Development Of An AMR Applying Cartographer Combined With Visual Odometry For Navigation

Chang-Ching Huang, Chien-Hung Huang, and Jin-Siang Shaw*

Institute of Mechatronic Engineering, National Taipei University of Technology, Taipei, Taiwan

** Corresponding author. E-mail: jshaw@ntut.edu.tw*

Received: Oct. 26, 2023; Accepted: Feb. 01, 2024

This study aims to develop an autonomous mobile robot (AMR) for accurate positioning in a robot operation system (ROS) architecture. The research method uses the information obtained by the laser rangefinder to cooperate with the Cartographer SLAM algorithm to build a map and position the robot in the environment. In addition, the RGBD camera is used for visual odometry to assist Cartographer SLAM in improving positioning accuracy. For path planning, the global planning method uses the A* algorithm, and through the time elastic band (TEB) local path planning algorithm. After repeating the accuracy test 15 times, the average absolute error of the localization was 4.48 cm, which is more accurate than the result obtained using Cartographer SLAM alone at 9.94 cm. Hence, the positioning accuracy was improved by 55%.

Keywords: ROS, Navigation, Cartographer, TEB, ORB-SLAM2

http://dx.doi.org/10.6180/jase.202501_28(1).0004

1. Introduction

Autonomous mobile robots (AMRs) need to build an environment map to navigate autonomously. Sensors scan and match data for localization and map creation. Precise mapping relies on accurate localization. Real-time mapping and localization (SLAM) [1, 2], proposed by Durrant-Whyte in 2006 , is pivotal for mobile robots. SLAM enables robots to calculate their position, trajectory, and map in an unknown environment. SLAM has LiDAR and visual SLAM, based on different sensors. 2D LiDAR commonly uses Gmapping [3] and Cartographer SLAM [4]. Visual SLAM uses various devices, like MonoSLAM [5] and ORB-SLAM2 [6]. Good mapping and localization are followed by path planning for navigation and dynamic obstacle avoidance. Global planning uses A* [7], and local planning includes DWA [8] and TEB [9] algorithms. Some studies solely use Cartographer SLAM, but it may lack stability and accuracy, especially in featureless corridors [10]. Using ORB-SLAM2 alongside 2D LiDAR SLAM improves accuracy [11].

This study aims to enhance Cartographer's localization with visual odometry, choosing ORB-SLAM2 to improve initial pose estimation for scan matching using Ceres-based nonlinear least squares [12].

2. System description

2.1. Hardware

Fig. 1 shows the architecture of the system hardware. AMR uses many sensors. Hokuyo and SICK are the lasers used for obstacle detection and building environment maps, respectively. The Kinect v2, an RGBD camera, provides image and depth information for visual odometry. The IMU measures the rotational movement in the Yaw direction of the AMR. The motor was a 24 V DC brushless motor, and the pulses per revolution (PPR) value of the encoder was 500. One Arduino Mega 2560 board was used to control the speed of the motor and two Arduino Mega 2560 board read the actual speed value of the motor measured by the encoder.

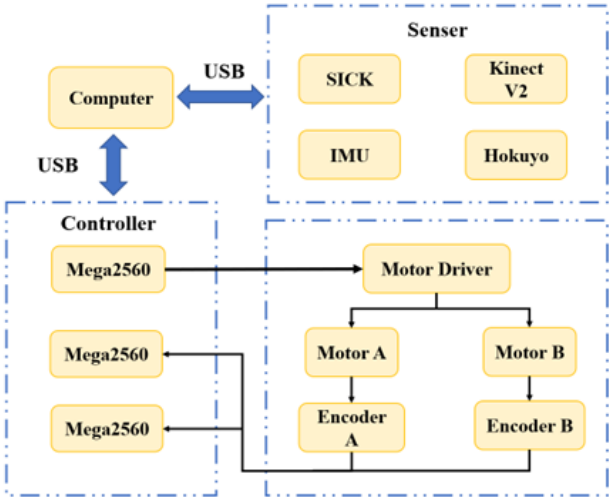

Fig. 1. Hardware architecture.

2.2. Software Methodology

2.2.1. *Cartographer*

Cartographer SLAM is a real-time algorithm that creates a 2D grid map at a 5 cm resolution. It incorporates each laser scan frame into the nearest submap for pose estimation, and employs scan matching. However, this approach accumulates errors in world coordinates. Instead of traditional particle filters, Cartographer relies on regular attitude optimization to rectify these errors and achieve accurate mapping and localization on common hardware. Completed submaps cannot accept additional scan data, only participating in closed-loop optimization, including all submaps and current frame scans. The system comprises two parts: local SLAM (front-end) for submap creation and global SLAM for closed-loop detection and optimization to correct errors within or between submaps.

1. Local SLAM

   Each continuous scan of each frame is matched to a submap(M) using nonlinear optimization to align the current scan with the submap. This step is known as scan matching. Assuming that the scanned pose of the current frame is $\xi = (\xi_x, \xi_y, \xi_\theta) \circ$

   ✓ Scan

   For each submap, the coordinate relationship between each scan frame and the submap was repeatedly iterated through each frame scan, and the scan information was then inserted into the submap. In the submap, ξ is the pose of the current scan frame, according to Eq. (1), T_ξ represents the transform of the scan point of the laser scan into the submap coordinate. It converts the scan points in the scan information from the scan coordinate system into a submap coordinate system.

$$T_{\xi p} = \begin{pmatrix} \cos\xi_\theta & -\sin\xi_\theta \\ \sin\xi_\theta & \cos\xi_\theta \end{pmatrix} p + \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \quad (1)$$

   ✓ Submap

   When inserting a scan into a probability grid map, we compute hit and miss grid points. For each hit set, we insert the nearest grid point (gray, marked with an X) after matching. Lines connecting scan origin to matched grid points add gray grid points they pass through to the miss collection. If an unobserved mesh is part of the hit or miss set, it's assigned hit probabilities (p_{hit}) or miss probabilities (p_{miss}). When observing the x-grid, submap grid points update using Eqs. (2) and (3). Odds are expressed as possibilities, and clamp is used to limit values within the $p_{\min}$ to $p_{\max}$ interval (Fig. 2).

$$odds(p) = \frac{p}{1-p} \quad (2)$$

$$M_{\text{new}\,(x)} = \text{clamp}\left(\text{odds}\left(M_{\text{old}}\,(x)\right) \cdot \text{odds}\left(p_{\text{hit}}\right)\right) \quad (3)$$

2. Global SLAM

   In local map construction, the current frame's scan is matched to the nearest submap, but in large scenes, multiple submaps accumulate errors. Global SLAM uses sparse pose adjustment to optimize scan and submap poses. Pose data from each frame scan inserted into the submap is saved for closed-loop optimization. When a submap is no longer updated, its data is considered for closed-loop detection. The attitude matcher continuously runs in the background, sending better closed-loop candidates for optimization.

   ✓ Optimization problem

   Closed-loop optimization is similar to scan matching but introduces a residual error for data reference. This formula can be structured as a least-squares nonlinear problem, typically solved using Ceres-based methods. Eq. (4) represents this, where Ξ^m is the pose in the submap map $\left(\{\xi_i^m\}_{i=1,\ldots,m}\right)$, and Ξ^S is the submap's

pose $\left(\left\{\xi_j^S\right\}_{i=1,\ldots,n}\right)$. Both are optimized with specific constraints. Constraints involve relative pose ξ_{ij} and covariance Σ_{ij} between submaps and scans. The residual constraint is $\xi_{ij}s$, which describes the scan's pose in the submap via frontend scan matching. ρ is the loss function, used to mitigate outliers' impact in case of incorrect constraints in scan matching.

$$\operatorname{argmin}\left(\Xi^m, \Xi^S\right) = \frac{1}{2}\sum_{ij}\rho\left(E^2\left(\xi_i^m, \xi_j^S; \Sigma_{ij}, \xi_{ij}\right)\right) \tag{4}$$

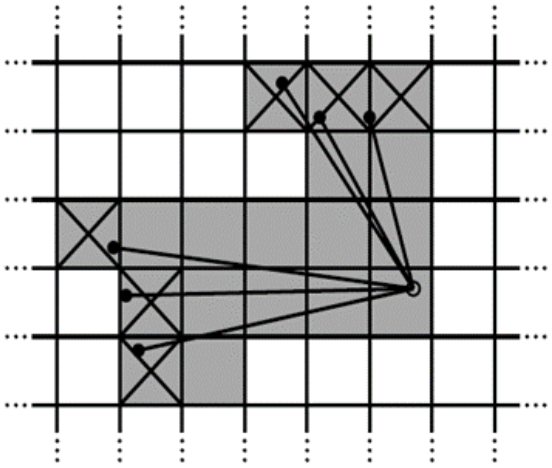

Fig. 2. Grid map of a submap.

2.2.2. *ORB-SLAM2*

1. Tracking
   Tracking is a core ORB-SLAM2 process that calculates the current frame's image and establishes the coordinate relationship with the local map for camera position optimization. It employs the Oriented FAST and Rotated BRIEF (ORB) feature extraction algorithm on FAST points across eight scale levels. When tracking the previous frame, a motion model predicts the camera's position and searches for matching map points to determine location. If insufficient matching points are found, the motion model becomes invalid. In such cases, the tracking reference keyframe model and the Bag of Words (BoW) vector of the current frame are used to select keyframes and map points as features. If both models fail, global relocalization is initiated.

2. Local Mapping
   In the tracking phase, when a new keyframe K_i is obtained, it's added to the covisibility graph. Keyframes sharing map points with K_i are connected through edges. The keyframe in the spanning tree of K_i that shares the most common points is updated, its BoW description calculated, and new map points generated through triangulation. Local boundary adjustment (local BA) optimizes the
   current keyframe K_i. The optimization process, shown in Fig. 3, involves keyframes Pos 3 (K_i), Pos 2 (K_c connected to K_i in the covisibility graph), and Pos1 (observable but not directly connected to K_i). Map points $X1$ and $X2$ are visible to both keyframes. To simplify the structure, marked points are eliminated during or after optimization to streamline the process.

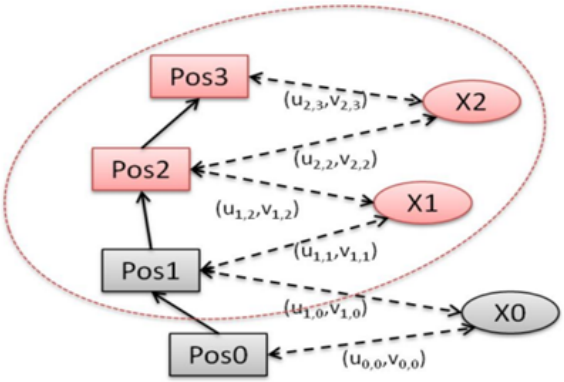

Fig. 3. Local BA process

3. Looping Closing
   Closed-loop detection begins with comparing the last keyframe K_i. in the local map to the previous keyframes. The keyframe with the lowest similarity score $S_{\min}$ is retained, while those with scores below $S_{\min}$ are discarded. Keyframes linked to K_i are deleted, and DboW2 statistics produce candidate loops. The process has two steps: first, merging duplicate map points and connecting loops in the covisibility graph; second, adjusting the current frame's position, along with associated keyframes, based on similarity transformation. Map point locations are optimized using the essential graph, dispersing closed-loop errors for effective detection.

4. Full Bundle Adjustment
   After closed-loop detection, the optimized covisibility graph is sent for global bundle adjustment. The optimization process runs independently to maintain continuous mapping and closed-loop detection. If a new closed-loop is detected during optimization, it's

paused and resumes after closure. Once global bundle adjustment is done, updated keyframes are sent to non-updated keyframes using a spanning tree, bolstering system robustness.

3. Result and discussion

Experiments fall into two categories: using Cartographer SLAM localization alone and using ORBSLAM2 as odometry alongside Cartographer SLAM. Both were repeated 15 times for precision assessment, recording X-axis error (Δx), Y-axis error (Δy), straight-line distance error (Δl), and Yaw angle $(\Delta\theta)$ error compared to the target point. Fig. 4 displays the experimental environment map. The experiments were
conducted in the narrow corridors of the school's basement. Both experimental groups start from point A and proceed towards point B. The total distance is 35 m. During the navigation process, there is an encounter with a parked stationary vehicle, and personnel may be present, moving within the area.

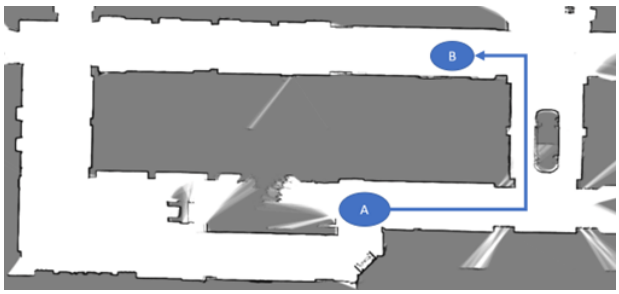

Fig. 4. Map of experiment environment

After 15 tests, using Cartographer SLAM alone, the error distribution is displayed in Fig. 5, with data listed in Table 1. The average absolute straight-line distance error across repeated tests was 9.93 cm, and the average Yaw angle error was 2.08°. We can observe from Fig. 5 that the repeatability of the final positions varies largely, and the distribution is even scattered.

When using the ORB-SLAM2-assisted Cartographer SLAM method, the error distribution is shown in Fig. 6, and data is presented in Table 2. The average absolute straight-line distance error for repeated tests was 4.48 cm, and the average Yaw angle error was 2.11°. We can observe from Fig. 6 that the repeatability of the final positions is better and the distribution is even more concentrated.

Results show that combining ORB-SLAM2 with Cartographer SLAM results in smaller, more evenly distributed errors, improving localization stability. In featureless corridors, Cartographer SLAM alone struggles due to similar

Table 1. Results of accuracy test repeated 15 times using cartographer only.

Test	Δx(cm)	Δy(cm)	ΔL(cm)	Δθ (°)
1	6.50	0.50	6.52	2.00
2	23.10	1.50	23.15	3.00
3	1.20	2.50	2.77	1.10
4	7.10	1.10	7.18	2.00
5	-6.50	1.70	6.72	1.50
6	10.00	0.50	10.01	1.60
7	11.00	-1.50	11.10	1.10
8	-15.00	0.70	15.02	1.40
9	21.10	1.80	21.18	1.00
10	3.50	1.60	3.85	2.50
11	-4.60	-1.50	4.84	2.50
12	8.30	1.10	8.37	2.50
13	-10.00	0.80	10.03	3.50
14	9.50	0.60	9.52	3.00
15	-10.50	1.10	10.56	2.50
MAE	9.86	1.23	9.94	2.08

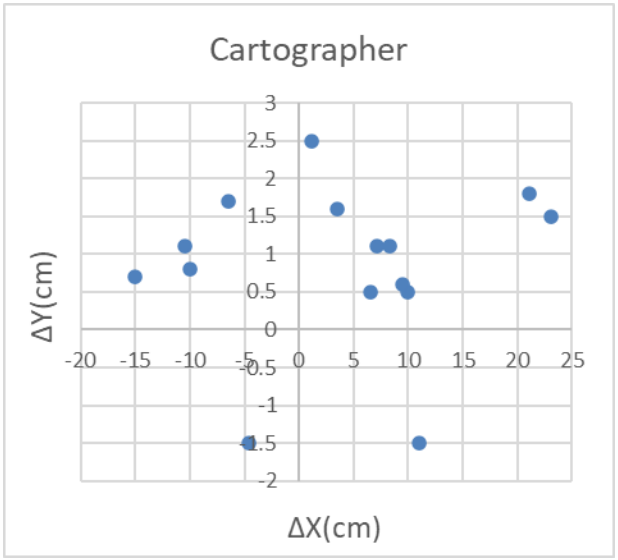

Fig. 5. Error distribution graph of using Cartographer only

environment at docking points, resulting in poor localization. Adding ORBSLAM2 as an odometry source improves performance by offering a stable initial position and matching value.

4. Conclusions

This study uses ORB-SAM2 to assist Cartographers in building maps and localization, and uses A* and TEB path planning algorithms to achieve the functions of autonomous navigation and obstacle avoidance of the AMR. Using visual SLAM to assist 2D LiDAR SLAM in building maps and positioning solves the problem of low localization accuracy of general 2D SLAM in a corridor environ-

Table 2. Results of 15 times accuracy test repeated 15 times using cartographer with ORB-SLAM2.

Test	Δx(cm)	Δy(cm)	ΔL(cm)	Δθ (°)
1	4.10	1.50	4.37	1.50
2	6.10	0.50	6.12	2.00
3	4.50	1.30	4.68	2.50
4	2.00	0.70	2.12	2.50
5	4.60	0.30	4.61	1.50
6	6.20	1.30	6.33	2.50
7	-3.80	1.20	3.98	2.00
8	5.20	-1.10	5.32	1.80
9	3.50	0.40	3.52	2.60
10	4.10	0.60	4.14	1.60
11	-4.50	0.50	4.53	2.10
12	-5.10	0.20	5.10	1.00
13	4.00	-0.70	4.06	2.00
14	3.90	1.10	4.06	2.60
15	4.20	0.80	4.28	3.50
MAE	4.39	0.81	4.48	2.11

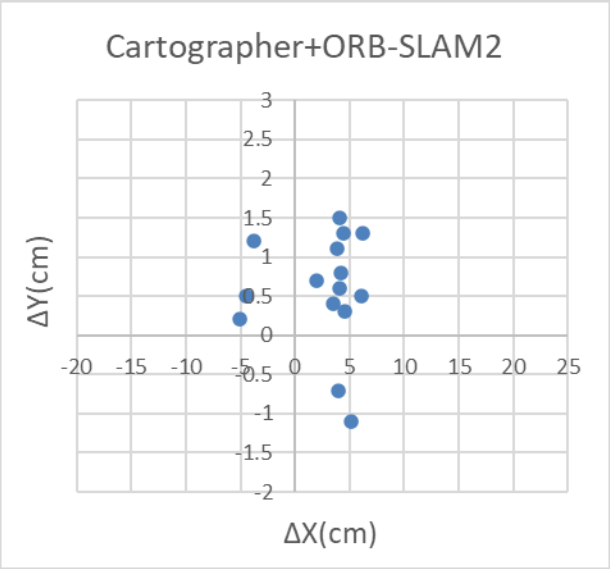

Fig. 6. Error distribution graph of using Cartographer with ORB-SLAM2

ment. However, limitations of the proposed approach may include: first, sufficient ambient lighting in the environment is necessary, as insufficient lighting may impact the performance of visual SLAM; second, during navigation, the AMR should not move too quickly, as excessive speed may lead to ORB-SLAM2 being unable to provide timely assistance.

Declaration of conflicting interests

The authors declare no potential conflicts of interest regarding the research, authorship, or publication of this study.

Funding

This study was supported by the National Science and Technology Council, Taiwan (NSTC 112-2218E-027-006).

References

[1] H. Durrant-Whyte and T. Bailey, (2006) *"Simultaneous localization and mapping: part I"* **IEEE robotics & automation magazine *13***(2): 99–110. DOI: 10.1109/MRA.2006.1638022.

[2] T. Bailey and H. Durrant-Whyte, (2006) *"Simultaneous localization and mapping (SLAM): part II"* **IEEE Robotics & Automation Magazine *13***(3): 108–117. DOI: 10.1109/MRA.2006.1678144.

[3] G. Grisetti, C. Stachniss, and W. Burgard, (2007) *"Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters"* **IEEE Transactions on Robotics *23***(1): 34–46. DOI: 10.1109/TRO.2006.889486.

[4] W. Hess, D. Kohler, H. Rapp, and D. Andor. "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, 1271–1278. DOI: 10.1109/ICRA.2016.7487258.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, (2007) *"MonoSLAM: Real-Time Single Camera SLAM"* **IEEE Transactions on Pattern Analysis and Machine Intelligence *29***(6): 1052–1067. DOI: 10.1109/TPAMI.2007.1049.

[6] R. Mur-Artal and J. D. Tardós, (2017) *"ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras"* **IEEE Transactions on Robotics *33***(5): 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[7] A. Bostel and V. Sagar, (1996) *"Dynamic contral systems for AGVs"* **Computing & Control Engineering Journal *7***(4): 169–176.

[8] D. Fox, W. Burgard, and S. Thrun, (1997) *"The dynamic window approach to collision avoidance"* **IEEE Robotics & Automation Magazine *4***(1): 23–33. DOI: 10.1109/100.580977.

[9] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram. "Efficient trajectory optimization using a sparse model". In: *2013 European Conference on Mobile Robots*. 2013, 138–143. DOI: 10.1109/ECMR.2013.6698833.

[**10**] L. Zhou, C. Zhu, and X. Su. "SLAM algorithm and Navigation for Indoor Mobile Robot Based on ROS". In: *2022 IEEE 2nd International Conference on Software Engineering and Artificial Intelligence (SEAI)*. 2022, 230–236. DOI: 10.1109/SEAI55746.2022.9832313.

[**11**] P.-L. Wu, J.-J. Li, and J.-S. Shaw, (2022) "*Development of an Omnidirectional AGV by Applying ORB-SLAM for Navigation Under ROS Framework*" **Journal of Automation, Mobile Robotics and Intelligent Systems** ***16***(1): 14–20.

[**12**] S. Agarwal, K. Mierle, et al. *Ceres solver*. 2012. URL: http://ceres-solver.org.