Proceedings of 2015 IEEE
International Conference on Mechatronics and Automation
August 2 - 5, Beijing, China

Design and Implementation of an ROS based Autonomous Navigation System

Qunshan Xu

Department of Automation
University of Science and Technology of China
Hefei, Anhui Province,230027,China

xqshan@mail.ustc.edu.cn

Jianghai ZHAO, Chunxia Zhang and Feng He

Laboratory of Robotic system
Institute of Advanced Manufacturing Technology,
Hefei Institute of Physical Science, CAS, Changzhou,
Jiangsu Province, 213164,China

jhzhao@iamt.ac.cn

***Abstract* - The presented work describes how to implement an autonomous navigation system on wheeled mobile robots based on Robot Operating System (ROS). ROS contains many reusable software stacks and we have analyzed location issues, map building and autonomous navigation packages in this work. With regard to our own designed robot, the hardware and software architectures are presented. Besides, robotic kinematics of motion has been discussed for the four-wheel differential drive mechanism of our robot. An inertial measurement unit is used to assist in providing a more accurate odometry model and precisely localizing robot itself within the world during the process of navigation. Experimental results carried out with a service robot are shown in this paper to demonstrate the performance of the designed system.**

Index Terms* - *Autonomous mobile robots, navigation, robot operating system, odometry model.

I. Introduction

Robotics is a rapidly growing subject which comprises profound scientific theories and has long appealed to many researchers. With the fast development of computer technique, a series of robots have been designed and produced to meet some specific needs in different domains around the world. Using robust robot control systems to achieve some tedious or dangerous tasks instead of human beings has got vast applied prospects especially in complex industrial environments. Human beings can acquire information around themselves through their inherent senses, while a robot has no ability to act like them unless some useful sensors are mounted on its body. These sensors are employed by a robot to sense the surroundings incorporate laser range finders (LRF), inertial measurement units (IMU), global positioning system (GPS), cameras, sonar, and so on.

In general, wheeled and legged patterns of locomotion are widely utilized for mobile robots. Compared with legged locomotion which requires higher degrees of freedom and greater mechanical complexity, wheeled locomotion are extremely well suited to flat ground [1]. A review of various control architectures for autonomous navigation of mobile robots has been presented by Nakhaeinia et al. [2]. In this work, robotic control architectures were classified into three categories: Deliberative navigation, Reactive navigation and hybrid navigation. The authors made a detailed analysis and discussion on the properties of significance, advantages and drawbacks for the three categories. Then they summarized that hybrid scheme has the best performing supervisory control architecture and can be applied to dealing with unknown, dynamic navigation problems promisingly [2].

For the purpose of autonomous navigation, a solution to the simultaneous Localization and Mapping (SLAM) problem is seen as a prerequisite for mobile robots [3]. Mallios et al. [4] proposed a pose-based algorithm to solve the SLAM problem for an autonomous underwater vehicle. The robot motion was estimated using an Extended Kalman Filter (EKF) method and the range scans gathered from a sonar. Another efficient SLAM solution was proposed in [5]. optimized Rao-Blackwellized particle filters, which drastically increased the performance than before when applied to SLAM, has been proposed and validated by means of adaptive proposals and selective resampling technique. After this step, the robot needs to find a proper way to go to the desired position. Cherubini et al. [6] presented a framework for avoiding both static and moving obstacles which uses an on-board LRF during visual navigation of a wheeled mobile robot. Information from range and vision sensors are frequently used in autonomous navigation.

Robot Operating System (ROS) is an open source software framework primarily based on Unix platform for operating robots. It is broadly employed for robotics research in recent years and a comprehensive overview of ROS has been presented by Quigley et al. [7]. ROS provides a great number of packages and stacks that are ready for developing robot software system, including basic motion control, cooperation between multiple robots and navigation module. Many developers and researchers have shared their knowledge and achievements in ROS community and these contributions dramatically accelerate the progress of robotics. In this paper, we presented a design and implementation of an ROS based autonomous navigation system for an intelligent service robot.

The rest of this paper is organized as follows. In section II, we introduced the system architecture that is designed for our robot. The analysis and implementation of navigation system is clearly addressed in section III and some experimental results are given in section IV to demonstrate the performance of the designed navigation system. Finally, a concise conclusion and future extension are proposed in section V.

II. System Architecture

A. Hardware devices

In order to accomplish the mission of autonomous movement, we have designed and analyzed the hardware components for this project. The experiment was carried out on a wheeled mobile robot named YFW robot, as shown in Fig. 1. YFW uses a pair of steered wheels and each pair contains two wheels which provides him with the ability to endure a heavier load. In addition, four castors are installed to keep the robot upright. We have equipped the robot with a Linux (Ubuntu 12.04) based computer that is used to run ROS system for operating. Multiple sensors such as four rotary odometry encodes, a LRF (Hokuyo URG-04LX), an IMU (Xsens MTi-30) are also used to make the robot capable of moving and localization precisely on the ground. As shown in Fig. 2, the onboard computer is connected to the LRF, IMU, and ARM development board which communicates with four motor drivers through CAN bus. A Wireless Local Area Network (WLAN) is also provided for cooperating programs among multiple computers. One merit of ROS is handily view the same set of topics, services and parameters for several machines [8]. Consequently, we can manually operate the robot from our personal computer or a desktop one via the wireless network.

B. *Software system establishment*

The basic structure of YFW robot control system is based on ROS. Like many other operating system, ROS provides an Integrated Development Environment for researchers to develop various robot software. It comprises a large number of reusable software packages which can be used in kinds of fields due to its open sourced property. The presented software system was shown in Fig. 3. Once a goal contains position and orientation is given within the map in *human-computer interface* or *rviz,* the navigation system starts to plan a suitable path from the beginning to end. *Twist* messages include linear and angular velocities are published to base controller for autonomous motion. The robot takes environmental messages from range sensor to prevent potential collisions and nearly stops at the desired position after finishing the navigation process.

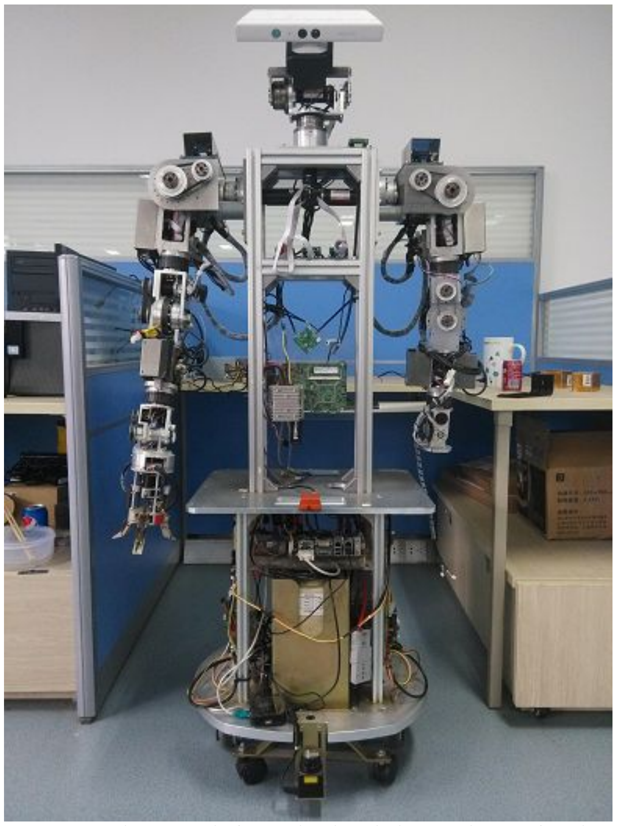
Fig.1. YFW robot: a platform developed by Institute of Advanced Manufacturing Technology, Chinese Academy of Science.

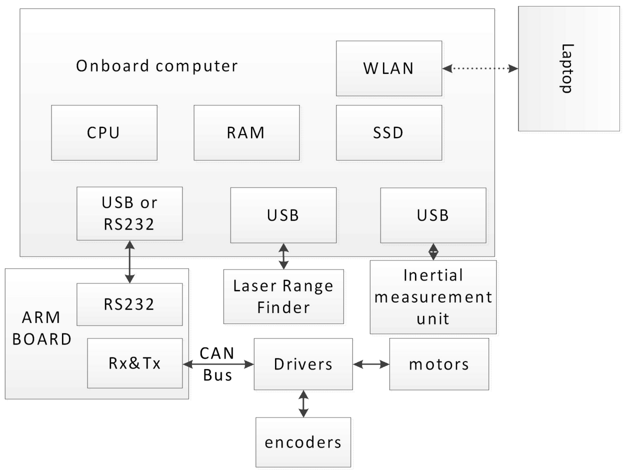

Fig.2 hardware devices of YFW robot

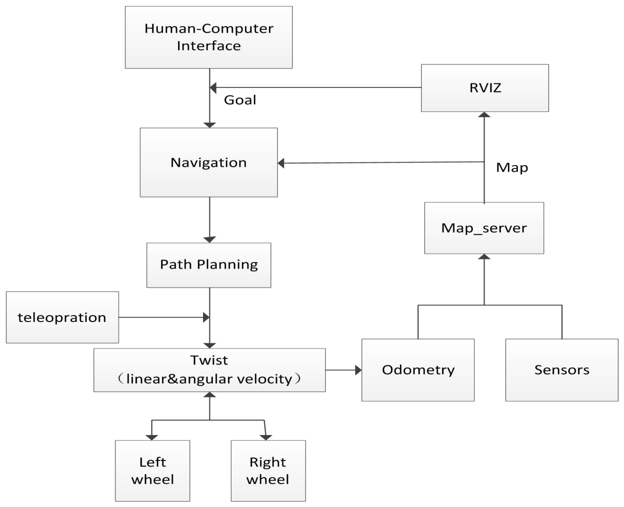

Fig.3. Software structure of navigation system

III. Analysis and Implementation of Navigation

The framework of hardware and software components has been presented above. we start to analyze and implement the autonomous navigation developed for YFW robot. The software architecture is based on ROS and the detailed procedures are presented gradually from base to top as noted below.

A. Base Control Layer

An ARM board, four motor drivers and encoders are used in the lowest control layer. The ARM board is responsible for the communication between onboard computer and drivers. An encoder registers the number of ticks per revolution of the corresponding wheel [8]. The speed and distance traveled of each wheel are able to know from encoder data and can be converted to robot itself on the condition of obtaining the diameter of wheels and the distance between them. These internal data provides robot with information of movement is known as odometry. The mechanical structure of our robot is shown in Fig. 4 and the odometry data can be calculated from robot kinematics.

Let L_2, L_1, R_1, R_2 separately denote four wheels with respect to the robot from left to right. The distance of wheel R_1 moves in a very short time is ΔS_{r1} and the same to others. The radius of each wheel is r . The distance between two adjacent wheels is a , and s is half length of the wheel axis. The relationship between rotation rate of wheels and linear speed *v,* angular speed *w* of robot is calculated as follows.

$$v_{l1} = v - w * s \tag{1}$$

$$v_{l2} = v - w * (s + a) \tag{2}$$

$$v_{r1} = v + w * s \tag{3}$$

$$v_{r2} = v + w * (s + a) \tag{4}$$

As shown in Fig. 5, a common center of rotation called Instantaneous Center of Curvature (ICR) exists in the motion of the robot. When the robot moves a very slight distance, the changes of position and angle are denoted by $\Delta\theta$ and ΔS respectively. Suppose the robot standing in the pose (X, Y, θ) at time *t* in the global coordinate system, then the new robot pose (X', Y', θ') at the next moment can be calculated by

$$\begin{bmatrix} X' \\ Y' \\ \theta' \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta S \cos(\theta + \Delta\theta/2) \\ \Delta S \sin(\theta + \Delta\theta/2) \\ \Delta\theta \end{bmatrix} \tag{5}$$

Where

$$\Delta S = (\Delta S_{l1} + \Delta S_{l2} + \Delta S_{r1} + \Delta S_{r2})/4 \tag{6}$$

$$\Delta\theta = (\Delta S_{r1} + \Delta S_{r2} - \Delta S_{l1} - \Delta S_{l2})/(4*s + 2*a) \tag{7}$$

From the presented formulas above, an estimated position and orientation of the robot are acquired. And we also provide a solution with the problem of controlling the robot to reach a given velocity. It is an essential information and used to be stored as a message named *nav_msgs/Odometry* in ROS. We need to write a program that involves carefully translating the information which comes from ARM board to this message using the mathematical expressions of odometry model.

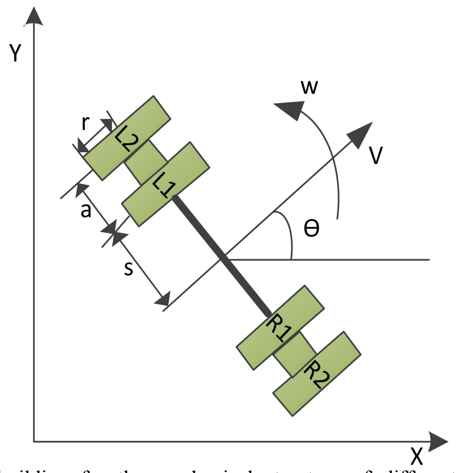

Fig.4. Model building for the mechanical structure of differential driver wheels

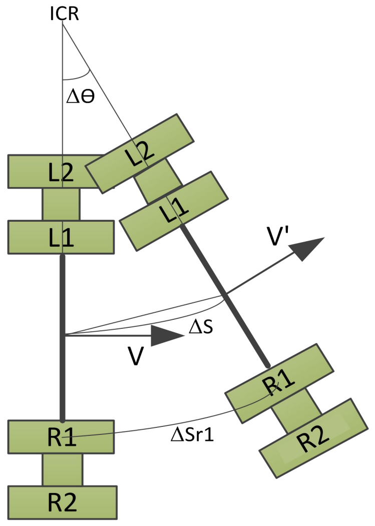

Fig. 5. Analysis of the robot kinematics of motion

B. Simultaneous localization and mapping

After the base control section is accomplished, the robot can move forward and rotate in place when we published appropriate orders for him. Linear speed and angular speed are both considered in this condition. This is also a prerequisite for robot proceeding any further. Then a map must build to make the robot has a clear understanding of the environment around. SLAM techniques happens to solve this problem. when a map is not available or highly uncertain in some applications, the robot can use the measurements provided by internal sensors to build a map and localize itself within the map at the same time [9]. Many solutions about SLAM problem have been presented in the past few decades and *Bayes* theorem was widely used among these different methods. A comparison of some common methods composed of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM algorithms is presented in [10]. The authors compared the localization and mapping accuracy of these methods and concluded their merits and drawbacks from several experimental results.

Slam_gmapping and *hector_slam* are mainly used in dealing with the SLAM problem and both of them are viable options that have been testified to work well in different environments. As for these two methods that based on laser scan, the former requires odometry messages to perform while the other does not. SLAM in different scenarios using *slam_gmapping* and *hector_slam* are shown in Fig. 6.

C. *Navigation stack*

It is uncomplicated for navigation stack on a conceptual level in ROS. Environmental information and internal data of the robot are conveyed to navigation section and outputs velocity commands are sent to the mobile base. Necessary configurations are described below.

1) Transform configuration: There are many different coordinate frames need to be described in the robot navigation system such as the location of robot mobile base and various sensors mounted on top of it. Thus, a transform tree is required to define the relevance between these coordinate frames generally using *tf* software library in ROS. As far as we are concerned, the relationship between the center point of the robot base and the sensors is fairly important. In order to help the robot avoid obstacles in the world successfully, information received from laser scanning need to be transformed into robot base precisely. And this transform is rely on the relationship we have built from their positions. The range and orientation of LRF and IMU are specified to the center of mobile base in our transform configuration.

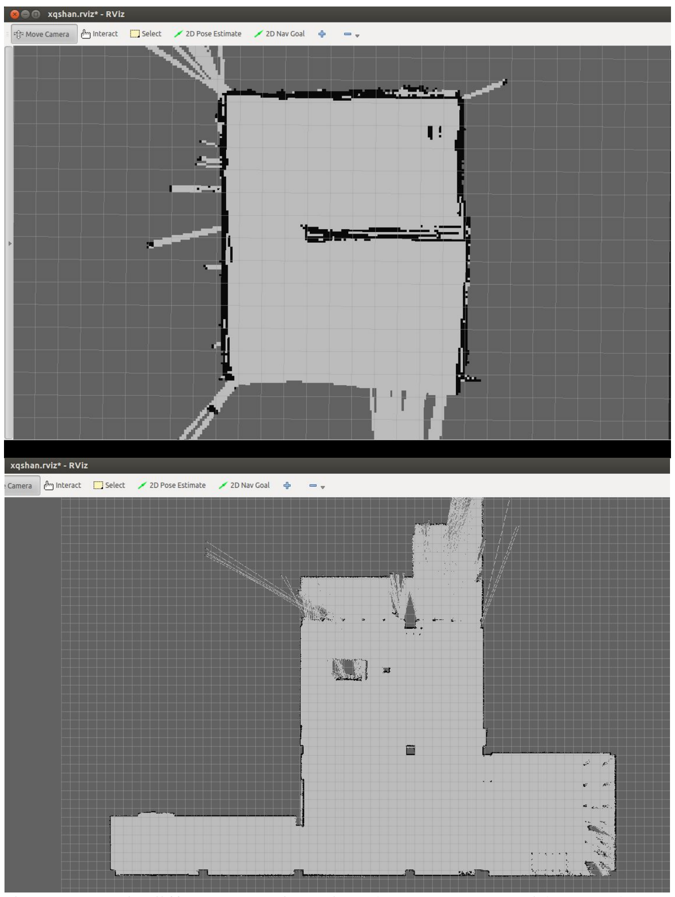

Fig. 6. SLAM in different scenarios using *slam_gmapping* and *hector_slam* respectively.

2) Sensors information: The navigation stack demands information from sensors for robot to localize itself and avoid obstacles in the world. ROS assumes that the messages published from range sensors have the types like *LaserScan* or *PointCloud.* In our design, laser device is mainly used to perceive the position and direction of barriers and then publishes these messages. However, no matter how reliable the odometry data we have calculated, errors still exist due to systematic and non-systematic reasons such as the variations in wheel diameters and inevitable slippage of wheels. An effective approach is employed in our work through the fusion of wheel encodes and IMU sensor which can be used to detect changes in yaw attribute of the robot. It is proved to be a useful method in global localization and navigation of our robot from the experimental results.

3) Autonomous navigation: When the environmental map is created, autonomous navigation can be achieved after a series of configuration. ROS contains a package named *map_server* which can convert map data into a service. *Amcl* is an implementation of adaptive Monte Carlo localization approach and the node enables the robot to localize itself in the world. Besides, a package called *move_base* is used to plan collisionless paths and send motion messages to the base during the procedure of navigation. As a powerful GUI node in ROS, *rviz* allows users to view the whole process for the navigation task.

IV. EXPERIMENT

In this section, the demonstration of the autonomous navigation system for our robot based on ROS has been presented. These trials are taken place in real environment at our Robot Lab. Relatively accurate to reach an assigned destination while not crashing anything it will meet is the mission of YFW robot. Three steps have been taken in our experiment.

A. Calibrating odometry model

we manually operated the robot to move straight and turn around by sending *twist* messages from a starting location. After traveling a proper distance, we need to dominate the robot returning from where it have been away. In addition, we also move the robot 2.0 meters forward and rotate it 360 degrees in place. Ultimately, we can compare these original data which calculated by odometry model with actual measurement data from these operations. Correction factors can be set in odometry model for more precise localization of our robot through these calibration routines.

B. Mapping and localization

Slam_gmapping method is applied in our map building procedure for the experimental environment and the robot is executing SLAM algorithm as shown in Fig. 7. The red lines are current obstacles scanned by laser. The light blue arrows denote the actual moving trajectory of robot. The robot and its initial location are denoted by different coordinate frames. The whole map of the robot lab constructed by YFW robot is shown in Fig. 8.

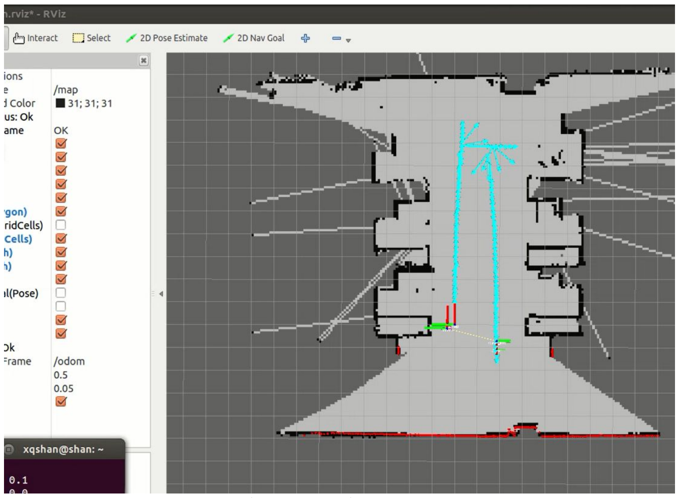

Fig. 7. Mapping process

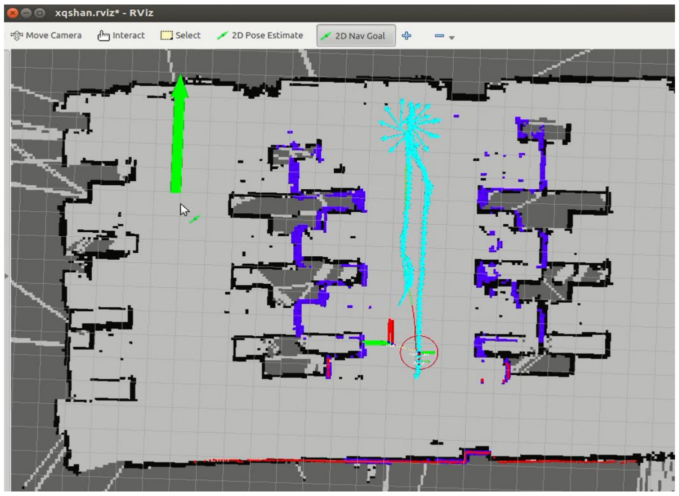

Fig. 9. set a target posture

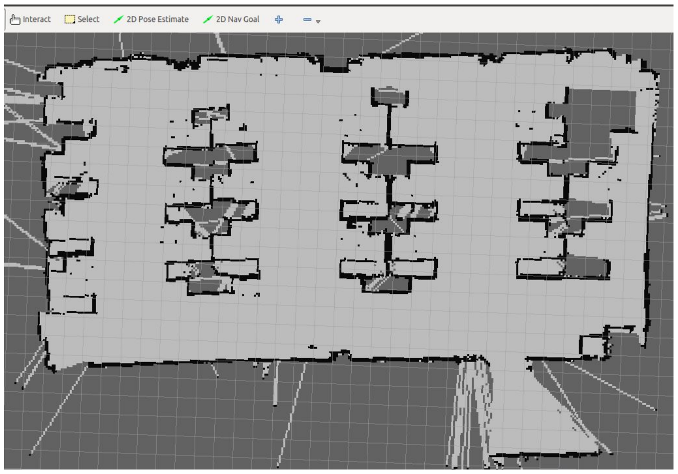

Fig. 8. Whole map of Robot Lab

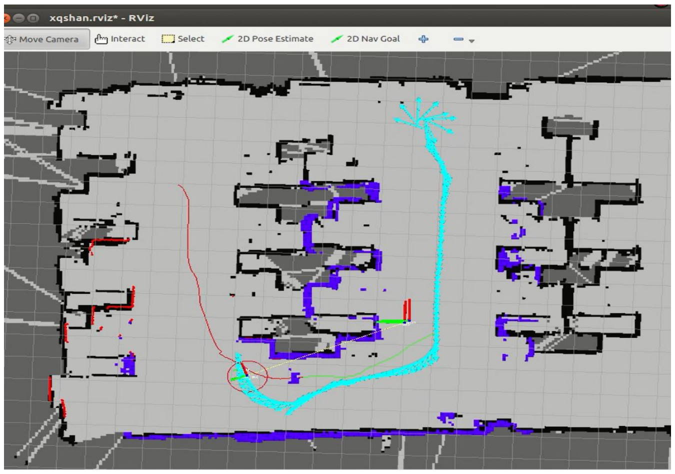

Fig. 10. Path planning and obstacles avoiding

C. Autonomous navigation

When we have built the map, we are able to command the robot to go to any positions with the ability of obstacle avoidance withing the known environment. The procedure of path planning is based on A* search algorithm and it provides global planner and local planner of robotic trajectory. We can point to set a goal on map as shown in Fig. 9, then the navigation system starts to calculate a optimal path and send velocity orders to the base tracking the global path. The local path represents the actual moving trajectory of robot which is constantly tuning to avoid changing obstacles until the target position is achieved. The red circle denotes the YFW robot and the deep blue pieces are obstacles detected by laser. As shown in Fig. 10 and Fig. 11, the robot promptly adjusts its previous route when a person suddenly appeared at the front of itself. We can see that the robot turns left to skirt the person and then keep moving to the destination.

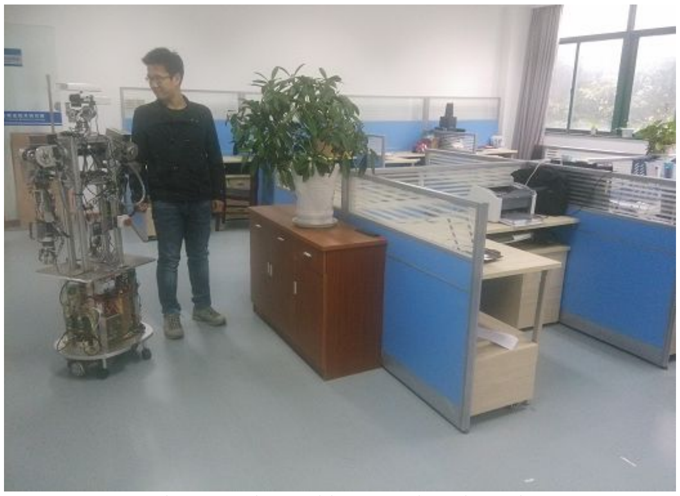

Fig. 11. Robot avoid a person in Robot Lab

V. Conclusion And Future Works

In this paper, we have analyzed and implemented an ROS based autonomous navigation system on our YFW robot. Both the robot software system and hardware architecture are described in detail. Base control layer, SLAM methods and path planning are primarily introduced in the navigation process. The robot can build a map of an unknown environment and receive instructions to automatically reach a given navigation goal while evading any collisions.

The presented work can basically achieve the autonomous navigation while the precision of robot to get to the goal posture still remains to be further improved. The future work will focus on fusion of laser and vision sensors in robot navigation for acquiring a better effect.

ACKNOWLEDGMENT

This work was supported by Natural Science Foundation of Jiangsu Province (Grant NO.BK2012587) and Supported by Science and Technology Support Program of Jiangsu Province (Grant NO.BE2012057). We would also like to thank our colleges in IAMT for their technical guidance and help.

REFERENCES

[1] R. Siegwart, I. Nourbakhsh, D. Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2004.

[2] D. Nakhaeinia, S.H. Tang, S.B. Mohd Noor and O. Motlagh. "A review of control architectures for autonomous navigation of mobile robots," International Journal of the Physical Sciences, vol. 6, no. 2, pp. 169-174, January 2011.

[3] H. Durrant-Whyte, T. Bailey. "Simultaneous localization and mapping: part I," IEEE Robotics and Automation Magazine, vol. 13, no. 2, pp. 99-110, 2006.

[4] A. Mallios, P. Ridao, D. Ribas, F. Maurelli, and Y. Petillot, "EKF-SLAM for AUV navigation under probabilistic sonar scan-matching," The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, pp. 4404-4411, October 18-22, 2010.

[5] G. Grisetti, C. Stachniss, W. Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters," IEEE Transactions on Robotics, vol. 23, no.1, pp. 34-46, February 2007.

[6] A. Cherubini, F. Spindler, and F. Chaumette. "Autonomous Visual Navigation and Laser-Based Moving Obstacle Avoidance," IEEE Transactions on Intelligent Transportation System, vol.15, no.5, pp. 2101-2110, October 2014.

[7] M. Quigley, et al., "ROS: an open-source Robot Operating System," in Open-source Software Workshop of the International Conference on Robotics and Automation, vol. 3, no. 3.2, 2009.

[8] R. Patrick Goebel. *ROS By Example* ,volume 1, Version 1.03 For ROS Groovy, April 2013

[9] A. Garulli, A. Giannitrapani, A. Rossi, A. Vicino, "Mobile robot SLAM for line-based environment representation," Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, pp. 2041-2046, December 12-15, 2005

[10]Z. Kurt-Yavuz and S. Yavuz. "A comparison of EKF, UKF, FastSLAM2. 0, and UKF-based FastSLAM algorithms," Proceedings of the 16th IEEE International Conference on Intelligent Engineering Systems, Lisbon, Portugal, pp. 37-43. June 13-15, 2012