidance directions for the robot through real-time planning [28,29]. A-star combines a heuristic search with a breadth-first algorithm to select the search direction through the cost function $f(n)$ and expands around the starting point. The cost value of each surrounding node is calculated by the heuristic function $h(n)$, and the minimum cost value is selected as the next expanding point. This process is repeated until the endpoint is reached, generating a path from the starting point to the endpoint. In the search process, since each node on the path is the node with the minimum cost, the cost of the path obtained is also minimum. The cost function of the A-star algorithm is

$$f(n) = g(n) + h(n) \tag{13}$$

where $f(n)$ is the cost function at the current position, $g(n)$ is the cost value from the starting position to the current position in the search space, and $h(n)$ is the cost value from the current position to the goal position. In the A-star algorithm, the selection of the heuristic function is crucial. Since the mapped environment is a grid map with obstacles, the Manhattan distance is used as the heuristic function, which is given by:

$$h(n) = |x_1 - x_2| + |y_1 - y_2| \tag{14}$$

where (x_1, y_1),(x_2, y_2) represent the coordinates from the current position to the target position. In the path planning of the A-star algorithm, the nodes are stored in two lists, Closelist and Openlist. The nodes that have been searched and generated cost values are stored in Openlist. The node with the minimum estimated cost is stored in Closelist, and the moving trajectory is formed by processing the trajectories of each node in Closelist. The specific steps are as follows:

Step 1. The starting point s of the robot is the first calculated point, the surrounding nodes are added to Openlist, and the cost function $f(n)$ of each point is calculated.

Step 2. Openlist is searched, and the node with the smallest cost value $f(n)$ is selected as the current processing node n, removed from Openlist, and put into Closelist.

Step 3. If the real cost value $g(n)$ of the adjacent node from the current processing node to the starting point s is smaller than the original $g(n)$ value, the parent node of the adjacent node is set to the current processing node; if it is larger, the current processing node is removed from Closelist, and the node with the second-smallest value of $f(n)$ is selected as the current processing node.

Step 4. The above steps are repeated until the target point g is added to Closelist; each parent node is traversed, and the obtained node coordinates are the path.

(2) Local path planning

The working environment of inspection robots is not always static. In the process of moving along the global path, real-time obstacles may appear. To avoid collisions, the timed elastic band (TEB) algorithm, which introduces local path planning with time elasticity, is used on the basis of global path planning to achieve real-time obstacle avoidance [30]. The TEB algorithm is an optimization algorithm that follows the path generated by the global path planner. The local trajectory it generates is composed of a series of continuous time and pose sequences, and the robot's pose is defined as:

$$X_i = [x_i, y_i, \beta_i]^T \tag{15}$$

where X_i represents the i-th pose in the robot coordinate system, including the position information x_i, y_i and angle β_i. The time interval between adjacent poses X_i and X_{i+1} is denoted by ΔT_i, as shown in Figure 5.

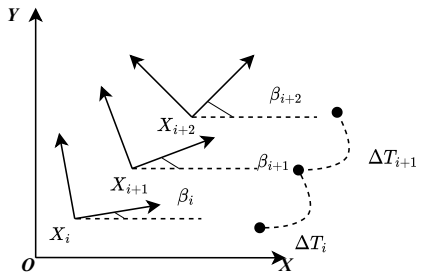


Figure 5. Time interval and pose sequence of the TEB.

In the optimization process, the TEB algorithm applies graph optimization to the adjacent time intervals and states of the robot as nodes and uses the velocity, acceleration, and non-holonomic constraints of the robot as edges. It also considers obstacle information, the discrete interval of the planned trajectory, and adjacent temporal and spatial sequence constraints. Finally, the G2O solver is used to calculate the control variable $V(v, \omega)$ (where v and ω represent the linear and angular velocities of the robot, respectively) to obtain the optimal trajectory. The TEB algorithm obtains the optimal pose points through weighted multi-objective optimization [31,32], where the mathematical description of the objective function is:

$$f(B) = \sum_k \gamma_k f_k(B) \tag{16}$$

$$B^* = \arg\min_B f(B) \tag{17}$$

where $f(B)$ is the objective function that considers various constraints, $f_k(B)$ is the constraint function, γ_k is the weight of each item, and B^* is the optimal TEB trajectory. The TEB algorithm has four constraint functions.

1. Path following and obstacle constraint objective function

The TEB algorithm aims to avoid collisions with static or dynamic obstacles while following the path. The algorithm treats piecewise continuous and differentiable functions as constraints and punishes behaviors that do not conform to the constraints. Specifically:

$$e_\Gamma(x, x_r, \varepsilon, S, n) \approx \begin{cases} ((x - (x_r - \varepsilon))/S)^n & x > x_r - \varepsilon \\ 0 & \text{other} \end{cases} \tag{18}$$

$$f_{\text{path}} = e_\Gamma\left(d_{\min,j}, r_{p_{\max}}, \varepsilon, S, n\right) \tag{19}$$

$$f_{\text{ob}} = e_\Gamma\left(-d_{\min,j}, -r_{p_{\min}}, \varepsilon, S, n\right) \tag{20}$$

Building on Equation (18), penalty functions f_{path} and f_{ob} are constructed. Here, x_r denotes the boundary, ε is the offset factor, S is the scaling factor, n is the order, $d_{\min,j}$ is the independent variable representing the distance between the path point and obstacle, $r_{P_{\max}}$ is the maximum distance of the trajectory deviation from the path point, and $r_{o_{\min}}$ is the minimum distance between the trajectory and obstacle.

2. The velocity and acceleration constraint functions of a robot

According to the dynamic equation, the constraint functions of the robot's velocity and acceleration are expressed as Equations (21)–(24):

Linear velocity:

$$v_i \approx \frac{1}{\Delta T_i}\begin{pmatrix} x_{i+1}-x_i \\ y_{i+1}-y_i \end{pmatrix} \tag{21}$$

Angular velocity:

$$w_i \approx \frac{\theta_{i+1}-\theta_i}{\Delta T_i} \tag{22}$$

Linear acceleration:

$$a_i \approx \frac{2(v_{i+1}-v_i)}{\Delta T_i + \Delta T_{i+1}} \tag{23}$$

Angular acceleration:

$$\alpha_i = \frac{2(w_{i+1}-w_i)}{(\Delta T_i + \Delta T_{i+1})} \tag{24}$$

3. Non-holonomic constraint

The robot used in the algorithm simulation and experiment is a differential drive structure with two degrees of freedom, which cannot perform translational motion along the y-axis of the robot coordinate system. The curvature of the circular arc between two adjacent robot poses is approximately constant, and the outer product of the direction vector $d_{i,i+1}$ and the turning angle θ_i between adjacent poses in the robot coordinate system is equal to the outer product of the turning angle θ_{i+1} and the direction vector $d_{i,i+1}$. β_i represents the orientation of the robot in the global coordinate system, and the corresponding relationship equation and non-holonomic constraint are:

$$\begin{pmatrix} \cos\beta_i \\ \sin\beta_i \\ 0 \end{pmatrix} \times d_{i,i+1} = d_{i,i+1} \times \begin{pmatrix} \cos\beta_{i+1} \\ \sin\beta_{i+1} \\ 0 \end{pmatrix} \Leftrightarrow \theta_i = \theta_{i+1} \tag{25}$$

$$d_{i+1} := \begin{pmatrix} x_{i+1}-x_i \\ y_{i+1}-y_i \\ 0 \end{pmatrix} \tag{26}$$

$$f_k(X_i, X_{i+1}) = \left\| \left[\begin{pmatrix} \cos\beta_i \\ \sin\beta_i \\ 0 \end{pmatrix} + \begin{pmatrix} \cos\beta_{i+1} \\ \sin\beta_{i+1} \\ 0 \end{pmatrix} \right] \times d_{i,i+1} \right\|^2 \tag{27}$$

The objective function $f_k(X_i, X_{i+1})$ punishes the quadratic error for violating this constraint, ensuring that the output velocity of the robot follows the non-holonomic constraint. However, when the vehicle experiences significant lateral motion, the fidelity of the non-holonomic constraint in the lateral direction is lost. A combined vehicle dynamics/kinematics-based lateral velocity estimation algorithm can be applied, which leverages the advantages of the vehicle dynamics model under low dynamic driving conditions and the kinematic model under high dynamic driving conditions [33].

4. Fastest-path constraint

The TEB algorithm incorporates the time interval information between poses, and the total time is the sum of all time intervals. The relevant objective function is:

$$f_k = \left(\sum_{i+1}^{n} \Delta T_i \right)^2 \tag{28}$$

After optimizing the TEB sequence, the objective function of the constraints is optimized to ensure that the path planned by the algorithm achieves the best results in terms of obstacle avoidance, time, and distance.

(3) Path planning based on fusion algorithm

The A-star algorithm yields a navigation path consisting of only the start point, key points, and destination point, but it cannot avoid unknown obstacles in the environment. The TEB algorithm exhibits good local obstacle avoidance ability, but with only a single final goal point as a guide, it is prone to becoming trapped in local optima. Therefore, we propose a hybrid path planning algorithm that combines the strengths of both algorithms. The specific algorithm process is shown in Figure 6.

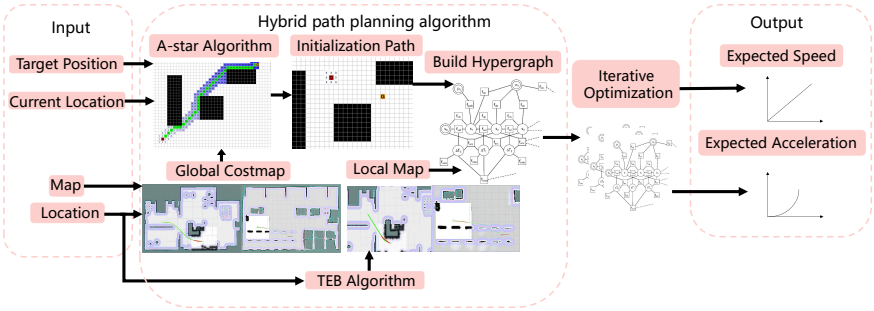


Figure 6. Hybrid path planning algorithm flow chart.

Global path planning takes a static obstacle cost map as input and does not consider the robot's mechanical performance and kinematic constraints when planning the path. It uses the A-star algorithm to plan the optimal path from the robot's current position to the desired target position and provides an initial value for local planning.

Local path planning collects path nodes on the global optimal path and optimizes the global path subset between the robot's current node and the collected path nodes. It combines the static obstacle cost map and dynamic obstacle cost map and uses the TEB algorithm to continuously adjust the pose and orientation of the robot during its movement, taking into account its shape, dynamic model, and motion performance in the scope of local planning. When encountering dynamic obstacles, it removes the old robot pose and adds a new robot pose so that a new path can be generated in each iteration, and an optimized path can be obtained through continuous iteration.

By fusing navigation algorithms, we achieve optimal global path planning and real-time obstacle avoidance functionality in the process of mobile robot navigation.

4. Experiment and Analysis

4.1. Experimental Settings

In order to verify the effectiveness of navigation systems in practical applications, we conducted experiments using a Turtlebot3 mobile robot in different types of scenarios built using the Gazebo simulation platform within the ROS on a 64-bit Ubuntu 18.04 operating system with 4GB of running memory. As shown in Figure 7, we constructed a home environment and a factory warehouse environment to simulate real-world environments. Using the real-time localization and mapping capabilities in Rviz, we scanned the simulated environments, constructed corresponding maps, and performed path planning. The code used in this article is open source and can be accessed at https://github.com/Mxiii99/RSPP_CS.git (accessed on 7 February 2023).

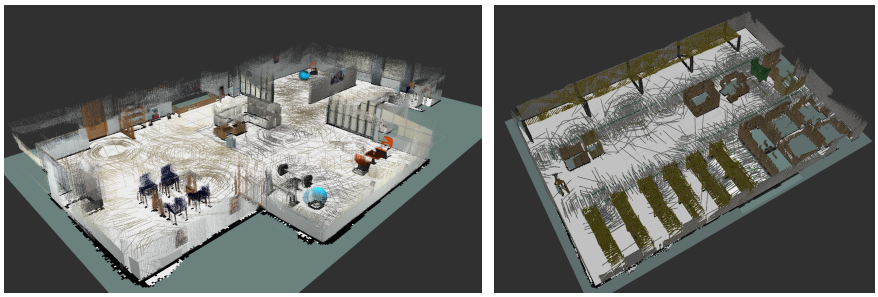
(**a**) (**b**)

Figure 7. Simulation environment. (**a**) House scene; (**b**) factory warehouse scene.

4.2. Performance Evaluation

4.2.1. Visual SLAM Algorithm Performance Evaluation

This paper presents a method for constructing a corresponding point cloud map using a depth camera in a ROS environment. The depth camera data are first read in the ROS environment, and then the front-end and back-end threads are executed to construct a sparse feature point map, which is continuously updated to create a real-time point cloud map. Keyframes from the front-end are passed into the point cloud construction thread to generate the point cloud map. The effectiveness of the proposed algorithm for generating maps is validated by the corresponding point cloud map in Figure 8, which demonstrates good 3D effects for constructing maps in indoor environments. As shown in Figure 8, the algorithm detects the object's motion trajectory, which is consistent with the actual trajectory. Although there are deviations between the detected trajectory and the actual trajectory, there are no serious deviations, which satisfies the perception requirements of the robot. When the object's motion trajectory changes significantly, there are still no serious deviations, which also meets the perception requirements of the robot.



(**a**) (**b**)

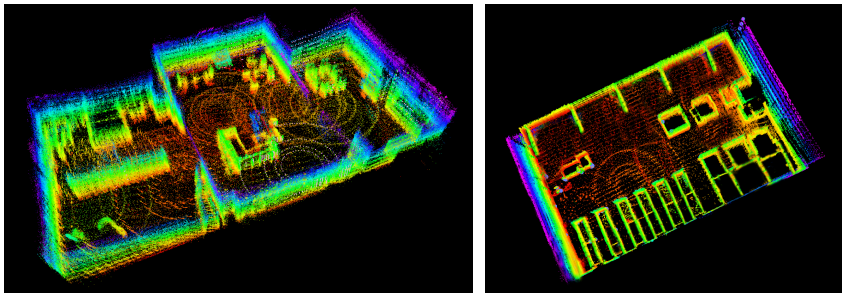
Figure 8. Vision-based mapping results. (**a**) House scene; (**b**) factory warehouse scene.

4.2.2. Multi-Line LiDAR-Based SLAM Algorithm Performance Evaluation

From the comparison between point cloud mapping and visual mapping, it can be observed that maps constructed using multi-line laser scanning are clearer than those constructed using visual algorithms, which reduces accumulated errors and provides better handling of edge contours. Furthermore, the mapping time, mapping effectiveness, and CPU utilization were compared in order to validate the feasibility, reliability, and accuracy of the algorithm.

To ensure the accuracy of the experiments, multiple tests were conducted. The robot was fixed at a certain position, denoted as the origin (0,0), and the output object motion data were compared with the actual object motion data. The results are shown in Figure 9. The algorithm detected that the point cloud map was generally consistent with the simulated scene and that the detected trajectory did not significantly deviate from the actual trajectory, which satisfies the perception requirements of the robot. As shown in the figure,

when there were large changes in the object's motion trajectory, there was a slight deviation between the detected trajectory and the actual trajectory, but no serious deviations occurred, which still satisfies the perception requirements of the robot.



(**a**) (**b**)

Figure 9. Laser-based mapping results. (**a**) House scene; (**b**) factory warehouse scene.

4.2.3. Path Planning Performance Evaluation

Through testing, the path planned by the A-star algorithm maintains a certain distance from obstacles, which prevents the robot from colliding with them. At the same time, global path planning has a good effect and can accurately reach the set target point location, satisfying the requirement for precise navigation. The robot moves along a square path. When encountering obstacles, it autonomously avoids them through local path planning. The process and result of local path planning are shown in Figure 10. After configuring the relevant parameters, the 3D view area of robot navigation was observed in Rviz. The environment of the map is displayed as a global cost map, and the environment around the robot is a local cost map. The blue area is the expansion layer of the obstacle, which is expanded outward on the map to avoid a collision between the robot and the obstacle. By adding the Path plugin in RViz, the path along which the robot moves can be seen. The green line is the route of global path planning, and the red line is the route of local path planning. It can be seen in Figure 10 that the local path planning route of the inspection robot is smooth, and the planned route does not enter the expansion layer of the obstacle, so it can reasonably avoid the surrounding obstacles and has a good obstacle avoidance effect. The global path planning route is shown in Figure 11. After testing, the inspection robot can accurately achieve autonomous obstacle avoidance and complete local path planning for the set target point, satisfying the requirement for precise navigation.

(**a**) (**b**)

Figure 10. Local path planning map. (**a**) House scene; (**b**) factory warehouse scene.

(**a**) (**b**)

Figure 11. Global path planning map. (**a**) House scene; (**b**) factory warehouse scene.

5. Conclusions and Outlook

The utilization of intelligent inspection robots has been shown to enhance production efficiency and reduce costs. However, the complex factory environment, filled with machinery equipment, pipelines, cables, and other obstacles, can pose a challenge to accurate inspections. To address this, we developed a high-precision navigation inspection system that is specifically designed for complex factory scenes. The system is equipped with two types of sensors, visual and LiDAR, to allow for rich environmental information and localization and mapping. Optimal path planning is achieved by combining the A-star algorithm and TEB algorithm for dynamic programming. To evaluate the performance of the navigation system, simulations were conducted in two scenarios using Gazebo simulation software in the ROS system: a residential area and a factory warehouse. The results indicate that the navigation system provides real-time localization and map construction, can navigate mobile platforms, and implements real-time obstacle avoidance in different scenarios. As such, this technology can be applied to the localization and navigation system of wheeled inspection robots in various complex environments and has significant reference value. As inspection scenarios become increasingly complex, a single mobile robot may find it difficult to complete navigation tasks accurately. Therefore, in the future, further research can be conducted on the problem of cooperative navigation and obstacle avoidance among multiple mobile robots.

Author Contributions: Conceptualization, X.M. and X.W.; methodology, X.M. and X.W.; software, X.M. and X.W.; validation, X.M., X.W. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Graduate Education and Teaching Quality Improvement Project of Beijing University of Architecture and Architecture (J2023017); the Lecturer Support Plan Project of Beijing University of Architecture and Architecture (YXZJ20220804); the Open Project of Anhui Provincial Key Laboratory of Intelligent Building and Building Energy Efficiency, Anhui Jianzhu University (IBES2020KF06); and the BUCEA Post Graduate Innovation Project.

Informed Consent Statement: For studies not involving humans or animals.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bahrin, M.A.K.; Othman, M.F.; Azli, N.H.N.; Talib, M.F. Industry 4.0: A review on industrial automation and robotic. *J. Teknol.* **2016**, *78*, 6–13.
2. Choi, H.; Ryew, S. Robotic system with active steering capability for internal inspection of urban gas pipelines. *Mechatronics* **2002**, *12*, 713–736. [CrossRef]
3. Xu, Z.; Chen, H.; Qu, Z.; Zhu, C.; Wang, X. Nondestructive testing of local incomplete brazing defect in stainless steel core panel using pulsed eddy current. *Materials* **2022**, *15*, 5689. [CrossRef]
4. Foumani, M.; Smith-Miles, K.; Gunawan, I. Scheduling of two-machine robotic rework cells: In-process, post-process and in-line inspection scenarios. *Robot. Auton. Syst.* **2017**, *91*, 210–225. [CrossRef]

5. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [CrossRef] [PubMed]
6. Pire, T.; Fischer, T.; Castro, G.; De Cristóforis, P.; Civera, J.; Berlles, J.J. S-PTAM: Stereo parallel tracking and mapping. *Robot. Auton. Syst.* **2017**, *93*, 27–42. [CrossRef]
7. Zhou, H.; Ummenhofer, B.; Brox, T. *DeepTAM: Deep Tracking and Mapping with Convolutional Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2020; Volume 128, pp. 756–769.
8. Grisetti, G.; Kümmerle, R.; Stachniss, C.; Burgard, W. A tutorial on graph-based SLAM. *IEEE Intell. Transp. Syst. Mag.* **2010**, *2*, 31–43. [CrossRef]
9. Strasdat, H.; Davison, A.J.; Montiel, J.M.; Konolige, K. Double window optimisation for constant time visual SLAM. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2352–2359.
10. Harik, E.H.C.; Korsaeth, A. Combining hector slam and artificial potential field for autonomous navigation inside a greenhouse. *Robotics* **2018**, *7*, 22. [CrossRef]
11. Xia, X.; Hashemi, E.; Xiong, L.; Khajepour, A. Autonomous Vehicle Kinematics and Dynamics Synthesis for Sideslip Angle Estimation Based on Consensus Kalman Filter. *IEEE Trans. Control Syst. Technol.* **2022**, *31*, 179–192. [CrossRef]
12. Xia, X.; Xiong, L.; Huang, Y.; Lu, Y.; Gao, L.; Xu, N.; Yu, Z. Estimation on IMU yaw misalignment by fusing information of automotive onboard sensors. *Mech. Syst. Signal Process.* **2022**, *162*, 107993. [CrossRef]
13. Liu, W.; Quijano, K.; Crawford, M.M. YOLOv5-Tassel: Detecting tassels in RGB UAV imagery with improved YOLOv5 based on transfer learning. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2022**, *15*, 8085–8094. [CrossRef]
14. Xia, X.; Meng, Z.; Han, X.; Li, H.; Tsukiji, T.; Xu, R.; Zhang, Z.; Ma, J. Automated Driving Systems Data Acquisition and Processing Platform. *arXiv* **2022**, arXiv:2211.13425.
15. Noto, M.; Sato, H. A method for the shortest path search by extended Dijkstra algorithm. In Proceedings of the Smc 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics.'Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions, Nashville, TN, USA, 8–11 October 2000; Volume 3, pp. 2316–2320.
16. Seet, B.C.; Liu, G.; Lee, B.S.; Foh, C.H.; Wong, K.J.; Lee, K.K. A-STAR: A mobile ad hoc routing strategy for metropolis vehicular communications. In Proceedings of the Networking 2004: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Third International IFIP-TC6 Networking Conference, Athens, Greece, 9–14 May 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 989–999.
17. Ogren, P.; Leonard, N.E. A convergent dynamic window approach to obstacle avoidance. *IEEE Trans. Robot.* **2005**, *21*, 188–195. [CrossRef]
18. Chang, L.; Shan, L.; Jiang, C.; Dai, Y. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. *Auton. Robot.* **2021**, *45*, 51–76. [CrossRef]
19. Rösmann, C.; Hoffmann, F.; Bertram, T. Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control. In Proceedings of the 2015 European Control Conference (ECC), Linz, Austria, 15–17 July 2015; pp. 3352–3357.
20. Ragot, N.; Khemmar, R.; Pokala, A.; Rossi, R.; Ertaud, J.Y. Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map. In Proceedings of the 2019 Eighth International Conference on Emerging Security Technologies (EST), Colchester, UK, 22–24 July 2019; pp. 1–6.
21. Yang, J.; Wang, C.; Luo, W.; Zhang, Y.; Chang, B.; Wu, M. Research on point cloud registering method of tunneling roadway based on 3D NDT-ICP algorithm. *Sensors* **2021**, *21*, 4448. [CrossRef] [PubMed]
22. Xue, G.; Wei, J.; Li, R.; Cheng, J. LeGO-LOAM-SC: An Improved Simultaneous Localization and Mapping Method Fusing LeGO-LOAM and Scan Context for Underground Coalmine. *Sensors* **2022**, *22*, 520. [CrossRef]
23. Zheng, X.; Gan, H.; Liu, X.; Lin, W.; Tang, P. 3D Point Cloud Mapping Based on Intensity Feature. In *Artificial Intelligence in China*; Springer: Singapore, 2022; pp. 514–521.
24. Zhang, G.; Yang, C.; Wang, W.; Xiang, C.; Li, Y. A Lightweight LiDAR SLAM in Indoor-Outdoor Switch Environments. In Proceedings of the 2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI), Nanjing, China, 28–30 October 2022; pp. 1–6.
25. Karal Puthanpura, J. Pose Graph Optimization for Large Scale Visual Inertial SLAM. Master's Thesis, Aalto University, Espoo, Finland, 2022.
26. Li, H.; Dong, Y.; Liu, Y.; Ai, J. Design and Implementation of UAVs for Bird's Nest Inspection on Transmission Lines Based on Deep Learning. *Drones* **2022**, *6*, 252. [CrossRef]
27. Moshayedi, A.J.; Roy, A.S.; Sambo, S.K.; Zhong, Y.; Liao, L. Review on: The service robot mathematical model. *EAI Endorsed Trans. AI Robot.* **2022**, *1*, 8. [CrossRef]
28. Zhang, B.; Li, G.; Zheng, Q.; Bai, X.; Ding, Y.; Khan, A. Path planning for wheeled mobile robot in partially known uneven terrain. *Sensors* **2022**, *22*, 5217. [CrossRef]
29. Vagale, A.; Oucheikh, R.; Bye, R.T.; Osen, O.L.; Fossen, T.I. Path planning and collision avoidance for autonomous surface vehicles I: A review. *J. Mar. Sci. Technol.* **2021**, *26*, 1292–1306. [CrossRef]
30. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P.; Forestiero, A. A consolidated review of path planning and optimization techniques: Technical perspectives and future directions. *Electronics* **2021**, *10*, 2250. [CrossRef]
31. Wu, J.; Ma, X.; Peng, T.; Wang, H. An improved timed elastic band (TEB) algorithm of autonomous ground vehicle (AGV) in complex environment. *Sensors* **2021**, *21*, 8312. [CrossRef] [PubMed]

32. Cheon, H.; Kim, T.; Kim, B.K.; Moon, J.; Kim, H. Online Waypoint Path Refinement for Mobile Robots using Spatial Definition and Classification based on Collision Probability. *IEEE Trans. Ind. Electron.* **2022**, *70*, 7004–7013. [CrossRef]
33. Gao, L.; Xiong, L.; Xia, X.; Lu, Y.; Yu, Z.; Khajepour, A. Improved vehicle localization using on-board sensors and vehicle lateral velocity. *IEEE Sens. J.* **2022**, *22*, 6818–6831. [CrossRef]