# Comparing Two SLAM Solutions to Improve the Perception Accuracy of Autonomous Vehicles

**Yu Ji[1] and Jian Qu[2*]**
[1,2]Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi, Thailand
E-mail: 6672100059@stu.pim.ac.th, jianqu@pim.ac.th[*]

Received: November 8, 2024 / Revised: March 24, 2025 / Accepted: May 28, 2025

***Abstract*— Sensing the environment through sensors has always been vital in realizing autonomous driving tasks. Environmental perception performance directly affects autonomous driving performance. Autonomous driving vehicles perceive the surrounding environment based on sensor data, and environmental perception is usually achieved through mapping and positioning. Two traditional perception methods exist: (1) Relying on existing map information and collecting data to calculate the location problem. (2) Relying on known positioning coordinates to infer map information. The real-time performance of these two methods is abysmal and cannot meet the needs of continuous autonomous driving. This paper analyzes the accuracy of two SLAM solutions, Gmapping and Cartographer, through a comparison experiment in which a car perceives maps and obstacles in a real environment, and concludes that cumulative error is a key factor affecting accuracy. This has drawn attention to loop detection.**

***Index Terms*—Autonomous Driving, Obstacle Avoidance, Environmental Awareness, Map Construction**

## I. Introduction

Autopilot can be realized in many ways, but many problems affect its robustness when moving from virtual to real-world deployment. Existing autopilot systems are generally divided into a perception layer, a decision-making layer, and an executive layer. The perception layer executes environment-aware commands and is the core technology enabling interaction between the vehicle and the external environment. At the perceptual level, the algorithm may classify a piece of transparent glass as a non-obstacle, but it still exists. If self-driving cars cannot detect problematic obstacles, then self-driving cannot be achieved. Therefore, improving self-driving perception has long been a focus of research. Once perception is solved, obstacle avoidance becomes much easier. In addition, real-time perception is also important. Self-driving cars do not need to collect road data before driving. SLAM scheme can improve self-driving vehicles' perception and real-time performance [1]. SLAM (Simultaneous Localization and Mapping), also known as CML (Concurrent Mapping and Localization), involves real-time positioning and map construction. It solves positioning, navigation, obstacle avoidance, and related problems in unfamiliar environments. The SLAM system framework is divided into five modules: sensor data, visual odometer, back-end, mapping, and loop detection [2]. Based on sensors, SLAM falls into three categories: visual SLAM, lidar SLAM, and Multi-sensor fusion SLAM. This paper aims to deploy a self-driving car in a real scene to test its environmental awareness and verify the SLAM scheme's reliability.

## II. Research Objective

In specific environments, self-driving cars can rely on a single camera to perceive the environment and piece together different areas based on the real-world layout through boundary features. However, in harsh environments, a single sensor is powerless. Moreover, multimodal perception algorithms have a threshold. If objects in the real environment fall below the threshold during calculation, the self-driving car may make mistakes. We aim to improve self-driving perception technology by comparing it with current standard SLAM methods.

## III. Literature Review

A debate has long existed about the strengths of visual and non-visual SLAM schemes. Generally, vision schemes have advantages and can provide rich environmental texture information. However, vision schemes require significant computation and are easily affected by external light [3]-[5], reducing the robustness of self-driving cars. Radar offers more accurate range, stable operation in strong light, and easier information processing. However, the radar scheme loses detail.

### *A. Vision SLAM*

Vision SLAM collects data using monocular [4]-[6], binocular, or stereoscopic cameras, or other vision sensors. A visual SLAM system has five parts: Data reading, front end, back end, loop detection, and drawing. The system's workflow usually involves the sensor collecting image data and delivering it to the front end, which extracts image features to build the local map and independently locate the vehicle. Then, the back end optimizes the front-end information. Loop detection combines maps generated by different parts of the same scene, using known locations, coordinates, and other information to complete a large map.

### *B. Laser SLAM*

Laser SLAM is a SLAM scheme in which self-driving vehicles estimate their position and pose from point cloud information fed back by lidar and build maps of the surrounding environment [2], [7]. Laser SLAM falls into two categories: filter-based and optimization-based. Representative examples include the Gmapping and Cartographer algorithms.

*1) Gmapping*

Gmapping, proposed by Grisetti G and others [8], [9], is based on the particle filter method to estimate the posture of the self-driving vehicle, propagates each particle with the kinematic model, calculates the propagated particles with the weight of the observation model, and constructs the map according to the measured pose of the self-driving vehicle. It can achieve better mapping in small environments. When using particle filters to estimate the pose of a self-driving car, each particle stores a specific time track of the self-driving car and a map of the environment at a particular time. This inevitably leads to particle resampling. The longer it takes to build the map, the more particles are resampled, and even if all particles are copied from one particle, the particles' diversity is lost entirely. To reduce memory use from resampling, Gmapping reduces the number of resamplings and uses a threshold to measure the difference between the predicted and actual distributions. [9], [10]. When the difference is minimal, it performs no resampling. When the difference is significant, it resamples. However, mapping treats all particles equally when the odometer model propagates, leading to particle degradation, such as replacing high-quality particles with low-quality particles. Therefore, odometer information is essential for the Gmapping solution, and odometer accuracy determines the quality of the map constructed.

Gmapping uses particle filtering to estimate the robot's position and posture. Each particle represents a possible robot trajectory, and a weight represents its credibility. The robot's state (position and posture) is (1-1), where x and y are positions and θ is the direction.

The particle filter formula is (1-2), where $x_{1:t}^{i}$ is the state trajectory of the i-th particle. $\omega_t^i$ is the weight of the i-th particle. $\delta(\cdot)$ is the Dirac delta function, indicating that the state of the particle is discrete.

The particle weight is updated using the observation data zt, as shown in formulas (1-3). Among them, $p(z_t|x_t^i, m_{t-1}^i)$ is the observation model, representing the probability of observing $z_t$ under particle I and map $m_{t-1}^i$.

To avoid particle degradation (some particles have too large weights), Gmapping periodically resamples, discards low-weight particles, and copies high-weight particles. Each particle independently maintains a map, and the map update is based on the current observation data, as shown in the formula (1-4). Maps are usually represented using an Occupancy Grid Map, where the probability of each grid is updated using observed data.

$$x_t = (x, y, \theta) \tag{1-1}$$

$$p(x_{1:t}|z_{1:t}, u_{1:t}) = \sum_{i=1}^{N} \cdot \omega_t^i \delta(x_{1:t} - x_{1:t}^i) \tag{1-2}$$

$$\omega_t^i \propto \omega_{1:t}^i \cdot p\left(z_t \middle| x_t^i, m_{t-1}^i\right) \tag{1-3}$$

$$p(m|x_{1:t}, z_{1:t}) = \Pi_{i=1}^{N} \cdot p\left(m \middle| x_{1:t}^i, z_{1:t}\right) \tag{1-4}$$

*2) Cartographer*

The Cartographer algorithm is Google's open-source solution. Its core is creating local sub-images fused with multi-sensor data and a scan-matching strategy for closed-loop detection [11].

In this scheme, the front-end scan-matching algorithm combines Correlative Scan Matching (CSM) and gradient optimization. After generating a sub-map, it performs local closed-loop detection; once all sub-maps are constructed, it performs global closed-loop detection using a branch-and-bound algorithm and a pre-calculated grid to speed it up. However, Cartographer does not verify results during closed-loop detection, which can lead to errors in closed-loop maps when the robot perceives similar boundaries.

Cartographer represents the robot's pose and observation data as nodes and edges in a graph and solves the SLAM problem by optimizing the graph structure.

Each node represents the robot's pose at a particular moment $x_t = (x, y, \theta)$, where x,y are positions and θ is the direction. Edges represent constraints between poses, such as relative poses from odometers or loop-closure detection. Cartographer optimizes the graph structure by minimizing the error function, as shown in formula (2-1), where $e_{ij}(x)$ is the error function representing the difference between the actual observation and the predicted value

between node i and node j. $\Omega_{ij}$ is the information matrix, which means the confidence of the constraint.

For pose constraints, the formula of the error function is as follows (2-2), where $\ominus$ represents the relative transformation of pose and $z_{ij}$ is the observed relative pose.

$$\hat{x} = \arg\min_{x} \sum_{i,j} e_{ij}(x)^{T} \Omega_{ij} e_{ij}(x) \quad (2\text{-}1)$$

$$e_{ij}(x) = x_i \ominus x_j - z_{ij} \quad (2\text{-}2)$$

A cartographer performs loop detection by matching the current laser scan data with the historical map data and finding possible loops. The probability model formula for loop detection can be expressed as (2-3), where $\pi_k$ is the mixing weight and $N(z_t|\mu_k, \sum_k)$ is the Gaussian distribution, which indicates the matching degree between the observed data $z_t$ and the map m.

$$p(z_t|x_t, m) = \sum_{k=1}^{K} \pi_k N(z_t|\mu_k, \sum_k) \quad (2\text{-}3)$$

Cartographer uses sub-maps to manage large-scale environments. Each submap is a local map constructed by local optimization. The global map is composed of multiple sub-maps aligned by global optimization.

### C. *Multi-sensor Fusion SLAM*

When comparing perception capabilities without considering factors such as price and operating speed, multi-sensor fusion SLAM technology outperforms single-sensor SLAM technology (including visual SLAM and radar SLAM). [12]-[15]. Multi-sensor fusion SLAM technology can provide information redundancy. In simple terms, when an object falls below a particular sensor's recognition threshold, another sensor fills the gap, making map building more comprehensive and accurate.

#### *1) RTAB-Map*

Given the urgent need for multi-sensor technology in the scientific research community, Michaud et al. proposed RTAB-Map. Real-Time Appearance-Based mapping (RTAB-Map) is an appearance-based closed-loop detection method that integrates visual and lidar SLAM solutions and supports real-time positioning and navigation of most current sensors in unknown environments.

SLAM emphasizes accurate, real-time mapping and positioning. When computer performance is limited, these two requirements often conflict. We must carefully balance them in the program to achieve the best practical results.

SLAM methods must identify when to revisit past locations (loop detection) to correct the map. Dynamic environments, lighting changes, geometric changes, and repeated environments can lead to incorrect positioning or positioning failures. Therefore, the method should be robust to the above problems.

Currently, robot navigation methods are based on grid maps. For static large scene maps, effective map management involves map sessions and call and memory management during positioning to reduce map maintenance time.

When the robot does not know its relative position to the previously created map or loses its position, it cannot plan a path to a previously visited location. To avoid restarting mapping from scratch or positioning itself in a previously built map, multi-session mapping allows the system to initialize a new map at startup and then merge the two maps when it encounters a previously visited location.

## IV. Methodology

This section introduces the laser SLAM algorithm Cartographer and completes mapping and positioning through experiments [16]-[20]. The experimental steps include: 1) constructing the model car; 2) selecting the real environment; 3) collecting data; and 4) evaluating. The experiment aims to build the model car in a real environment, collect feedback data to generate map information and model vehicle location information, and ultimately enable environmental perception.

### A. *ROS2 System Model Car*

The model car is equipped with the ROS2 operating system, an Astra Pro depth camera, a lidar, an IMU, a Jetson Orin Nano development board, a ROS car expansion version, a USB UHB expansion version, an Ackerman backplane, and a touch screen.



Fig .1. ROS2 autonomous driving car

TABLE I
AUTONOMOUS DRIVING CAR CONFIGURATION

| Classification | Detailed |
|---|---|
| Operating System | ROS2 |
| | Astra Pro Depth Camera |
| Sensor | Lida |
| | IMU |
| GPU | Jeston Orin Nano 8GB |
| Extended board | ROS Car Expansion Board |
| | USB Expansion Board |
| Base Plate | Ackerman baseboard |
| Operation board | Touch Screen |

### B. Data Collection

To intuitively observe the effect of mapping [16], [19], we chose a closed rectangular space, arranged four obstacles in the space, and divided them into one easy-to-identify obstacle and three difficult-to-identify obstacles based on recognition difficulty. Then, we set up two scenes in the same rectangular space. In Scene 1, the self-driving car is in the middle of the rectangular space, with an obstacle in each of the four directions. In Scene 2, place obstacles only in front of and to the left of the self-driving car. Use this to observe SLAM results and calibrate the real environment and obstacles in the SLAM map [21], [22]. Rely on vehicle radar to measure the barriers and feed them to the map. For this reason, I used another algorithm, Gmapping, to compare its effects with Cartographer. Both Gmapping and Cartographer algorithms rely on radar sensors.

*1) Experimental Purpose*

To test and compare the accuracy of autonomous vehicles in perceiving the environment and detecting closed loops on maps using Cartographer and GMapping algorithms, we designed the following experiment: placing the autonomous vehicle in an unfamiliar environment and placing obstacles in the four directions of the scene, allowing it to restore the scene and obstacles as much as possible on the 2D grid map. Based on the real environment, the advantages and disadvantages of the two algorithms are analyzed, and a better perception algorithm is proposed for the future.

*2) Obstacle Classification*

LiDAR does not perform well in identifying transparent objects and may fail to identify them. However, it performs well in identifying non-transparent objects. Therefore, we classify transparent obstacles as challenging to locate and non-transparent obstacles as easy to identify.

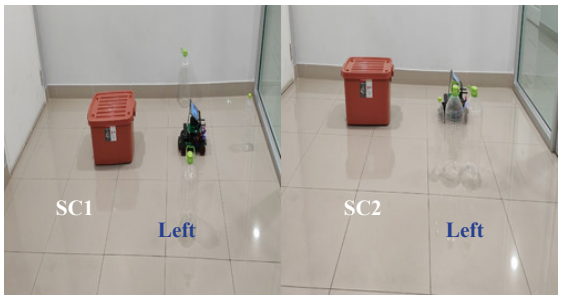


Fig. 2. Scene layout

### C. Evaluation Method

While keeping the experimental environment and equipment unchanged, we tested the Gmapping and Cartographer algorithms once each, recorded their results for perceiving maps and obstacles in the experimental environment, and plotted them in a table. If the surrounding environment is successfully perceived to form a raster map, we write a checkmark in the environment column; if a simple obstacle is successfully identified, we write a checkmark in the simple column; if N difficult obstacles are successfully identified, we write the corresponding N in the problematic column [23]-[24].

TABLE II
ALGORITHM RELIABILITY

| Algo | Sce | Env | Easy | Hard |
|---|---|---|---|---|
| Cartographer | 1 | √ | √ | 2 |
| | 2 | √ | √ | √ |
| Gmapping | 1 | √ | √ | 1 |
| | 2 | √ | √ | √ |

## V. RESULT

Through the map comparison, we find that both Cartographer and Gmapping can restore the actual scene and identify obstacles that are easy to locate. However, Cartographer performs better at identifying three difficult-to-find barriers. In scene 1, Cartographer identified the easily identifiable obstacles directly in front and the challenging obstacles on the left and rear. In scenario 2, the Cartographer can identify three stacked barriers that are difficult to locate. When using the Gmapping algorithm, Gmapping recognizes the easy-to-identify obstacles in front of scene 1 but does not identify the problematic obstacles on the left, right, and rear. In scene 2, the Cartographer can identify three stacks of challenging obstacles.

When Gmapping performs loop Detection, computational complexity increases with the number of particles, particle degradation occurs, and accuracy

decreases after long-term operation. Cartographer uses scan matching for loop detection. Although the calculation is complex, the accuracy is more stable and does not change over time.

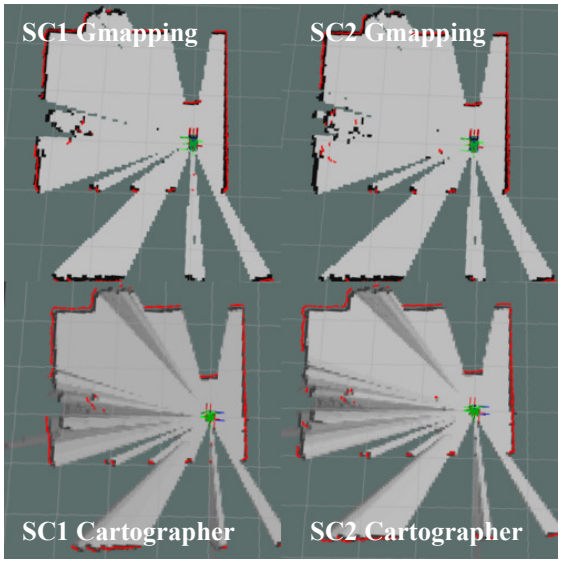


Fig. 3. Mapping

, vol. 22, no. 6, Art. no. 013, 2022, https://doi.

## VI. Conclusion

This article introduces the SLAM framework, classification, and laser SLAM methods. Generally, no matter the algorithm, relying on a single sensor to achieve full environmental awareness is very difficult. Neither Cartographer nor Gmapping can detect all obstacles, but at this stage, Cartographer detects more obstacles, giving it a perception advantage.

For 2D LiDAR SLAM, perceiving obstacles below the radar is impossible. Highly transparent objects can cause the laser-point return trajectory to deviate, so the radar may scan the object and fail to identify it as an obstacle. At this point, we need other sensors to improve autonomous vehicle perception. We could design relevant experiments based on sensor capabilities. Still, in complex real environments, achieving autonomous driving by relying on a single sensor is difficult. Therefore, in SLAM research, multi-sensor fusion provides better perception than a single sensor, which cannot be achieved by improving the algorithm alone.

In subsequent SLAM research, we should pay more attention to global optimization. Accumulated error reduces accuracy, and loop detection can eliminate it.

## Author Contributions

Conceptualization, Y. J. and J. Q.; Methodology, Y. J. and J. Q.; Software, Y. J. and J. Q.; Validation, Y. J. and J. Q.; Formal analysis, Y. J. and J. Q.; Investigation, Y. J. and J. Q.; Data Curation, Y. J. and J. Q.; Writing—Original Draft Preparation, Y. J. and J. Q.; Writing—Review and Editing, Y. J. and J. Q.; Visualization, Y. J. and J. Q.; Supervision, J. Q.; All authors have read and agreed to the published version of the manuscript.

The first author received scholarship support from CPALL for conducting this research in PIM.

## References

[1] A. Dwijotomo, M. A. Abdul Rahman, M. H. M. Ariff, H. Zamzuri, and W. M. H. Wan Azree, "Cartographer SLAM method for optimization with an adaptive multi-distance scan scheduler," *Appl. Sci.*, vol. 10, no. 1, Art. no. 347, Jan. 2020, https://doi.org/10.3390/app10010347

[2] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," *J. Open Source Softw.*, vol. 6, no. 61 p. 2783, May 2021, https://doi.org/10.21105/joss.02783

[3] Ł. Sobczak, K. Filus, J. Domańska, and A. Domański, "Finding the best hardware configuration for 2D SLAM in indoor environments via simulation based on Google Cartographer," *Sci. Rep.*, vol. 12, Art. no. 18815, Nov. 2022, https://doi.org/10.1038/s41598-022-22938-y

[4] Z. Liu, Z. Cui, Y. Li, and W. Wang, "Parameter optimization analysis of Gmapping algorithm based on improved RBPF particle filter," *J. Phys.: Conf. Ser.*, vol. 1646, no. 1, p. 012004, Sep. 2020, https://doi.org/10.1088/1742-6596/1646/1/012004

[5] X. Liu, G. Gong, X. Hu, G. Shang, and H. Zhu, "Cognitive enhancement of robot path planning and environmental perception based on Gmapping algorithm optimization," *Electronics*, vol. 13, no. 5, Art. no. 818, Feb. 2024, https://doi.org/10.3390/electronics13050818

[6] P. Wang, Z. Chen, Q. Zhang, and J. Sun, "A loop closure improvement method of Gmapping for low cost and resolution laser scanner," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 168-173, 2016, https://doi.org/10.1016/j.ifacol.2016.07.569

[7] Y. Li and J. Qu, "Intelligent road tracking and real-time acceleration-deceleration for autonomous driving using modified convolutional neural networks," *Curr. Appl. Sci. Technol.*, vol. 22, no. 6, Art. no. 013, 2022, https://doi.org/10.55003/cast.2022.06.22.013

[8] C. Tian, H. Liu, Z. Liu, H. Li, and Y. Wang, "Research on multi-sensor fusion SLAM algorithm based on improved Gmapping," *IEEE Access*, vol. 11, pp. 13690-13703, 2023, https://doi.org/10.1109/ACCESS.2023.3243633

[9] Q. Lin, Z. Ke, S. Bi, S. Xu, Y. Liang, F. Hong, and L. Feng, "Indoor mapping using Gmapping on embedded system," in *Proc. 2017 IEEE Int. Conf. Robotics and Biomimetics (ROBIO)*, 2017, pp. 2444-2449, https://doi.org/10.1109/ROBIO.2017.8324786

[10] C. Kim, H. Kim, and W. K. Chung, "Exactly Rao-Blackwellized unscented particle filters for SLAM," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 3589-3594, https://doi.org/10.1109/ICRA.2011.5980086

[11] B. Vincke, A. Elouardi, and A. Lambert, "Design and evaluation of an embedded system based SLAM applications," in *Proc. IEEE/SICE Int. Symp. System Integration (SII)*, 2010, p. 345, https://doi.org/10.1109/SII.2010.5708329

[12] W. G. Teame, W. Zhongmin, and Y. Yu, "Optimization of SLAM Gmapping based on simulation," *Int. J. Eng. Res. Technol.*, vol. 9, no. 4, pp. 74-81, Apr. 2020, https://pdfs.semanticscholar.org/c075/14524528655b397b19f1596a71597b38bf3e.pdf

[13] S. Lee, G. Tewolde, J. Lim, and J. Kwon, "2D SLAM solution for low-cost mobile robot based on embedded single board computer," in *Proc. World Congress on Advances in Nano, Bio, Robotics and Energy (ANBRE17)*, 2017, https://jrkwon.com/woordpress/wp-content/uploads/2023/02/YP.06RR1712_3813F5.pdf?utm_source=chatgpt.com

[14] M. Mazuran, G. D. Tipaldi, L. Spinello, W. Burgard, and C. Stachniss, "A statistical measure for map consistency in SLAM," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 3650-3655, https://doi.org/10.119/ICRA.2014.6907387

[15] S. Ding and J. Qu, "Smart car with road tracking and obstacle avoidance based on ResNet18-CBAM," in *Proc. 7th Int. Conf. Business and Industrial Research (ICBIR)*, 2022, pp. 582-585, https://doi.org/10.1109/ICBIR54589.2022.9786406

[16] B. L. E. A. Balasuriya et al., "Outdoor robot navigation using Gmapping based SLAM algorithm," in *Proc. Moratuwa Engineering Research Conf. (MERCon)*, 2016, pp. 403-408, https://doi.org/10.1109/MERCon.2016.7480175

[17] W. Zhang, Q. Zhang, K. Sun, and S. Guo, "A laser-SLAM algorithm for indoor mobile mapping," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. XLI-B4, pp. 351-355, 2016, https://doi.org/10.5194/isprs-archives-XLI-B4-351-2016

[18] Z. Zhang, H. Guo, G. Nejat, and P. Huang, "Finding disaster victims: A sensory system for robot-assisted 3D mapping of urban search and rescue environments," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2007, pp. 3889-3894, https://doi.org/10.1109/ROBOT.2007.364075

[19] P. Yang, "Efficient particle filter algorithm for ultrasonic sensor-based 2D range-only simultaneous localisation and mapping application," *IET Wireless Sens. Syst.*, vol. 2, no. 4, pp. 394-401, Dec. 2012, https://doi.org/10.1049/iet-wss.2011.0129

[20] S. Ding and J. Qu, "Research on self-driving based on dynamic recognition of traffic signs," in *Proc. 6th Int. Conf. Information Technology (InCIT)*, 2022, pp. 64-68, https://doi.org/10.1109/InCIT56086.2022.10067858

[21] A. V. Kanhere and G. X. Gao, "LiDAR SLAM utilizing normal distribution transform and measurement consensus," in *Proc. 32nd Int. Tech. Meeting Satellite Div. Inst. Navig. (ION GNSS+ 2019)*, 2019, pp. 2228-2240, https://doi.org/10.33012/2019.16954

[22] M. Rojas-Fernandez, D. Mújica-Vargas, M. Matuz-Cruz, and D. Lopez-Borreguero, "Performance comparison of 2D SLAM techniques available in ROS using a differential drive robot," in *Proc. Int. Conf. Electronics, Communications and Computers (CONIELECOMP)*, 2018, pp. 50-58, https://doi.org/10.1109/CONIELECOMP.2018.8327175

[23] T. J. Chong, X. J. Tang, C. H. Leng, M. Yogeswaran, O. E. Ng, and Y. Z. Chong, "Sensor technologies and Simultaneous Localization and Mapping (SLAM)," *Procedia Comput. Sci.* vol. 76, pp. 174-179, Dec. 2015, https://doi.org/10.1016/j.procs.2015.12.336

[24] Q. Shen, H. Sun, and P. Ye, "Research of large-scale offline map management in visual SLAM," in *Proc. 4th Int. Conf. Systems and Informatics (ICSAI)*, 2017, pp. 215-219.



**Yu Ji** is pursuing a master's degree in the Faculty of Engineering and Technology, Panyapiwat Institute of Management, Thailand. Our research mainly focuses on autonomous driving, with a a special emphasis on multi-sensor fusion technology. We are also working on autonomous driving systems based on image recognition.



**Jian Qu** is a full-time lecturer in the Faculty of Engineering and Technology, Panyapiwat Institute of Management. He received a Ph.D. with Outstanding Performance award from the Japan Advanced Institute of Science and Technology, Japan, in 2013. He received a B.B.A. with Summa Cum Laude honours from the Institute of International Studies of Ramkhamhaeng University, Thailand, in 2006, and an M.S.I.T. from Sirindhorn International Institute of Technology, Thammasat University, Thailand, in 2010. His research interests are natural language processing, intelligent algorithms, machine learning, machine translation, information retrieval, and image processing.