Indoor Localization Algorithm for Service Robot by Using Deep Neural Network

Chia-Wei Jan [1], Ping-Huan Kuo [1,2,*]

***Abstract*—The requirement for service robots has grown in many industries recently. Traditionally, Simultaneous Localization and Mapping (SLAM) is used for localization. However, it is not an efficient way since particles are needed to scatter every time. And it takes a lot of time for particles to calculate the position of the robot. Also, GPS has poor signal in indoor environments. In this paper, an indoor localization algorithm based on a deep neural network is proposed. In the deep neural network model, the inputs are the distance to obstacles and the angle of the robot gotten by LiDAR and compass. The output is the robot position. Since we are familiar with the indoor environment, the data is collected, and the model is trained in advance. Furthermore, a model that combines GoogleNet and Random Forest is used for prediction. In the path planning section, Probabilistic Roadmap (PRM) algorithm is used. Finally, the proposed localization algorithm is reliable and efficient shown in the experimental results.**

***Index Terms*—Indoor Localization, Path Planning, Deep Neural Network, GoogleNet, Random Forest**

I. Introduction

In recent years, the localization algorithm has attracted many researchers due to the development of mobile robots and autonomous vehicles. Traditionally, Simultaneous Localization and Mapping (SLAM) based on particle filter [1] and Kalman filter [2] is used to deal with localization problems [3]. In [4], Nak Yong Ko1 and Tae Gyun Kim compared the Kalman filter and particle filter used for underwater vehicle localization. Another SLAM involves cameras, known as visual SLAM [5], which lower the cost and generate numerous information. Furthermore, deep learning is also utilized for localization. In [6], the DDL-SLAM (Dynamic Deep learning SLAM) is proposed. Background Inpainting is used to improve the localization accuracy. Also, in [7], the robot localization problem is seen as a classification problem by using a convolution neural network (CNN).

The difference between outdoor and indoor localization is that GPS can be used outdoors. However, receiving poor GPS signals in indoors makes it a big challenge for localization. Therefore, Wi-Fi-based [8], [9], [10], [11] or RF-based [12], [13], [14], [15] localization systems are proposed by researchers.

However, since SLAM uses particles and keeps iterating to calculate the position of the robot, which may spend excessive time. In [16], the time complexity of SLAM is discussed and a square root unscented Kalman filter (SRUKF) is developed. This paper proposes an indoor localization algorithm based on a deep neural network. In an indoor environment, the data is collected, and the model is trained in advance. The algorithm of the deep neural network can immediately predict the position instead of spending a great time calculating particles and building a map.

The paper is organized as follows. Section II presents the system architecture to introduce the PR2 robot and the structure of the algorithm. Section III presents the proposed methods to show the deep learning model and path planning algorithm. In Section IV, experimental results are presented and discussed. Finally, section V shows the conclusion.

II. System Architecture

A PR2 robot is used in this research. The PR2 has a variety of sensors. It has 2 LiDARs called base laser and tilting laser. Also, there are 3 cameras on the head and 2 cameras on the left and right hand. For gripper sensors, accelerometer and fingertip pressure sensors are installed. An inertial measurement unit (IMU) and a speaker are located next to the tilting laser. The PR2 robot is shown in Fig. 1. An indoor environment is built in the simulator shown in Fig. 2.

Fig. 1. PR2 robot.

Fig. 2. The environment in the robot simulator.

This work was supported by the Ministry of Science and Technology, Taiwan, under Grants MOST 109-2221-E-194-053-MY3.

[1] Department of Mechanical Engineering, National Chung Cheng University, Chiayi 62102, Taiwan (e-mail: jcwccu@gmail.com, phkuo@ccu.edu.tw)

[2] Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI), National Chung Cheng University, Chiayi 62102, Taiwan (e-mail: phkuo@ccu.edu.tw)

[*] Corresponding author: Ping-Huan Kuo (e-mail: phkuo@ccu.edu.tw)

The distance to obstacles and the angle of the robot gotten by LiDAR and compass installed on the robot. These data are used to be the input of the deep learning model. Robot position is the output. The LiDAR installed on the base of PR2 has 270 degrees and 30 meters scanning range. And the output of the model is the position of the robot. Before training the model, the robot was controlled to move in the environment and stored the values of the LiDAR, compass, and position. Finally, 7,458 data was collected from the simulator.

In the proposed algorithm, values are gotten from LiDAR and compass first. Second, the data is used as input for deep learning models. Third, the output of models is gotten as the current position and the robot is moved to the next position. Finally, check whether the current position equal to the target position. If true, end the program; otherwise, values are kept getting from LiDAR and compass. The structure of the algorithm is shown in Fig. 3.

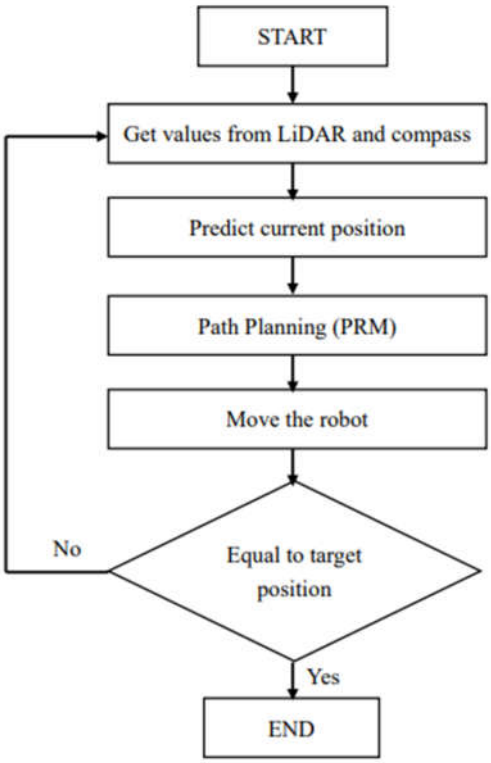

Fig. 3. Structure of the algorithm.

III. Proposed Method

Before training a model, we need to prevent the bias from relying on one particular way of training and testing datasets. For example, if the testing data same as the training data, we can not prove the model can predict well other data even if the accuracy is 100%. Therefore, we need to use a stricter way to access the model.

Cross-validation uses several ways to partition the original dataset into training and testing data and calculate the average result of different partitions. *K*-fold and leave-one-out are both common ways of cross-validation. Here I choose k-fold cross-validation to evaluate the performance.

K-fold cross-validation means the original dataset is split into k sets. Using k-1 folds for training and one fold for testing the model, and then iteration for k times. Finally, average the results of each iteration. Fig. 4 shows *K*-fold cross-validation.

Decision Tree (DT) is like a tree structure, including a root node, internal node, and leaf node. Each branch holds a result of the test and each leaf node has a class label. It starts from the root node and divides the dataset into more accurate subsets. Fig. 5 shows the structure of DT.

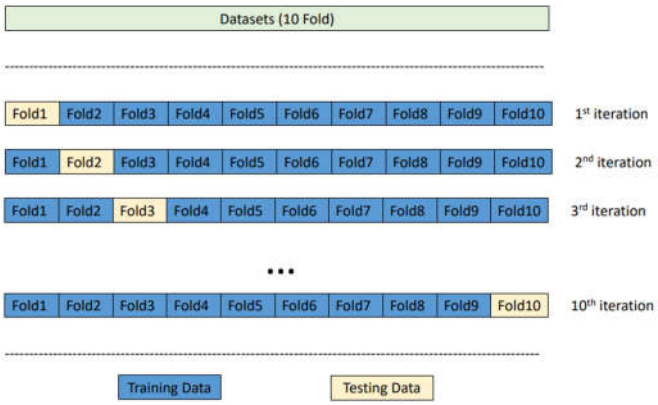

Fig. 4. *K*-fold cross-validation.

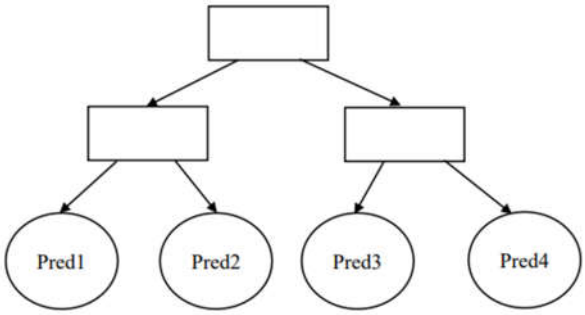

Fig. 5. Decision Tree.

Random Forest (RF) is constructed by several decision trees and the final result is determined by the output of each tree. By iterative calculation, the target will converge. Finally, the majority voting or average method is used to get the final result of the random forest model. Fig. 6 shows the structure of RF.

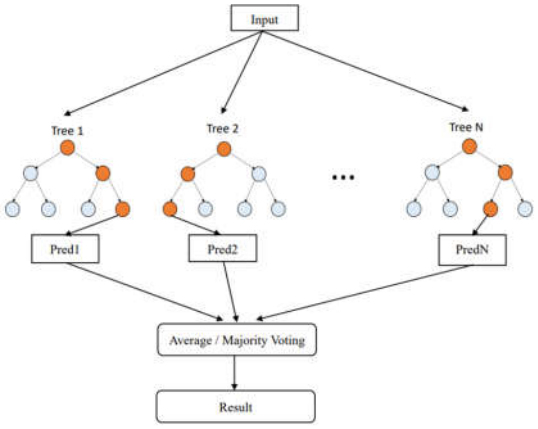

Fig. 6. Random Forest.

Support Vector Regression (SVR) is based on Support Vector Machine (SVM). If the distance between $f(x)$ and y is short, the prediction will be considered correct. An ε is added to the left and right of $f(x)$ as the model tolerance interval. Therefore, only the errors outside the dotted line will be calculated during the training process. Fig. 7 shows SVM.

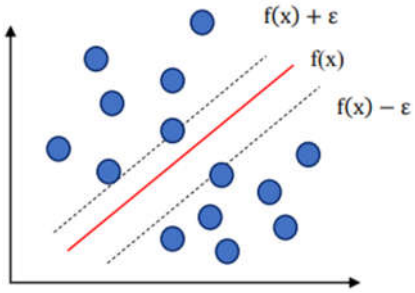

Fig. 7. Support Vector Machine.

AdaBoost is the abbreviation of "Adaptive Boosting." The wrong samples of the previous basic learner will be strengthened, and the weighted whole samples will be used to train the next basic learner again. At the same time, a new weak learner is added to each round until it reaches a predetermined sufficiently small error rate or reaches the pre-specified maximum number of iterations. Fig. 8 shows AdaBoost.

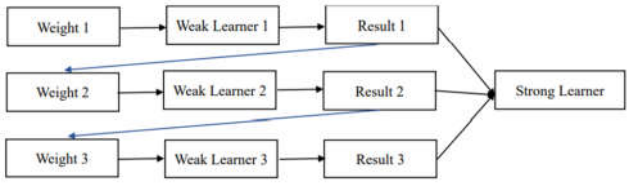

Fig. 8. AdaBoost.

ResNet is a network based on residual learning. More layers are added to increase the performance, but Degradation may happen. ResNet solved this issue and add more layers successfully. Having skip connections allows the network to learn identity mappings more easily. (Fig. 9) Therefore, with a residual block, the model will not degrade. Instead of convolution layers, dense layers are used to build a ResNet-like model in this paper. Fig. 10 shows the ResNet-like block.

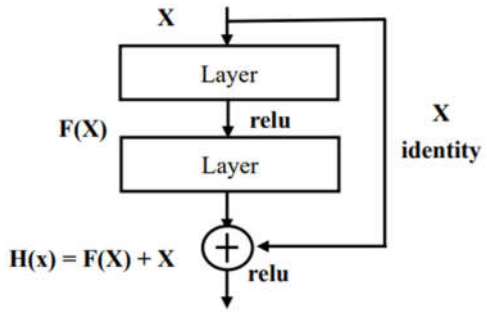

Fig. 9. Skip Connection.

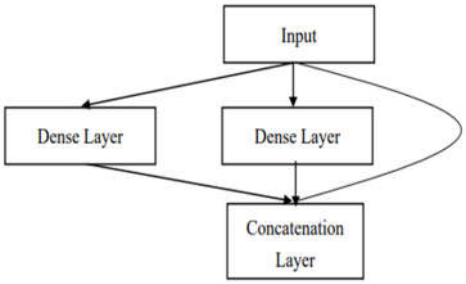

Fig. 10. ResNet-like block.

GoogleNet was proposed in 2014. Its structure is based on Inception Net. Due to over parameters in Inception Net, GoogleNet uses techniques such as 1×1 convolutions in the middle of the architecture and global average pooling to decrease the number of parameters. Instead of convolution layers, dense layers are used to build a GoogleNet-like model in this paper. Fig. 11 shows the GoogleNet-like block.

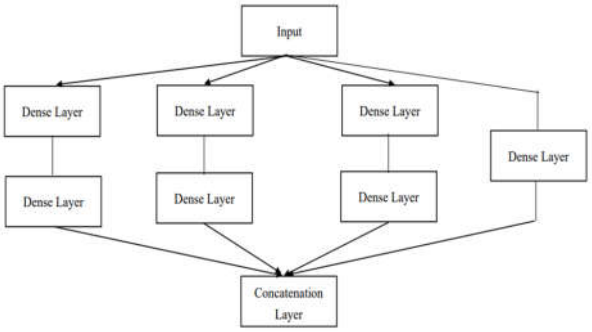

Fig. 11. GoogleNet-like block.

Since Random Forest and GoogleNet-like models have better performance, the voting method is used in this paper to get characteristics from both models. The voting formula is shown in (1).

$$Pred = \frac{{MSE_{RF}}^{-1} \times Pred_{RF} + {MSE_G}^{-1} \times Pred_G}{{MSE_{RF}}^{-1} + {MSE_G}^{-1}} \quad (1)$$

MSE_{RF} and $Pred_{RF}$ are mean square error (MSE) and prediction of training data using Random Forest. MSE_G and $Pred_G$ are MSE and prediction of training data using GoogleNet.

Before doing path planning, it is required to get the position of obstacles. The map is built to imitate the top view of the indoor. The black blocks are the obstacles. The top view of the indoor environment is shown in Fig. 12.

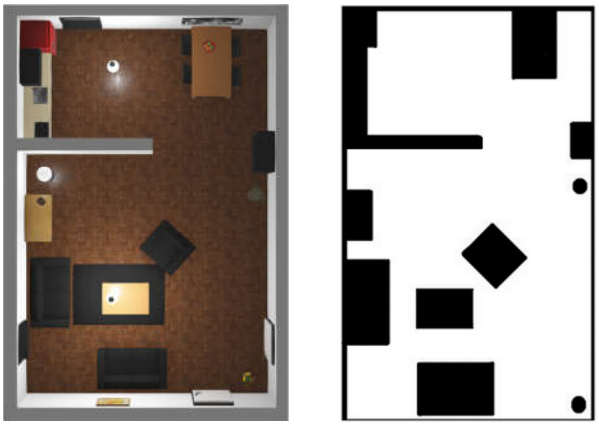

Fig. 12. Indoor Environment.

Path planning is a computational problem that makes a robot find an optimal path between two positions and avoids colliding with obstacles. Most path planning algorithm is based on a data structure called a Graph consisting of nodes and edges. In this paper, Probabilistic Roadmap (PRM) algorithm is used for path planning. Before implementing PRM, the map mentioned above is read in the program. The black areas will be transformed into obstacles.

After getting the position of obstacles, the PRM algorithm generates random nodes in the configuration space. (Fig. 13) Add the node that does not intersect with obstacles into the graph. Then, connect the new node with the closest node through an edge. Add the edge that does not intersect with obstacles into the graph. Finally, a graph is constructed by the above step. After building a complete graph, the shortest path can be found using the Dijkstra algorithm. The PRM algorithm is shown in Fig. 14.

Compare with the general A* and Dijkstra algorithm which computes all the points in the map, Probabilistic Roadmap only samples a bunch of points so that it is more efficient.

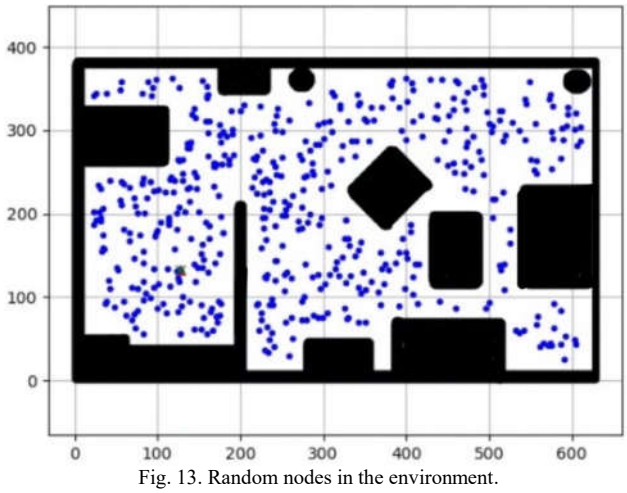

Fig. 13. Random nodes in the environment.

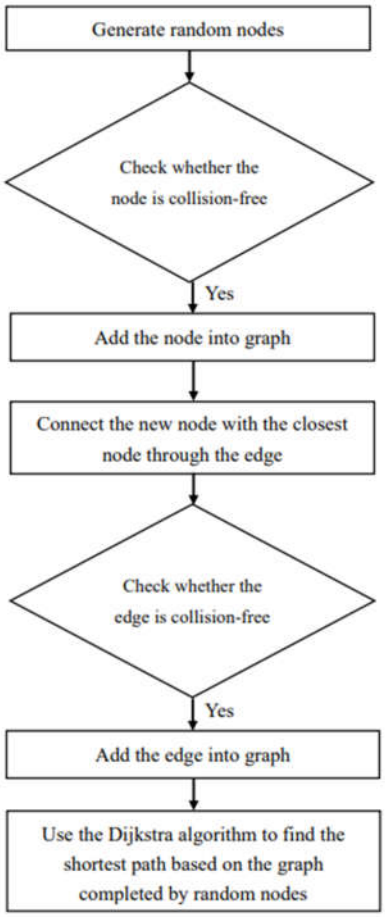

Fig. 14. Probabilistic Roadmap Algorithm.

Forward kinematics determines the end of a kinematics chain, which end effector. There is only one solution to the forward kinematic equation. Inverse kinematics calculates the joint positions that are needed to place the end effector of the robot at a specific position and orientation. The forward kinematics formula is shown in (2).

$$\mathrm{T} = \mathrm{f}(\theta_1, \theta_2, \ldots, \theta_n) \tag{2}$$

The inverse kinematics formula is shown in (3).

$$[\theta_1, \theta_2, \ldots, \theta_n] = f^{-1}(\mathrm{T}) \tag{3}$$

Homogeneous Transformation Matrix (HTM) combines both rotation and displacement into a matrix. It can be expressed as a 4×4 matrix, including a 3×3 rotation matrix and a vector of 3 dimensions. HTM is shown in Fig. 15.

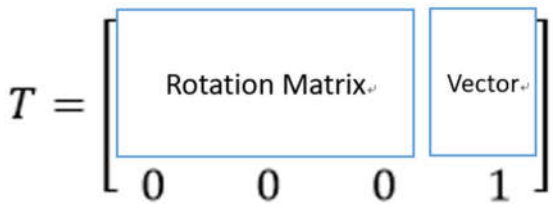

Fig. 15. Homogeneous transformation matrix.

The rotation of HTM around the x-axis is shown in (4).

$$T(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

The rotation of HTM around the y-axis is shown in (5).

$$T(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The rotation of HTM around the z-axis is shown in (6).

$$T(\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

IV. Experimental Results

The mean absolute error (MAE) and mean square error (MSE) are two model evaluation metrics for regression models. MAE uses absolute value which can avoid to offset positive and negative numbers. The MAE formula is shown in (7).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (7)$$

Compared with MAE, MSE can amplify the value with larger prediction deviation and compare the stability of various prediction models; in addition, it can be differentiated, so it is often used as a loss function. RMSE is the square of MSE, which is the same unit as data.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (8)$$

Table I use MAE and RMSE to compare the performance of the models mentioned above.

TABLE I.
Performance of The Models

Model	MAE	RMSE
Decision Tree	0.155	0.6
Random Forest	0.142	0.341
SVM	0.548	0.753
AdaBoost	1.62	1.86
MLP	0.249	0.44
ResNet	0.213	0.41
GoogleNet	0.131	0.344
Random Forest & GoogleNet	0.117	0.283

The process of the robot grasping an object is shown in Fig. 16. Inverse kinematics is used to calculate the angle of the robot arm.

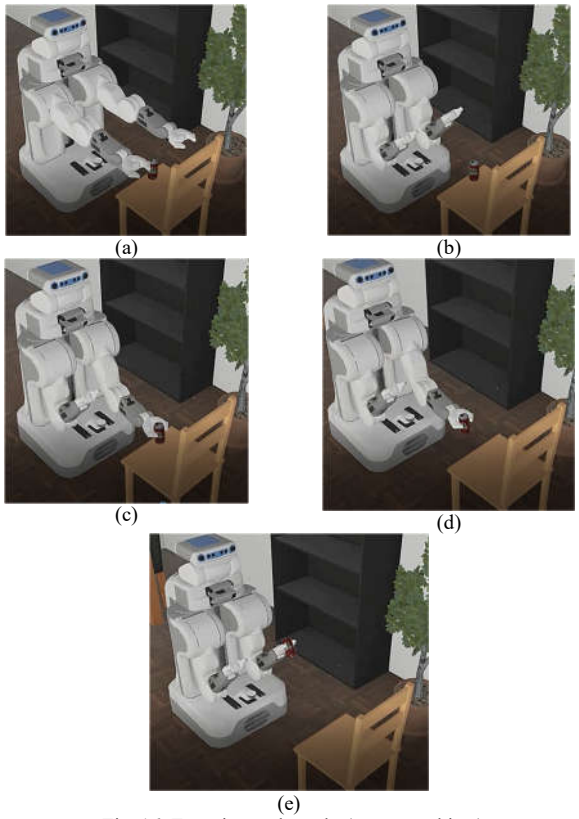

Fig. 16. Experimental result. (grasp an object)

The path of the PRM algorithm is shown in Fig. 17. Also, Fig. 18 shows the process of localization. The robot starts from the corner of the environment, and the target position is close to the sofa.

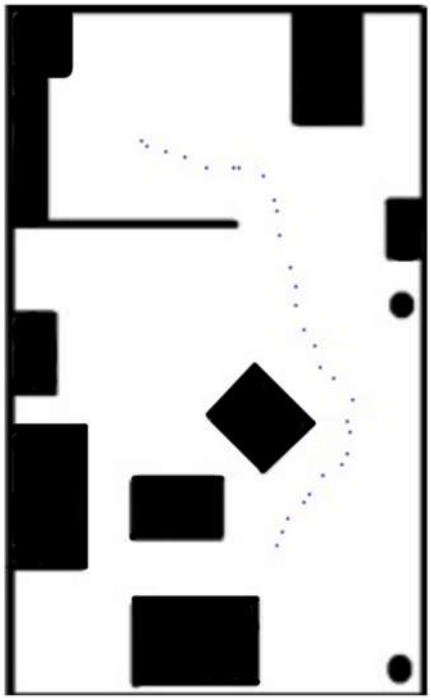
Fig. 17. Path of PRM algorithm.

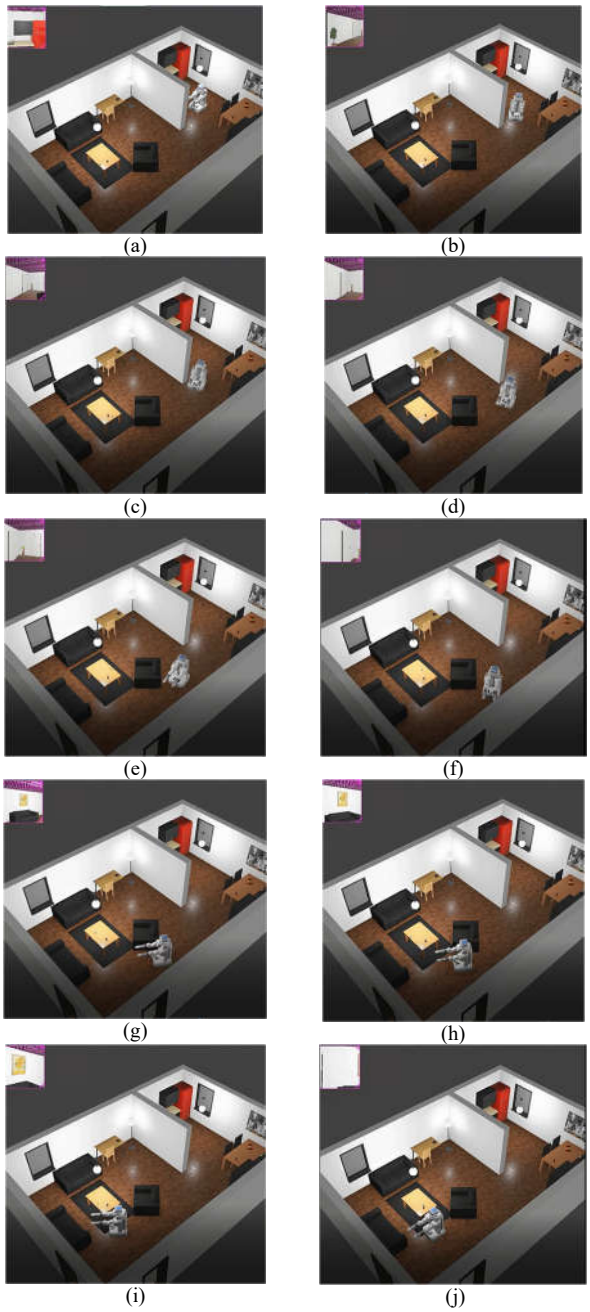

Fig. 18. Experimental result. (localization process)

V. CONCLUSION

In this paper, an algorithm of localization using a deep neural network is proposed. We firstly collect the data in the indoor environment, including the values of LiDAR, compass, and robot position. A map is drawn to show the position of obstacles, so that path planning can be done. For the localization, get the values of LiDAR and compass as input, then get the output of the model as the current position. In an indoor environment, using a deep neural network is more efficient since a bunch of particles does not need to be calculated. In the future, we plan to implement a localization algorithm using the particle filter and compare its performance with the deep neural network.

REFERENCES

[1] Chunlei Ji, Haijun Wang, and Qiang Sun, "Improved particle filter algorithm for robot localization," in *2010 2nd International Conference on Education Technology and Computer*, Jun. 2010, pp. V4-171-V4-174, doi: 10.1109/ICETC.2010.5529710.

[2] I. Rhodes, "A tutorial introduction to estimation and filtering," *IEEE Trans. Automat. Contr.*, vol. 16, no. 6, pp. 688–706, Dec. 1971, doi: 10.1109/TAC.1971.1099833.

[3] I. Ullah, Y. Shen, X. Su, C. Esposito, and C. Choi, "A Localization Based on Unscented Kalman Filter and Particle Filter Localization Algorithms," *IEEE Access*, vol. 8, pp. 2233–2246, 2020, doi: 10.1109/ACCESS.2019.2961740.

[4] N. Y. Ko and T. G. Kim, "Comparison of Kalman filter and particle filter used for localization of an underwater vehicle," in *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Nov. 2012, pp. 350–352, doi: 10.1109/URAI.2012.6463013.

[5] H. Bavle, P. De La Puente, J. P. How, and P. Campoy, "VPS-SLAM: Visual Planar Semantic SLAM for Aerial Robotic Systems," *IEEE Access*, vol. 8, pp. 60704–60718, 2020, doi: 10.1109/ACCESS.2020.2983121.

[6] Y. Ai, T. Rui, M. Lu, L. Fu, S. Liu, and S. Wang, "DDL-SLAM: A Robust RGB-D SLAM in Dynamic Environments Combined With Deep Learning," *IEEE Access*, vol. 8, pp. 162335–162342, 2020, doi: 10.1109/ACCESS.2020.2991441.

[7] G. Kim, B. Park, and A. Kim, "1-Day Learning, 1-Year Localization: Long-Term LiDAR Localization Using Scan Context Image," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1948–1955, Apr. 2019, doi: 10.1109/LRA.2019.2897340.

[8] H. Abdelnasser *et al.*, "SemanticSLAM: Using Environment Landmarks for Unsupervised Indoor Localization," *IEEE Trans. Mob. Comput.*, vol. 15, no. 7, pp. 1770–1782, Jul. 2016, doi: 10.1109/TMC.2015.2478451.

[9] M. Shu, G. Chen, and Z. Zhang, "3D Point Cloud-Based Indoor Mobile Robot in 6-DoF Pose Localization Using a Wi-Fi-Aided Localization System," *IEEE Access*, vol. 9, pp. 38636–38648, 2021, doi: 10.1109/ACCESS.2021.3060760.

[10] M. Zhou, Y. Li, M. J. Tahir, X. Geng, Y. Wang, and W. He, "Integrated Statistical Test of Signal Distributions and Access Point Contributions for Wi-Fi Indoor Localization," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5057–5070, May 2021, doi: 10.1109/TVT.2021.3076269.

[11] L. Chen, K. Yang, and X. Wang, "Robust Cooperative Wi-Fi Fingerprint-Based Indoor Localization," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1406–1417, Dec. 2016, doi: 10.1109/JIOT.2016.2609405.

[12] I. T. Haque and C. Assi, "Profiling-Based Indoor Localization Schemes," *IEEE Syst. J.*, vol. 9, no. 1, pp. 76–85, Mar. 2015, doi: 10.1109/JSYST.2013.2281257.

[13] W. Zhu, J. Cao, Y. Xu, L. Yang, and J. Kong, "Fault-Tolerant RFID Reader Localization Based on Passive RFID Tags," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2065–2076, Aug. 2014, doi: 10.1109/TPDS.2013.217.

[14] A. Tzitzis *et al.*, "Localization of RFID Tags by a Moving Robot, via Phase Unwrapping and Non-Linear Optimization," *IEEE J. Radio Freq. Identif.*, vol. 3, no. 4, pp. 216–226, Dec. 2019, doi: 10.1109/JRFID.2019.2936969.

[15] Z. Chen, M. I. AlHajri, M. Wu, N. T. Ali, and R. M. Shubair, "A Novel Real-Time Deep Learning Approach for Indoor Localization Based on RF Environment Identification," *IEEE Sensors Lett.*, vol. 4, no. 6, pp. 1–4, Jun. 2020, doi: 10.1109/LSENS.2020.2991145.

[16] S. A. Holmes, G. Klein, and D. W. Murray, "An $O(N^2)$ Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 7, pp. 1251–1263, Jul. 2009, doi: 10.1109/TPAMI.2008.189.