# ROS-base Multi-Sensor Fusion for Accuracy Positioning and SLAM System

Yi-Xiang,Wang
National Yunlin University of Science and Technology
Yunlin,Taiwan
M10717011@gemail.yuntech.edu.tw

Ching-Lung,Chang
National Yunlin University of Science and Technology
Yunlin,Taiwan
chang@yuntech.edu.tw

**To facing lower birth rate and aging society trend, how to design intelligent robots demanded in different fields to reduce the shortage of human resource has become an one of important research topics in recent years. In order to satisfy mobility ability of intelligent robots, accurate indoor positioning is an important function。**

**In this paper, we use the Robot Operation System(ROS) as our platform, combining with different positioning sensors and technologies, such as Light Detection and Ranging(LiDAR)、Inertial Measurement Unit (IMU)、odometer and Ultra-wideband(UWB), and fuse these data by Extended Kalman Filter(EKF) to provide accuracy positioning. Thus, we can achieve more refined Simultaneous Localization and Mapping(SLAM) which meets the intelligent robot application demand. Experiment results show that the average error distance of the mobile robot in our system can be limited in 10cm.**

**Keywords: Automated guided vehicle(AGV), Robot Operation System(ROS), Ultra-wideband(UWB), Light Detection and Ranging(LiDAR), Simultaneous Localization and Mapping(SLAM)**

## I. INTRODUCTION

As technology advances, with aging society and the lower births, in order to reduce related impacts, there is a demand for various types of intelligent robots, promote countries to invest a lot of resources in the intelligent robot industry.

For mobile robots, field mapping and accurate indoor positioning are important functions. Have good field map and positioning information can provide path planning, precise mobile control and obstacle avoidance capabilities.

In terms of positioning, it's divided into outdoor positioning and indoor positioning. Outdoor positioning mainly relies on GPS system to provide positioning functions. For indoor positioning, base on the positioning accuracy and cost are have many different technologies to choose, such as inertial positioning, base on signal strength estimation to Bluetooth/Wi-Fi positioning, and UWB (Ultra-wideband) positioning base on signal transmission time and LiDAR positioning by laser ranging.

ROS (Robot Operating System) is an operating platform exclusively designed for robot control, this paper is based on ROS platform, combined with inertial positioning, LiDAR, UWB and other sensors, combined with an Extended Kalman Filter(EKF) to achieve accurate field mapping and positioning system.

## II. RELATED WORKS

When the robot is moving control, it wants to walk steadily according to the planned path to reach the designated location, the positioning function is very important. In recent years, due to the maturity of LiDAR technology and the breakthrough of software algorithms, the SLAM technology combined with the development of LiDAR laser ranging scanning has been continuously improved, the advent of robot vacuum cleaner and unmanned aerial vehicle makes SLAM an important function of self-propelled robots. Next, we will introduce the SLAM algorithm and related papers.

### *A. Simultaneous Localization and Mapping*

The earliest origin was developed in the field of robots. It originally meant that robots started from an unknown environment, acted in this environment, and obtained their own positioning and status through repeated observation of environmental information, and then learned the surroundings based on their positions to constructed map. Currently, according to sensors for SLAM can be divided into two categories: laser SLAM and visual SLAM.

The earliest origin was developed in the field of robots. It originally meant that robots started from an unknown environment, acted in this environment, and obtained their own positioning and status through repeated observation of environmental information, and then learned the surroundings based on their positions to constructed map. Currently, according to sensors for SLAM can be divided into two categories: laser SLAM and visual SLAM.

At present, the most common SLAM algorithm is Gmapping SLAM developed by [1] in 2007, [2] developed Hector SLAM proposed in 2010, and [3] developed Cartographer SLAM proposed in 2016. Gmapping SLAM use the RBpf particle filter algorithm to process the environment perceived by the robot works with the wheel odometer to calculate the robot's position and construct map. Hector SLAM does not need to use odometer to calculate, only rely on LiDAR to complete SLAM, but must use high-precision LiDAR. Cartographer SLAM first performs a certain number of laser points to construct a sub-map, and the sub-maps are assembled into a whole map, and the ring avoidance detection is introduced. The sub-map is optimized every certain time, it likes similar to the visual SLAM algorithm for various environments.

### *B. Multi-Sensor Fusion in SLAM*

In order to overcome environmental problems and reduce positioning errors, people have begun to research

978-1-7281-8741-9/20/$31.00 ©2020 IEEE

the fusion of different types of sensors to improve positioning accuracy and apply it to SLAM in recent years.[4] proposed a new method of fusion of LiDAR and visual SLAM to obtain more accurate positioning.

Base on laser SLAM and base on monocular camera SLAM has the disadvantage of large positioning error. Integrating two SLAM algorithms can improve this problem.[5] proposed a method based on least squares and obtaining squared measurements from UWB sensors to position the robot in an indoor environment. In addition, an initialization algorithm based on LiDAR scanning is proposed, using the advantage of estimated position to find the initial direction of the robot relative to the previously obtained map.[6] proposed an inertial sensor with additional radio signal support for indoor navigation. Experiment of data fusion between an IMU based navigation system and a pulse-based UWB positioning system. In order to achieve localized EKF, [7] fused the odometer and Pozyx range measurement values to improve the accuracy of Pozyx algorithm. [8] proposed the fusion of odometer and use LiDAR scanning sensor for indoor positioning, using LiDAR scanning to compensate the accumulated error of the odometer, and applying it to the Extended Kalman Filter to achieve robot positioning

## III. BACKGROUND

### A. Robot Operation System

Robot Operating System also known as ROS which is an open source software based on Linux environment. It is a framework developed for writing robot software. It can provide services similar to the operating system, including hardware abstract description, low-level driver management, execution of shared functions, message between programs, program distribution package management, and also provides some libraries and tools for get, create, write and execute programs between computers.

The ROS architecture is shown in Fig. 1, divided into three levels: OS layer, Middleware layer and Application layer. The OS layer needs to be built in the Linux system, usually use Ubuntu. The Middleware layer is mainly used for TCPROS/UDPROS communication. Based on the TCP/UDP network, the communication system uses subscribe/publish, client/server and other models to implement multiple communication mechanisms of data transmission. In the Application layer, ROS needs a manager "Master", responsible for managing the operation of the entire system. In addition, a large number of robot application function packages have been shared in the ROS community. The models in these packages are operated in Node units. Through the ROS standard input and output as interfaces, developers do not need to understand the implementation mechanism in the model, only need to understand the interface The regulations can be reused, greatly improving the efficiency of robot development.

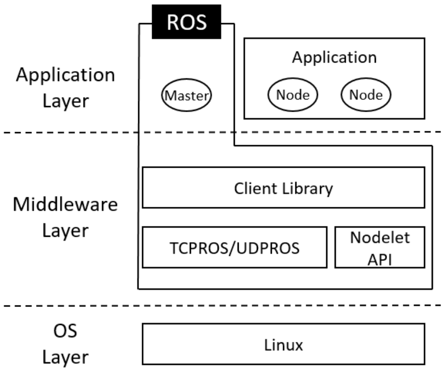


Fig. 1. ROS architechure

About between nodes communication in Fig. 2, include Nodes, Master, Messages, Topics, Services, Bags. Nodes are running programs. ROS is designed as a multi-module system. A robot system usually contains many Nodes, for example: one Node controls LiDAR, another Node handles robot positioning, etc. Master is the main manager of ROS. Without Master, Node cannot find other Node to exchange messages. The transmission of information through Messages between Node and Node is a simple data structure. Messages are delivered in a subscribe/publish manner. A node can get Message by subscribe/publish the topic. Topic is a name used to describe the content of the message. There may be multiple Nodes publish or subscribe to the same Topic at the same time; there may also be a Node publish or subscribe to multiple Topics at the same time. In ROS, there is also a synchronous transmission mode-Services, which is based on the client/server model and contains two parts of the communication model: one for request and the other for response, similar to Web services. Bags is a format used to store and play ROS messages. Bags is an important mechanism for storing data. For example, sensor data is difficult to collect but is necessary for development and testing.

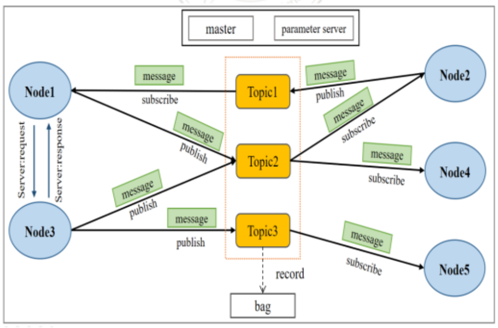


Fig. 2. Between nodes communication

### B. Gmapping SLAM

Simultaneous Localization and Mapping is a chicken and egg problem. Perfect positioning require use an error-free map, but such a map require accurate location estimation to construct. This is the starting condition for an iterative mathematical problem solving strategy. Gmapping SLAM is a mature SLAM algorithm based on the RBpf particle filter algorithm and then improved, it makes us relatively faster in robot development.

The RBpf algorithm is derived from a particle filter. The particle filter is a recursive filter using the Monte Carlo method. It passes a set of random samples (call particles ) to represent the posterior probability of random events, from the measurement sequence containing noise, the state of the dynamic system is estimated, compared with the Kalman Filter is based on linear state space and Gaussian distribution of noise, The state space model of the particle filter can be non-linear, and the noise distribution can be of any type. As shown in Fig. 3, first, initialize the particles and evenly disperse the particles, then make predictions and update them with the measured state comparison, and then resampling. Through such continuous recursion, eventually it converges into a small group of particles which represents the position of the robot.

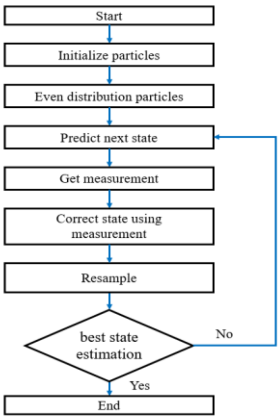


Fig. 3. Particle Filter process

The main problem of the RBpf algorithm is its high complexity because it require more particles to construct the map and frequently perform resampling. Since the number of particles is closely related to the amount of calculation, the larger the number of particles, the more complicated the algorithm will be. Therefore, reducing the number of particles and reducing the number of resampling to prevent particle degradation is an improvement of the RBpf algorithm. However, Gmapping SLAM proposes two points on the basis of RBpf: improved proposed distribution and selective resampling to improve the problem of RBpf algorithm. The improved proposal distribution not only considers the information of the motion model (odometer information), but also the information of the last measurement model (LiDAR information) so that the proposal distribution can be more accurate and close to the target. Selective resampling is relatively simple. By setting a threshold, the resampling action is only performed when the weight of the particles changes beyond the threshold, which greatly reduces the number of resampling, makes the calculation less complicated and also the calculation performance

## IV. System Architecture

The system architecture is shown in Fig. 4. There are three types of sensor information: odometer (motor), inertial measurement unit(IMU),and ultra-wideband (UWB). Data fusion is performed through an extended Kalman filter, and the combined multi-sensor The sensor information and LiDAR information were sent to SLAM for mapping. When the robot wants to move to the target, it must know where it is on the map. At this time, AMCL Positioning needs to be used to locate the robot, and then the robot will start path planning after knowing its location. , and perform moving control actions.

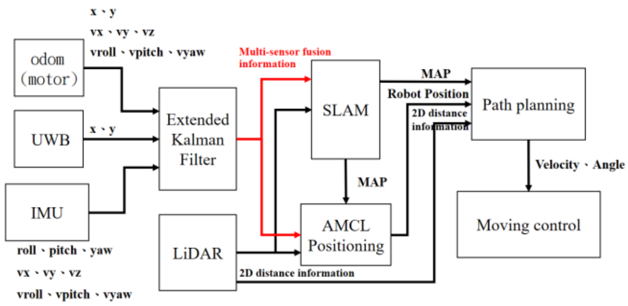


Fig. 4. System architecture

### A. Hardware Architecture

The hardware architecture shown in Fig. 5 is divided in PC and robot. The CPU of the PC is i5-6200, the RAM has 8G, the hard disk capacity is 500G, and it supports wireless network 802.11 ac/a/b/g/n. The robot has four parts: control board, main board, power system, and sensors. There are two kinds of devices in the control, OpenCR control board and the XL430-W250-T intelligent motor. The two communicate through I/O. The main board Raspberry Pi 3 b+ serves as the core of the entire robot and communicates with other components via USB. There are two types of power systems, one is the power bank supply for Raspberry Pi 3 b+, and the other is the Li-Po battery for OpenCR control board. There are three types of sensing elements, inertial measurement unit(IMU) on OpenCR, RPLiDAR A1 and Pozyx(UWB), IMU provides nine-axis information, RPLiDAR A1 provides 2D distance information, and Pozyx (UWB) provides absolute position. In addition, Pozyx has an anchor and a tag, and the anchor is arranged at four positions in the area. The position of the tag is obtained by an algorithm. Therefore, we install the tag on the robot to receive position information. Fig. 6 and Fig. 7 is the forward graph and lateral graph of the robot entity.

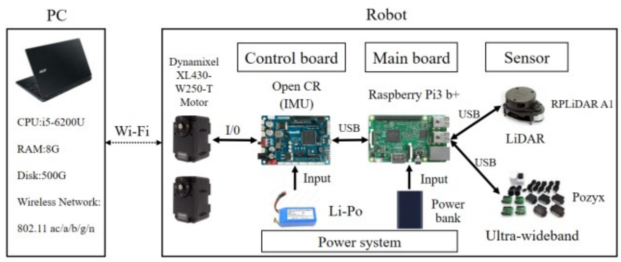


Fig. 5. Hardware architecture

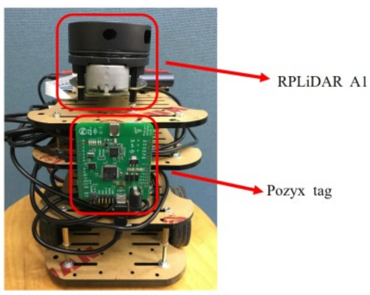


Fig. 6. Robot entity front view

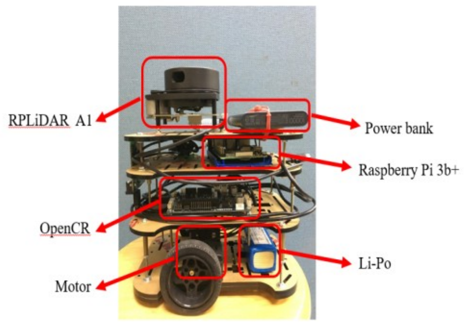


Fig. 7. Robot entity side view

### B. *Gmapping SLAM*

Fig. 8 shows the Gmapping SLAM architecture. The SLAM construction requires information from the odometer (motor) and LiDAR. The odometer (motor) provides x, y, vx, vy, vz, vroll, vpitch, vyaw. LiDAR provides 2D distance information. After building a map, map will be generated with some relevant information, such as: map resolution, width, height, origin position and direction, and map information (occupied is 100, unoccupied is 0, unknown is -1).

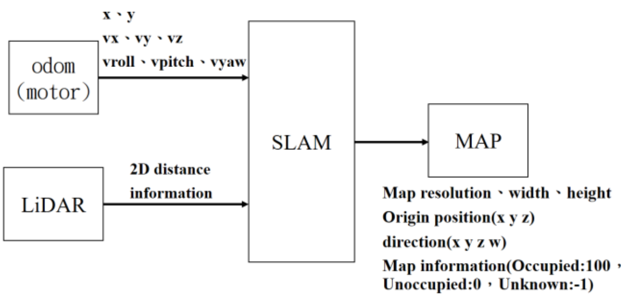


Fig. 8. Gmapping SLAM architecture

### C. *Multi-Sensor Fusion for Positioning and SLAM*

Fig.9 is Method 1、odom + IMU fusion positioning architecture. Combines the odometer (motor) and inertial measurement unit (IMU) with extended Kalman filter as the sensing data fusion. The odometer (motor) provides x, y, vx, vy, vz, vroll, vpitch, vyaw. Inertial measurement unit(IMU) provide roll, pitch, yaw, vx, vy, vz, vroll, vpitch, vyaw information, and then the multi-sensor fusion information and LiDAR information are sent to SLAM to build the map.

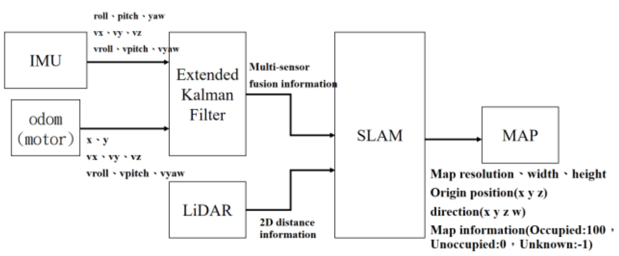


Fig. 9. Method 1、odom+imu fusion positioning

Fig.10 is Method 2、odom + UWB fusion positioning architecture. Combines the odometer (motor) and ultra-wideband(UWB) with extended Kalman filter as the sensing data fusion. The odometer (motor) provides x, y, vx, vy, vz, vroll, vpitch, vyaw. Ultra-wideband(UWB) provide x, y information, and then the multi-sensor fusion information and LiDAR information are sent to SLAM to build the map.

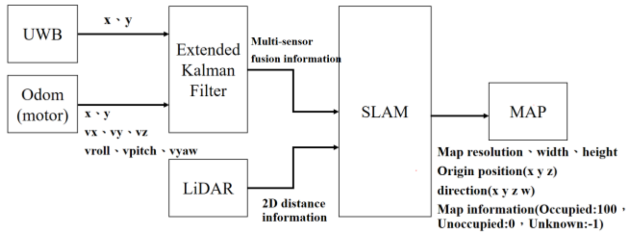


Fig. 10. Method 2、odom+uwb fusion positioning

Fig.11 is Method 3、odom + UWB + IMU fusion positioning architecture. Combines the odometer (motor)、inertial measurement unit (IMU) and ultra-wideband(UWB) with extended Kalman filter as the sensing data fusion. The odometer (motor) provides x, y, vx, vy, vz, vroll, vpitch, vyaw. Inertial measurement unit(IMU) provide roll, pitch, yaw, vx, vy, vz, vroll, vpitch, vyaw information. Ultra-wideband(UWB) provide x, y information, and then the multi-sensor fusion information and LiDAR information are sent to SLAM to build the map.

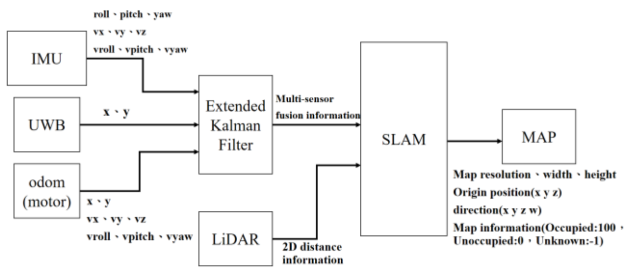


Fig. 11. Method 3、odom+imu+uwb fusion positioning

The Extended Kalman Filter works as formulas (1) ~ (5). First, we make an estimate of the position of the robot with an estimated noise, and then use the information from the sensor as a measurement and update it to get a more realistic value, and continue through such steps the recursion is used to get more accurate robot position.

Predicted state estimate：

$$\hat{x}_{(k|k-1)} = f(x_{k-1}, u_k, 0) \tag{1}$$

Predicted estimate covariance：

$$\mathrm{P}_{(k|k-1)} = F_k \mathrm{P}_{(k-1|k-1)} F_k^T + Q_k \tag{2}$$

Updated state estimate：

$$\hat{x}_{(k|k)} = \hat{x}_{(k|k-1)} + K_k \tilde{y}_k \tag{3}$$

Optimal Kalman gain：

$$\mathrm{K}_k = \mathrm{P}_{(k|k-1)} \mathrm{H}_k^T S_k^{-1} \tag{4}$$

Updated estimate covariance：

$$\mathrm{P}_{(k|k)} = (\mathrm{I} - \mathrm{K}_k \mathrm{H}_k) \mathrm{P}_{(k|k-1)} \tag{5}$$

## V. EXPERIMENTAL RESULTS

### A. Experimental environment

Fig. 12 shows the actual environment area used for the experiment. We fixed the four Pozyx Anchors in four different positions.


Fig. 12. Actual environment area

### B. Experimental method

The experimental method of path planning is shown in Fig. 13 We plan a path with a total length of 5.2 meters, place an obstacle on the way, start from the starting point, and pass A (2.1,0), B (2.1,0.4), C (3.4 0.4), D (3.2, 0), and finally reach point E (end point) and measure the error from the end point.

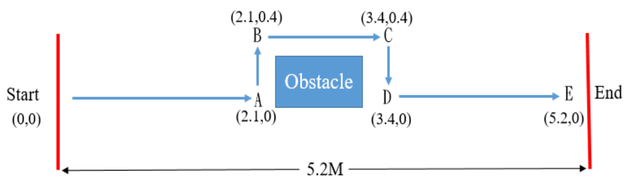


Fig. 13. Path planning with obstacles

### C. Experimental result

We use different method architectures, conduct a total of ten times through the path planning method with obstacles, and record the path of the robot move and measure the error from the final distance.

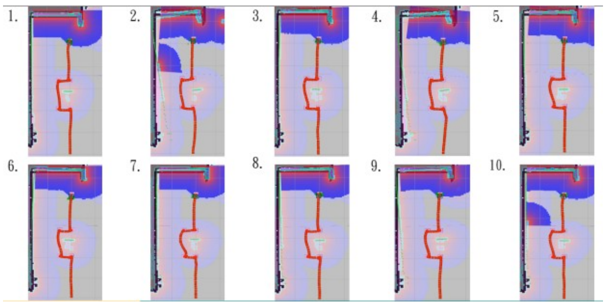

Fig. 14. The travel path of Gmapping SLAM

TABLE I. The final distance error of Gmapping SLAM

| Item | Final distance estimate(cm) | Final distance measure(cm) | Error (cm) | Time (s) |
|---|---|---|---|---|
| 1 | 520 | 510. 8 | 9. 2 | 52 |
| 2 | 520 | 508. 4 | 11. 6 | 50 |
| 3 | 520 | 506. 1 | 13. 9 | 47 |
| 4 | 520 | 506. 2 | 13. 8 | 47 |
| 5 | 520 | 509. 6 | 10. 4 | 49 |
| 6 | 520 | 507. 0 | 13. 0 | 47 |
| 7 | 520 | 503. 0 | 17. 0 | 53 |
| 8 | 520 | 506. 7 | 13. 3 | 51 |
| 9 | 520 | 501. 7 | 18. 3 | 50 |
| 10 | 520 | 507. 2 | 12. 8 | 48 |

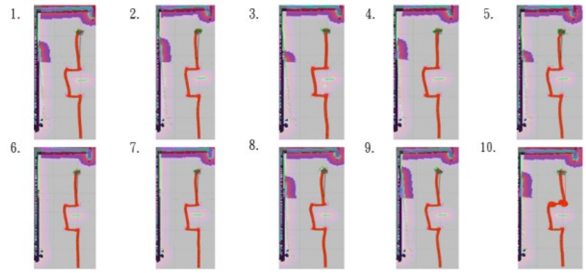

Fig. 15. The travel path of method 1 、odom+imu fusion positioning

TABLE II. The final distance error of method 1 、odom+imu fusion positioning

| Item | Final distance estimate(cm) | Final distance measure(cm) | Error (cm) | Time (s) |
|---|---|---|---|---|
| 1 | 520 | 518. 6 | 1. 4 | 50 |
| 2 | 520 | 511. 5 | 8. 5 | 52 |
| 3 | 520 | 516. 3 | 3. 7 | 65 |
| 4 | 520 | 514. 0 | 6. 0 | 51 |
| 5 | 520 | 511. 9 | 8. 1 | 50 |
| 6 | 520 | 515. 8 | 4. 2 | 49 |
| 7 | 520 | 517. 4 | 2. 6 | 52 |
| 8 | 520 | 523. 5 | 3. 5 | 50 |
| 9 | 520 | 518. 8 | 1. 2 | 53 |
| 10 | 520 | 522. 8 | 2. 8 | 63 |

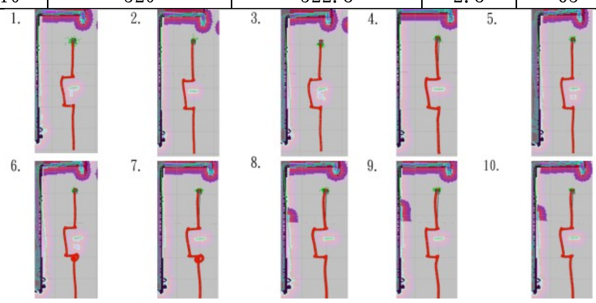

Fig. 16. The travel path of method 2 、odom+uwb fusion positioning

TABLE III. The final distance error of method 2 、odom+uwb fusion positioning

| Item | Final distance estimate(cm) | Final distance measure(cm) | Error (cm) | Time (s) |
|---|---|---|---|---|
| 1 | 520 | 508. 3 | 11. 7 | 52 |
| 2 | 520 | 514. 2 | 5. 8 | 49 |
| 3 | 520 | 506. 4 | 13. 6 | 50 |
| 4 | 520 | 510. 6 | 9. 4 | 50 |
| 5 | 520 | 506. 1 | 13. 9 | 49 |
| 6 | 520 | 509. 4 | 10. 6 | 63 |
| 7 | 520 | 510. 3 | 9. 7 | 60 |
| 8 | 520 | 509. 1 | 10. 9 | 52 |
| 9 | 520 | 512. 2 | 7. 8 | 51 |
| 10 | 520 | 505. 6 | 14. 4 | 53 |

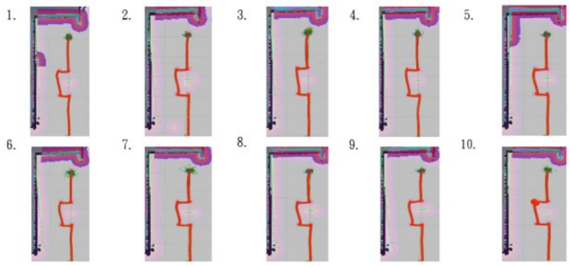


Fig. 17. The travel path of method 3、odom+imu+uwb fusion positioning

TABLE IV. The final distance error of method 3、odom+imu+uwb fusion positioning

| Item | Final distance estimate(cm) | Final distance measure(cm) | Error (cm) | Time (s) |
|---|---|---|---|---|
| 1 | 520 | 523.6 | 3.6 | 50 |
| 2 | 520 | 511.2 | 8.8 | 48 |
| 3 | 520 | 518.4 | 1.6 | 50 |
| 4 | 520 | 522.7 | 2.7 | 49 |
| 5 | 520 | 510.9 | 9.1 | 49 |
| 6 | 520 | 527.2 | 7.2 | 53 |
| 7 | 520 | 517.9 | 2.1 | 51 |
| 8 | 520 | 512.4 | 7.6 | 51 |
| 9 | 520 | 514.7 | 5.3 | 52 |
| 10 | 520 | 525.8 | 5.8 | 53 |

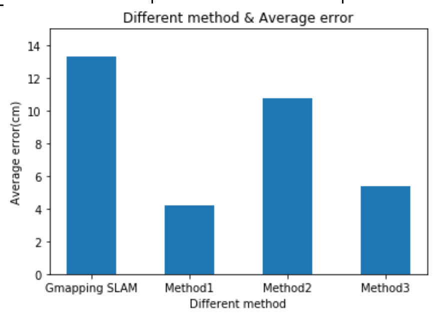


Fig. 18. Comparison of different methods on the average error

TABLE V. Comparison of different methods on the average error

| Method | Average error of final distance(cm) |
|---|---|
| Gmapping SLAM | 13.33 |
| Method 1<br>odom+imu fusion positioning | 4.2 |
| Method 2<br>odom+uwb fusion positioning | 10.78 |
| Method 3<br>odom+imu+uwb fusion positioning | 5.38 |

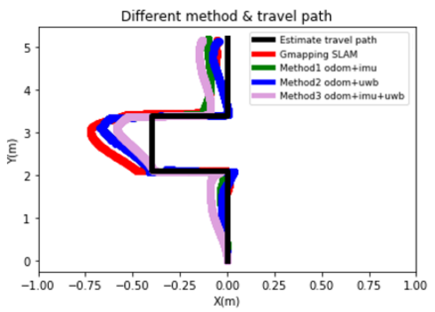


Fig. 19. Comparison of the maximun moving distance of different methods

TABLE VI. Comparison of the maximun moving distance of different methods

| Method | Average error of final distance(cm) |
|---|---|
| Gmapping SLAM | 28.2 |
| Method 1<br>odom+imu fusion positioning | 25.7 |
| Method 2<br>odom+uwb fusion positioning | 25.3 |
| Method 3<br>odom+imu+uwb fusion positioning | 23.6 |

In Fig. 14 to Fig. 17, Table 1 to Table 4 show the travel path and the final distance error data for each method. Finally, in Fig. 18 and Table 5, we can see the comparison of different methods on the average error. The method 2、odom+imu fusion positioning is the best in all method. Average error of final distance is only 4.2cm. In Fig. 19 and Table 6 show the comparison of different methods on the maximum move distance. The method 3、odom+imu+uwb fusion positioning is the best in all method. It means that the robot can walk more accurately and it can pass smoothly when it is surrounded by many obstacles.

## VI. Conclusion

We use the Extended Kalman Filter to fuse multiple sensor information to obtain accurate positioning and use it in SLAM and path planning.The experimental results prove that our multi-sensor fusion method is feasible, and the positioning error can be controlled within 10cm.

## References

[1] Giorgio Grisetti; Cyrill Stachniss; Wolfram Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters", IEEE Transactions on Robotics, Vol. 23, 2007, pp. 34-46.

[2] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Oskar von Stryk, and Uwe Klingauf, "Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots", RoboCup 2013; Robot World Cup XVII, 2014, pp. 624-631.

[3] Wolfgang Hess, Damon Kohler, Holger Rapp, Daniel Andor, "Real-time loop closure in 2D LIDAR SLAM", IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 1271-1278.

[4] Shao-Hung Chan,Ping-Tsang Wu,Li-Chen Fu, "Robust 2D Indoor Localization through Laser SLAM and Visual SLAM Fusion" 2018 IEEE International Conference on Systems, Man, and Cybernetics.

[5] Bekir Bostanci, Sercan Tekkok, Emre Soyunmez, Pinar Oguz-Ekim, Faezeh Yeganli, "The LiDAR and UWB based Source Localization and Initialization Algorithms for Autonomous Robotic Systems" 2019 11th International Conference on Electrical and Electronics Engineering (ELECO), Bursa, Turkey.

[6] Lukasz Zwirello, Xuyang Li, Thomas Zwick, Christian Ascher, Sebastian Werling, Gert F. Trommer, "Sensor data fusion in UWB-supported inertial navigation systems for indoor navigation", 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany.

[7] Teresa Concei¸c˜ao, Filipe Neves dos Santos, Paulo Costa, Ant´onio Paulo Moreira, "Robot Localization System in a Hard Outdoor Environment", ROBOT 2017: Third Iberian Robotics Conference, pp 215-227

[8] Espinosa F, Santos C, Marrón-Romera M, Pizarro D, Valdés F, Dongil J, "Odometry and laser scanner fusion based on a discrete extended Kalman Filter for robotic platooning guidance", Sensors, Vol. 11, 2011 , pp. 8339-8357.