An Approach to Design Navigation System for Omnidirectional Mobile Robot Based on ROS

Hiep Do Quang
University of Economics-Technology for Industries, Hanoi, Vietnam
Email: dqhiep@uneti.edu.vn

Tien Ngo Manh, Cuong Nguyen Manh, Dung Pham Tien, and Manh Tran Van
Institute of Physics, Vietnam Academy of Science and Technology, Hanoi, Vietnam
Email: {tieniop, manhcuong3131.ng, phamdung210597, manhtran4321}@gmail.com

Kiem Nguyen Tien and Duy Nguyen Duc
Hanoi University of Industry, Hanoi, Vietnam
Email: duyndfx01662@funix.edu.vn

***Abstract*—The paper presents the Simultaneous Localization and Mapping, and implement a path-planning for the movement of Omni-directional self-driving robots based on the navigation stack. The virtual environment consists of known static obstacles and unknown dynamic obstacles, the mobile robot is required to achieve both local obstacle-avoidance and follow the global path during the moving process. All tasks have been performed on a four-wheeled Omnidirectional robot with Jetson TX2 high-performance processor for central processing tasks, an Astra depth camera and a RPlidar sensor. The achieved results show the efficiency, research direction of using Robot Operating System for controlling and monitoring autonomous robots, self-driving cars as well as developing intelligent robot systems.**

***Index Terms*—Robot Operating System (ROS), GAZEBO, RVIZ, simultaneous localization and mapping (SLAM), omni robot, Nav_core, navigation**

I. Introduction

Nowadays, Robotics is widely used in not only heavy industries but also human life. Owing to the ability of transportation, autonomous robots are utilized to take the place of human workers in many manufactories where the robots hoist and transfer the goods on production lines. Therefore, autonomous robots are expected to be able to take more flexible as well as perform more tasks required by people. To achieve these tasks, the conventional wheel is replaced by the Omni wheel that can move simultaneously and independently in rotation and movement in the flat surface. There are two types of Omni robots that are usually used namely three-wheeled directional Omni robot and four-wheeled directional Omni robot. In this paper, the model of the four-wheeled robot will be presented [1].

In recent years, the issues of cooperative motion control of the Omni robot have significantly attracted the researchers [2] that leads to a dramatic increase in computational issues and hardware abstractions are inevitable due to the more complexity of tasks required for robot applications. With the aim of addressing these challenges, ROS (Robot operating system) is considered as a software environment in robotics disseminated to facilitate the projects' robot [3]. The concept of ROS goes far beyond just a framework. It is a Linux based meta-operating system, which is responsible for synchronizing the parts of the robot together, collecting data to design localized maps. With powerful tools and libraries integrated in ROS, users are able to conveniently implement the project's robot for obtaining, building, writing, and running code across computers. Therefore, it is used widely in designing control systems for many kinds of objects like self-driving cars, autonomous robots that had been shown in [4], [5].

Simultaneous localization and Mapping (SLAM) algorithm may play an integral role in navigation for robots and become an increasingly hot topic in robot applications. This is a method allowing to create an incremental map and acknowledge obstacles surroundings in unknown environments in [6], [7], and [8]. By gathering data of IMU or encoder through ROS [9], the location of the robot could be estimated successfully with no expensive sensor and the mobile robot generates an incremental map of the environment.

Ros has integrated Gazebo and Rviz software to enable simulation and monitoring of robot operation. They also play essential tools for implementing the navigation planner on the autonomous robot. The task of path planning for a mobile robot is received considerable attention from many researchers [10], [11]. Some studies in this realm have been focused on dynamic obstacle-avoidance under complicated circumstances, which includes avoidance of both static and dynamic obstacles [12].

Inspired by the aforementioned studies, the objective of this paper focuses on implementing Slam to recognize

Manuscript received January 29, 2020; revised October 5, 2020.

doi: 10.18178/ijmerr.9.11.1502-1508

the robots' positions and environment conditions in the unknown area as well as performs autonomous navigation with the Omni mobile robot on the processor Jetson TX2. The rest of this paper is organized as follows. Section II proposes the Slam algorithm for the Omni robot. After that, software architecture for robot control is constructed based on ROS to implement a navigation planner on the Omni robot in section III. The results in the simulation and the practical environment are analyzed in section IV. Finally, a conclusion is mentioned in section V.

II. SLAM FOR THE OMNI ROBOT

In the motion control design for a mobile robot, there are more modern ways to perform effective movements that require the optimization in the robot trajectory to reach the goal, the obstacles avoidance, and the smooth operation. In this case, simultaneously localization and mapping is an outstanding method to overcome the problem which is the information shortage about the operating environment. Slam algorithms are used for the robot when it performs the movement for the first time in an unknown environment extracted based on data about the surrounding generated from some sensors such as RP lidar and depth camera. To achieve this target, Many algorithms have been developed for the purpose of extracting a map facilitating the accurate motion of the robot Currently, Gmapping package is widespread use to perform the Slam task either in indoor or outdoor environments. The technique consists of building a map of an environment and at the same time using this map to deduce its location. When this process of gathering information from environment through sensors and generating a map is completed, the generated map can be saved to use in the Navigation stack.

Therefore, to implement the Slam algorithm, Gmapping stack is used to draw an incremental map for Omni robot, see Fig. 1:

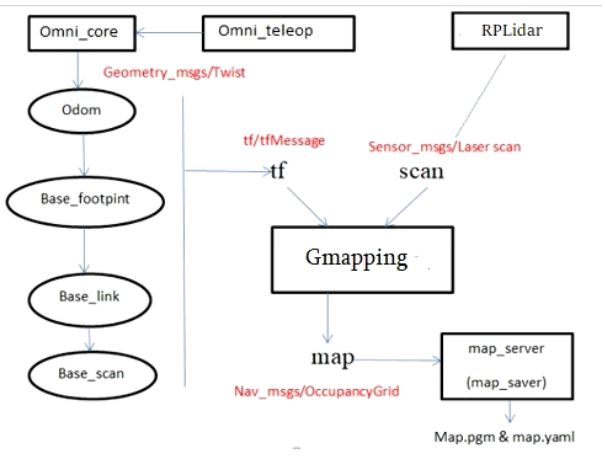

Figure 1. Scheme for Slam with the Omnidirectional robot.

The Omni robot navigates in the real environment, a set of data is collected via sensors such as Astra camera and RPLidar. The Gmapping package estimates the position of the robot and builds the map given the robot's observations about the environment and its odometry measurements. For measuring the distance between the robot and unknown obstacles, the robot is equipped with a laser scanner RPLidar that can achieve 360-degree to take the relative locations between any individual landmark and the robot.

The TF library was designed to provide standards to keep track of coordinate frames and transform data within the entire system. Thus, the technician utilizing individual components is guaranteed that the desired data is in the coordinate structure frame without requiring knowledge of all the structures in the system.

The above figure depicts the SLAM task used for the Omni robot. Overall, this diagram is organized into three main parts: the robot part, the control part, and the SLAM part. The first part includes the transformation between the robot's link, and the most significant is the relationship between the robot's base coordinate: /Base_footprint, and the position of Rplidar: /Base_scan which captured the environmental data. The control part is designed to command the robot's movement through the keyboard: Omni_teleop. The final part plays a key role in performing the SLAM task with Gmapping is the center element. Additionally, the map_sever tool could be used to save the generated map in both the image file and the data file.

III. DESIGN NAVIGATION SYSTEM

A. Robot Controller Design

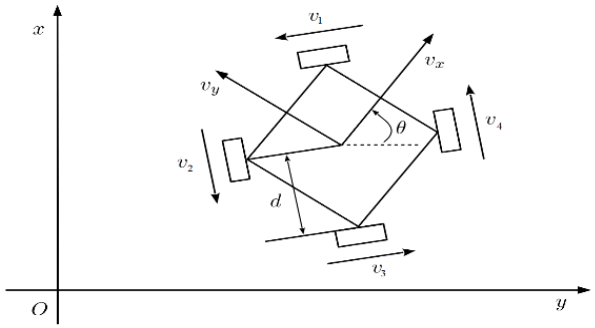

Figure 2. The Four-wheeled Omnidirectional robot.

The Omni robot has four wheels which are 90 degrees apart. Oxy represents the global coordinate axis, the distance between wheels and the robot center was defined by d. The robot movement would be identified for the navigation stack. As the global coordinate chosen in Fig. 2, it is obvious that the velocity of the robot contains three components: a linear velocity along Ox-axis and Oy-axis an angular velocity

The robot's coordinate vector is defined as $\underline{q} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$ and the velocity of the robot in the global coordinate could be obtained by taking the derivative of $\underline{q}$. To facilitate the robot controller design, the relationship between the velocity in the robot's axis and the velocity in the world's axis is described by the kinematic model of the robot:

$$\dot{\underline{q}} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \underline{v}. \quad (1)$$

From (1), we obtain the equations which would be used to program the robot's navigation in ROS:

$$\begin{cases} \dot{x} = v_x \cos\theta - v_y \sin\theta \\ \dot{y} = v_x \sin\theta + v_y \cos\theta \\ \dot{\theta} = w \end{cases} \quad (2)$$

where v_x, v_y, and $\dot{w}$ are the velocity in the robot's axis and the velocities are the control signal generated from the local planner in the following section. The robot position is directly controlled by these signals which are transformed into the desired signals for four wheels of the robot. The transformation formula is described as below:

$$\begin{cases} v_x = \dfrac{\sqrt{2}}{8}(-v_1 - v_2 + v_3 + v_4) \\ v_y = \dfrac{\sqrt{2}}{8}(v_1 - v_2 - v_3 + v_4) \\ \omega = \dfrac{(v_1 + v_2 + v_3 + v_4)}{4d} = r\dfrac{(\omega_1 + \omega_2 + \omega_3 + \omega_4)}{4d} \end{cases} \quad (3)$$

With the reference value transformed from the local planner, the velocity for each wheeled can be computed to ensure the robot to track the desired value.

B. Navigation

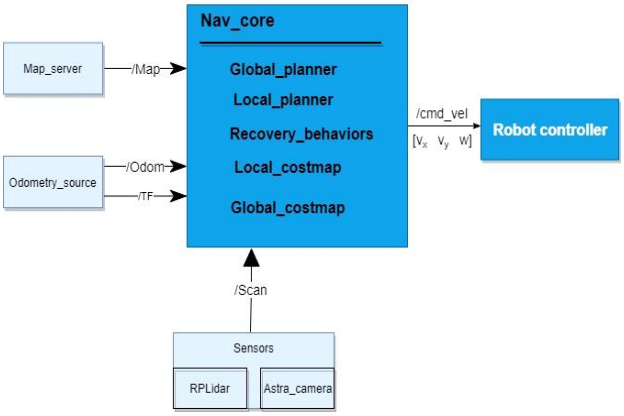

Figure 3. The structure of the navigation stack.

Mapping, localization and path planning are crucial navigation assignments for autonomous robots. In Ros, the navigation stack provided the adequate nodes and topics is powerful for a mobile robot to move from place to place reliably. The navigation stack produces a safe path for the robot to execute, by processing data from odometry, sensors and environment map. In order to navigate the Omni robot, the Gmapping stack is a localization system for building the map (Fig. 3), while nav_core containing the local planner and global planner is responsible for linking them to achieve the nav goal and deal with complex planning problem. The odometry publishes the Odom topics and the nav_core will take the velocity data from cmd_vel topic and assure the robot reproduces these velocities.

The global planner and local planner: The global planner takes the current position of the robot and the desired goal to generate the shortest path for robot navigation considered the obstacles from the static map. However, the actual path that the robot is that of the local planner, it can be considered the controller implements more tasks. The local planner incorporates current sensors readings to generate avoidance strategies for dynamic obstacles and creates the trajectories to follow the global path.

There are several types of available local planners, including the base local planner [13], the elastic band [14] and the time elastic band (TEB) [15]. In this paper, the selected method for local planning is Time Elastic Bands that consists of deforming the initial global plan based on the kinetic model of the Omni robot and generating the local path based on the dynamic obstacle in the global map.

The pose of the Omni robot is defined in the above section is $\underline{q} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T$. The TEB local planer requires the velocity and acceleration of the Omni robot, the security distance of the obstacles and the kinematic and dynamic constraints of the Omni robot that generates a set of commands for the velocity v_i and the angular velocity ω_i of each wheel as $cmd_i = \begin{bmatrix} v_i & w_i \end{bmatrix}^T$.

The costmap: The global costmap and local costmap are the topics containing the information about the obstacle and the path built for robot navigate. The costmap will be updated by using sensor data to maintain the information about the obstacle in the world, in which, the global costmap represents the whole environment while the local costmap is a scrolling window that is attached to the current position of robot and moves in the global costmap.

The global costmap displays data in a wide range which could be the whole or almost map. This costmap could be defined as the static map or the sliding window based on the purpose of map generation or robot navigation. By using the map generated by SLAM in the previous section, all the static obstacles of the environment are identified with the layer plugin. Moreover, in the navigation process of the robot, the unknown obstacles also recognized by the sensor data and additionally added to the map. To achieve the smooth movement and improve the navigation performance of the robot, in global costmap file, there are three layers would be declared:

- Static map: contains information about all the static obstacles and landmarks
- Obstacle: contains unknown obstacles data
- Inflation: is used to compute the secure radius around the obstacles to prevent robot colides them

On the other hand, the local costmap contains the parameters which affect the local planner to calculate the

control signal for the robot controller. In almost circumstances, this map plays the role as a dynamic sliding window following the robot, and identifies all the unknown obstacles as well as the dynamic obstacles which are not determined in the generated map. Furthermore, the information is used to directly generate the control signal to help the robot not only avoid the obstacles but also track to the global path generated by the global planner.

IV. Simulation Result

In order to evaluate the capability of the mobile robot, some tasks are performed in simulations by designing a robot simulator as well as an actual robotic platform.

A. The Omnidirectional Robot Model

The hardware architecture of the Omni robot is presented in Fig. 4, in which each module will perform some tasks in the Omni robot's operation.

Figure 4. The components of the Omni robot.

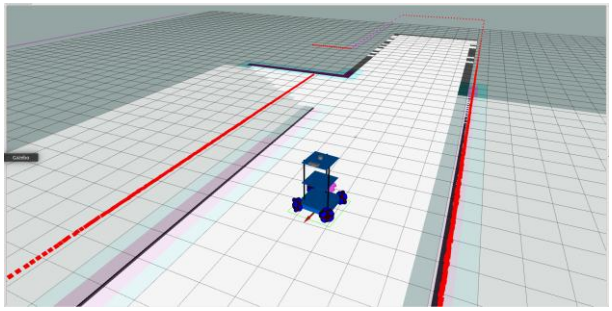

Figure 5. Robot model in Rviz.

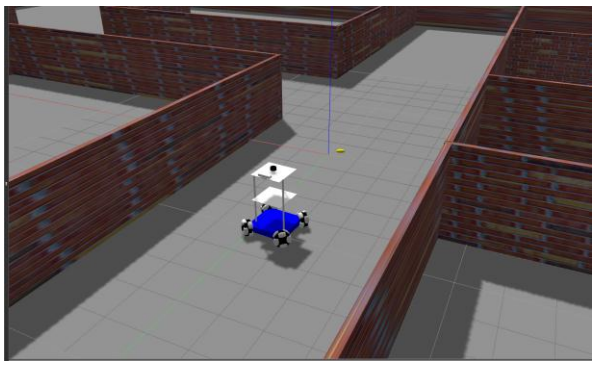

Figure 6. Robot model in Gazebo.

In the experiments, we implement some tasks in the Omni robot's operation on a Jetson-TX2 which supports Ubuntu. The embedded computer Jetson-TX2 directly processes information from a series of sensors including Astra camera, Lidar, and then transmit the command to a microcontroller. For capturing images from environment as well as measuring the distance between Omni robot and unknown obstacle, the Omni robot is equipped with Astra camera and RPLidar, in which, the Astra camera as the robot's eye comprises of RGB camera and depth camera while the RPLidar can perform 360-degree and the laser scanning range within 12m that produces map data used for mapping process. The STM32F103C8T6 control circuit will be the part that receives the control signal from Jetson TX2 and then directs the signal to the MOSFET bridge circuit to operate four motors.

Fig. 5 and Fig. 6 illustrate the robot model in Rviz and Gazebo.

The robot model is constructed using the URDF package, the Astra camera and the RPLidar are placed on the top of the robot and the Omni wheels are designed with some rollers arranged around the perimeter of the wheel. The RPLidar is installed on the roof of the platform in order to perform the Slam task for building a map.

B. Simulation Results

In this section, some simulations are conducted based on the powerful tool Gazebo. The visualization is performed by using Rviz to obtain the results (Fig. 7).

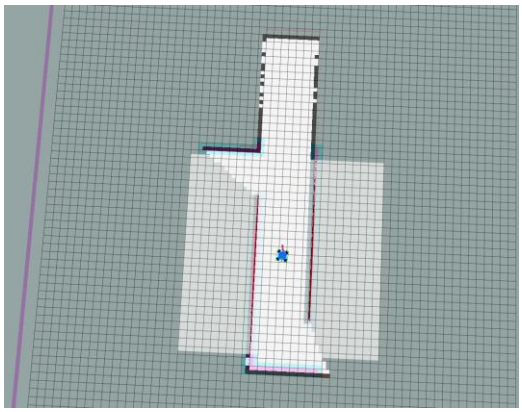

Figure 7. Slam vizualization in Rviz.

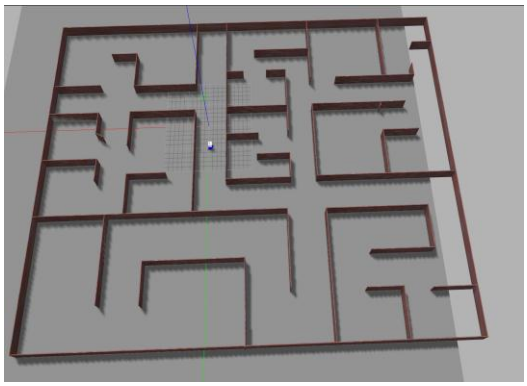

Figure 8. Visual map construction in Gazebo.

Fig. 8 shows the map built on Gazebo, the created map has strict walls and the Omni robot is controlled to move around the environment to obtain the necessary data which would be used to construct the map. The red lines are the laser scan signal generated from RPLidar and the current position of the robot is updated by using the odometry measurements.

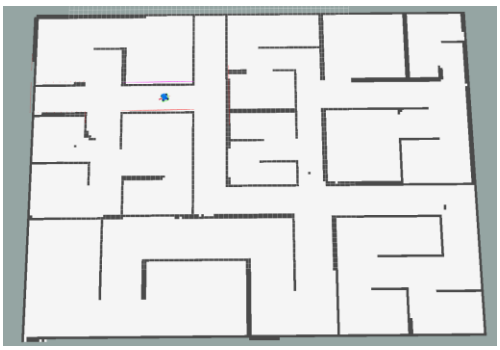

Figure 9. The generated map in Rviz.

In Fig. 9, the virtual map is exactly built and visualized in Rviz that is very similar to the created environment in Gazebo. After the map of the working area was generated by Gmapping package, the starting position of the Omni robot is determined on that map by utilizing a combination of the AMCL package and the odometry information.

Fig. 10 shows a fully autonomous navigation system, the path planning will determine the route for the Omni robot to follow in order to reach the desired final destination position.

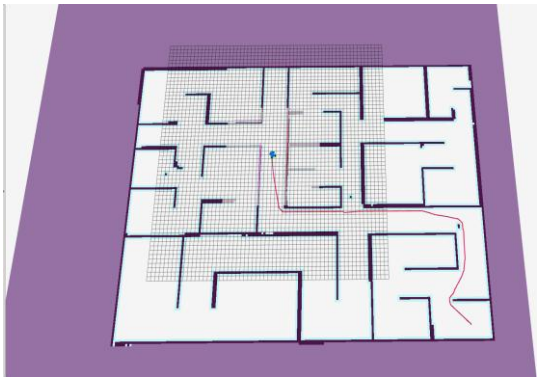

Figure 10. Navigation for Omni robot.

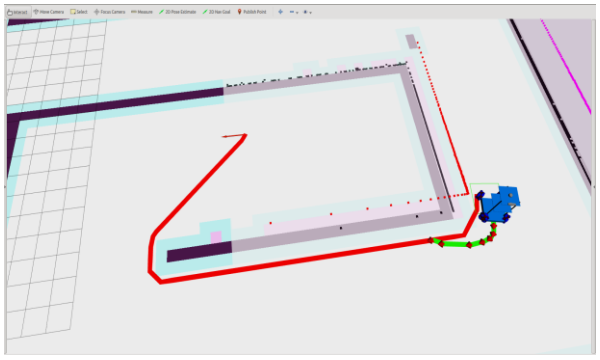

Figure 11. The obstacle avoidance.

The global planner is used to generate the shortest path to the goal, which is represented by the red line. Once a global plan has been generated, the local planner translates this path into velocity commands for the robot's motors.

Fig. 11 shows the environment with the detection of the obstacles which are the wall stricts in Gazebo. When the movement starts, the sensors update the local grid maps with the new obstacles in the field of view, the costmap uses the laser sensor information to create a local costmap in front of the Omni robot. The local planner publishes the local plan with a configurable lookahead distance and modifies the local path based on this distance, the selected lookahead distance is 0.5m. The local path generated by the local planner defined as a green line while the red line shows the global path of the robot's operation. As can be seen the Fig. 11, after avoiding the obstacle, the Omni robot can follow perfectly the desired path in a short time and move to the goal destination.

C. Practical Results

Fig. 12 shows the environment used for the practical test.

Figure 12. The practical environment.

Fig. 13 illustrates the navigation results when combining 2 sensors RPLidar and Astra camera on the real robotics platform.

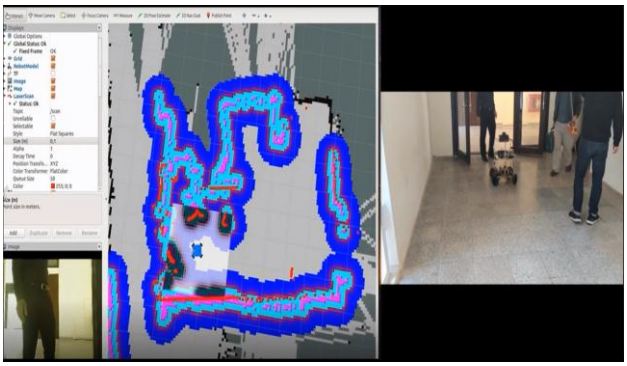

Figure 13. Navigation for Omni robot in practical map.

In this experiment, the robot shows the good performance of tracking the path generated by global planner and avoiding the obstacle that is a dynamic obstacle or a static obstacle which are not on the global map. Outer edges of the walls and people are successfully

recognized by the laser scanner and marked with the black dots and the lines shown in Fig. 13.

V. Conclusion

This paper has presented the construction of the four-wheeled Omnidirectional mobile robot. It shows the good efficient of Slam using Gmapping stack for building the 2D map. This paper also performs successfully the path planning for the Omnidirectional robot under the effect of both the static obstacle and the unknown dynamic obstacle due to the continuous update of the local path in the simulation and realistic environment. The tasks are based on the data generated from the 3D Astra camera and RPLidar. The robot hardware is also built to facilitate the integration of peripherals based on ROS. s. Moreover, the robot's operation could be monitored through the visualization tool.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Hiep Do Quang, Tien Ngo Manh, and Cuong Nguyen Manh designed the proposed method; Dung Pham Tien, Manh Tran Van, Kiem Nguyen Tien, and Duy Nguyen Duc conducted the experiment; Hiep Do Quang, Cuong Nguyen Manh, Dung Pham Tien, and Manh Tran Van wrote the paper.

Acknowledgement

This research was funded by Project "Research, Design And Manufacturing Smart Human-Form IVASTBot Robot Applied In Communication And Serving Human" coded VAST01.01/20-21 implemented by the Institute of Physics, Vietnam Academy of Science and Technology.

References

[1] E. Hashemi, M. G. Jadidi, and O. B. Babarsad, "Trajectory planning optimization with dynamic modeling of four wheeled omni-directional mobile robots," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation-(CIRA)*, 2009, pp. 272-277: IEEE.

[2] L. Xie, C. Scheifele, W. Xu, and K. A. Stol, "Heavy-duty omni-directional Mecanum-wheeled robot for autonomous navigation: System development and simulation realization," in *2015 IEEE International Conference on Mechatronics (ICM)*, 2015, pp. 256-261: IEEE.

[3] M. Galli, R. Barber, S. Garrido, and L. Moreno, "Path planning using Matlab-ROS integration applied to mobile robots," in *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2017, pp. 98-103: IEEE.

[4] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3, no. 3.2, p. 5: Kobe, Japan.

[5] I. Zamora, N. G. Lopez, V. M. Vilches, and A. H. Cordero, "Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo," *arXiv preprint arXiv:1608.05742*, 2016.

[6] S. Park and G. Lee, "Mapping and localization of cooperative robots by ROS and SLAM in unknown working area," in *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2017, pp. 858-861: IEEE.

[7] B. M. da Silva, R. S. Xavier, and L. M. Gonçalves, "Mapping and Navigation for Indoor Robots under ROS: An Experimental Analysis," *Creative Commons CC BY license*, 2019.

[8] Q. Lin *et al.*, "Indoor mapping using gmapping on embedded system," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017, pp. 2444-2449: IEEE.

[9] L. Zhi and M. Xuesong, "Navigation and Control System of Mobile Robot Based on ROS," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2018, pp. 368-372: IEEE.

[10] Z. Wu and L. Feng, "Obstacle prediction-based dynamic path planning for a mobile robot," *International Journal of Advancements in Computing Technology*, vol. 4, no. 3, pp. 118-124, 2012.

[11] G. Priyandoko, T. Ming, and M. Achmad, "Mapping of unknown industrial plant using ROS-based navigation mobile robot," in *IOP Conference Series: Materials Science and Engineering*, 2017, vol. 257, no. 1, p. 012088: IOP Publishing.

[12] P. Marin-Plaza, A. Hussein, D. Martin, and A. d. l. Escalera, "Global and local path planning study in a ros-based research platform for autonomous vehicles," *Journal of Advanced Transportation*, vol. 2018, 2018.

[13] K. Zheng, "ROS Navigation Tuning Guide," *arXiv preprint arXiv:1706.09068, 2017*, 2016.

[14] S. K. Gehrig and F. J. Stein, "Elastic bands to enhance vehicle following," in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*, 2001, pp. 597-602: IEEE.

[15] M. Keller, F. Hoffmann, C. Hass, T. Bertram, and A. Seewald, "Planning of optimal collision avoidance trajectories with timed elastic bands," *IFAC Proceedings Volumes* vol. 47, no. 3, pp. 9822-9827, 2014.

Tien Ngo Manh Graduated Engineering Degree major in Automatic Control at Hanoi University of Science and Technology (HUST) from 1996-2001. Defensed Dr. Degree in Electrical Engineering at HUST in 2014. Now, works at Institute of Physics, Vietnam Academy of Science and Technology. The main researches: Process control, adaptive control, fuzzy logic and neural network control, automatic robot control, electro-optical system, image processing.

Cuong Nguyen Manh senior student major in Electrical – Automatic Control at Hanoi University of Science and Technology (HUST). Now, working at Institute of Physics, Vietnam Academy of Science and Technology. The main researches: Adaptive control, fuzzy logic and neural network control, Deep learning and Robot Operating System programing for robotics.

Manh Tran Van senior student major in Electrical – Automatic Control at Hanoi University of Science and Technology (HUST). Now, working at Institute of Physics, Vietnam Academy of Science and Technology. The main researches: Adaptive control, fuzzy logic and neural network control, Deep learning and Robot Operating System programing for robotics.

Dung Pham Tien senior student major in Electrical – Automatic Control at Hanoi University of Science and Technology (HUST). Now, working at Institute of Physics, Vietnam Academy of Science and Technology. The main researches: Adaptive control, fuzzy logic and neural network control, Deep learning and Robot Operating System programing for robotics.

Hiep Do Quang received the B.E in Instrumentation and Industrial informatics (2001), M.E (2005) in Instrumentation and Control from Hanoi University of Science and Technology (HUST). He is a lecturer, Faculty of Electrical Engineering. University of Economic and Technical for Industries, Hanoi, Vietnam. The main researches: AI, robotics, automatic control.

Kiem Nguyen Tien received a master's degree in *Measurement and Control System of* the Hanoi University of Technology (HUST) in 2003. In 2018, he received a doctor degree in Control and Automation engineering. From 2000 to 2013 he is lecturer of Department of Electronics - Automation of Hanoi University of Industry (HaUI). Currently he is a lecturer, head of Division of Industrial Electronics, Department of Electronics – Hanoi University of Industry (HaUI), fellow at the Information Technology Institute - Vietnam Academy of Science and Technology in Vietnam. The main research is the design and implementation of measurement systems, industrial control, industrial robot control and industrial network control system.

Duy Nguyen Duc student major in Computer Engineering at Hanoi University of Industry. Now, working at Institute of Physics, Vietnam Academy of Science and Technology. The main researches: Deep learning and Robot Operating System programing for robotics.