SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348 – 8379 / https://doi.org/10.14445/23488379/IJEEE-V8I1P106

Volume 8 Issue 1, 32-36, January 2021

Building Environmental Awareness System for Mobile Robot Operating in Indoor Environment on ROS Platform

Nguyen Duc Dien[1], Nguyen Duc Duong[2], Vu Anh Nam[3], Tran Thi Huong[4]

[1,3,4]*University of Economics - Technology for Industries, Viet Nam*

[2]*Viet Nam National University of Agriculture*

Received Date: 15 December 2020
Revised Date: 16 January 2021
Accepted Date: 28 January 2021

Abstract *- The paper presents a navigation system for robots operating in indoor environments, with three basic functions of positioning, mapping and planning the path to a robot in indoors with high flexibility and fast movement speed. Specifically, the robot's positioning data is extracted from an IPS indoor positioning system using high-frequency ultrasonic technology. A small error is very suitable for use in the indoor environment. Map creation and analysis function developed based on open-source ROS software devices, combined with a 360-degree scanning LIDAR depth sensor to produce a 2D map with high accuracy of cm. compared with the actual environment. Finally, the popular route searching algorithms currently used are based on the analyzed map data and robot positioning data.*

Keywords — *Robot Operating System (ROS), Rviz, Navigation, Simultaneous Localization and Mapping (SLAM).*

I. INTRODUCTION

Technological advances in the robotics sector have contributed to many industrial and social sectors in recent times. Today, many robotic systems applications can be found in factory automation, surveillance systems, quality control systems, AGVs (automatic vehicle navigation), disaster protection, support medical, etc. More and more robot applications aim to improve our daily lives, and robots are now being caught more often than ever before performing various tasks [1]. The SLAM problem arises in navigating mobile robots through unspecified environments without maps [2-5]. Techniques using robotic probabilities have been studied to suggest SLAM problem-solving techniques [6] and [7]. When the ROS operating system came into existence, robotic systems' construction using SLAM techniques for mapping and positioning was more and more popular, as shown in [8] and [9]. For many such applications, the robot's automatic mobility is a must-have issue. Mobile autonomous robots can perform tasks in a structured or unstructured environment without constant human guidance. Fully automatic mobile robot capable of: Collect information about the environment; Working for a long time without human intervention; Movement in whole or in part in its operating environment without human assistance; Avoid situations that are harmful to people, property, or yourself, unless it's part of design specifications. Automated mobile robots can also learn or acquire new abilities such as adapting strategies to complete their tasks or adapting to changing surroundings [10-13]. For any self-propelled moving machine, its ability to navigate within its operating environment is crucial. The ability to avoid dangerous situations such as collisions and unsafe conditions comes first (temperature, radiation, exposure to weather, etc.) [14]. Navigating a robot means that the robot can locate itself in its frame of reference and then plan its path to a number of target locations. To navigate in its environment, a robot or any other mobile device requires a map of the environment and the ability to analyze that map. Navigation can be defined as combining three basic functions: Self-positioning; Construction and analysis of maps; Plan your way. Some robot navigation and navigation systems use simultaneous mapping and positioning techniques to create 3D versions of their surroundings. Positioning for the robot denotes the robot's ability to set its position and orientation in the frame of reference. Path planning is an extension of positioning in that it requires determining the robot's current position and the target's position, both of which need to be in a reference or coordinate system. . Map construction can be in the form of a metric map or any symbols describing locations in the robot frame of reference. The block diagram of the navigation system for the mobile robot is shown in Fig. 1.

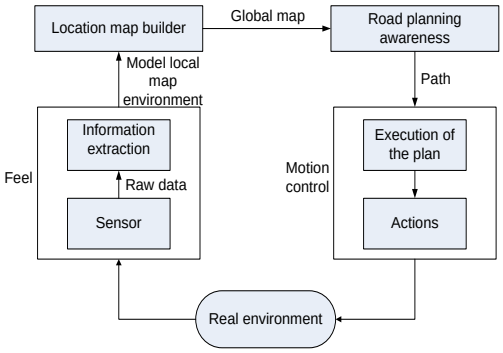

Fig. 1 Navigation system block diagram for mobile robot

II. NAVIGATION AND NAVIGATION SYSTEM FOR ROBOT

A. Robot Operating System

The ROS (Robot Operating System) is a flexible platform for programming software for robot systems. It includes tools and libraries that simplify the construction of complex robot systems by combining robot platforms. More than that, ROS is built to facilitate the convenient development and combination of the robot software. It provides the modes of operation of a system operating, including connecting to hardware, controlling low-end equipment, and performing tasks within a unified robot system. It also provides tools and libraries to build, write, and function on multiple computers.

ROS has enabled users to establish an environment that can collaborate on software development for robots globally. Using researched and developed APIs for robots will help shorten the research and application process, and this is also the ultimate goal of ROS.

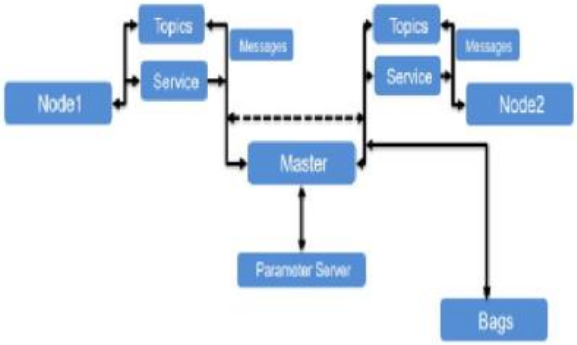

Fig. 2 The program structure of ROS

According to each task, the communication structure of ROS is developed through nodes, and the nodes are packaged in different packages. Communication between nodes in the form of topics, messages, and services is illustrated in Fig. 2.

B. Develop a Map of the Environment

The room map was built using Lidar sensors to scan and store the integrated map on the Turtlebot robot that moves the entire area in the room. TurtleBot is a ROS standard platform robot. There are 3 versions of the TurtleBot line. TurtleBot1 was developed by Tully (Platform Manager at Open Robotics) and Melonee (CEO of Fetch Robotics) from Willow Garage based research on iRobot's Roomba, Create, to implement ROS. It was developed in 2010 and sold since 2011. In 2012, TurtleBot2 was developed by Yujin Robot based on a research robot, iClebo Kobuki. In 2017, TurtleBot3 was developed with features that added the missing functionality of its predecessor and its users' needs. TurtleBot3 uses ROBOTIS DYNAMIXEL intelligent actuator to drive [15].

TurtleBot3's core technologies are SLAM and Navigation, making it suitable for service robots for homes. TurtleBot can run SLAM (simultaneous positioning and mapping) algorithms to build a map and move around the room independently. Fig. 3 shows the Turtlebot robot with a 360^0 scanning Lidar laser sensor for SLAM and Navigation purposes.

Fig. 3 Pictures of the actual TurtleBot robot

To build 2D maps, TurtleBot is a commonly used option today; the output of the SLAM package installed on TurtleBot will be a 2-dimensional image map, the type of map commonly used also to copper ROS. Fig. 4 shows an example of the map TurtleBot scanned. The map is very basic construction, making our analysis easier. As in the image shown, the white area is the area where the robot can freely move, the black area is the area occupied by obstacles such as (walls, tables, chairs, cabinets), which are fixed objects where The robot cannot move, and finally, the gray part is an area that has not been explored by the robot. This map, modified when created, will be used to navigate the robot.

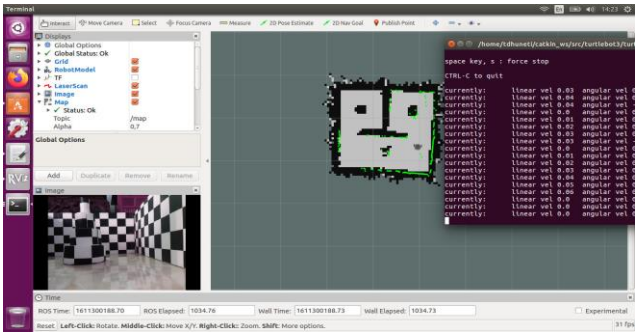

Fig. 4 Room map constructed by robot TurtleBot

TurtleBot robot can implement SLAM software packages using different algorithms based on ROS such as G mapping, Cartographer, hector, karto, but the most common use is G mapping. With 360^0 scan laser sensor data, TurtleBot can create a map and position itself in that map. For the G mapping algorithm, the robot uses the Rao-Blackwellized Particle Filter to find the coordinates close to the robot's actual position on the map. Fig. 5 shows the Particle Filter algorithm flowchart for the process of both locating and updating the map to the database. This approach uses particle filters where each particle carries an individual map of its environment. The particles, after being co-ordinated from the sensor data, will be assigned a certain weight. After the re-sampling, the higher weighted particles will be retained. The lower weight particles will be except that the particle with the greatest weight will be designated as the robot's current position. This process continues every time data is returned from the sensor, the robot locates itself and updates the map to the database at the end of the process.

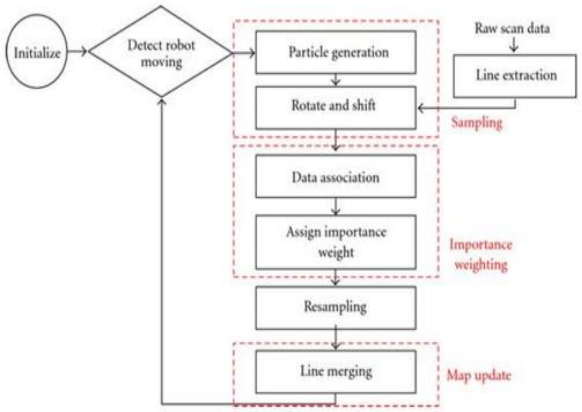

Fig. 5 SLAM algorithm flowchart of the robot using Particle Filter

Fig. 6 shows the particle sampling process when the robot is in the mapping process; as shown in the figure, the particles that are red arrows after the there-sampling process tend to converge to the robot's position in the real world. In fact, in this case, the longer the robot rotates in place, the more the particles will converge, thereby giving the most accurate coordinates.

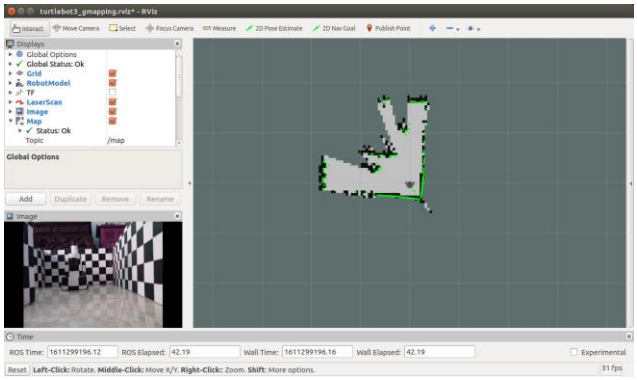

Fig. 6 Robot is in the process of creating a map using a particle filter

C. Indoor Navigation System for Robots

IPS is used to get robot coordinates to determine locations in small or medium-sized places where GPS and other satellite technologies are inaccurate or completely defective, such as inside multi-story buildings and courtyards. Bay, lane, garage, and underground locations. However, the IPS doesn't always give the correct signal, especially if too many people interfere with the transmission. Moreover, not only the IPS but sometimes the sensor itself is also inaccurate, and there is a lot of endogenous noise. To make the Robot smarter and move stably, stable and reliable coordinate data is required. The most reasonable way is to add additional sensors such as an Accelerometer, rotation angle sensor, and wheel encoder to combine the sensor data to produce a new coordinate data. , more stable, more reliable. To solve these problems, the Kalman filter is considered. On the other hand, if the robot has enough sensors combined with IPS for the Kalman filter, it will sense everything around very quickly and accurately. Filtration is a very popular method in embedded engineering and systems, especially in Robots. A good filtering algorithm can reduce noise from the signals while retaining user data. [16] The Kalman filter is one of the popular filters used for signal processing.

The Extended Kalman Filter (EKF) is a mathematical tool that can estimate variables of various processes for nonlinear systems. It works by linearizing the nonlinear state dynamics and measurement models. It is widely used in robotics engineering, popular in navigation, navigation, and control applications. This type of filter works very well in practice, and that is why it is often deployed in embedded control systems and because robots need to estimate the process variables accurately. The Extended Kalman filter is a smarter way to integrate measurement data into an estimate by realizing that the measurements are noisy and should sometimes be ignored or only slightly affect the estimate. Thai. It smooths out the effect of noise in the estimated state variables by combining more information from more reliable data from unreliable data. The user can tell the extended Kalman filter how much noise in the system, and it calculates a position estimate taking into account the noise.

The Extended Kalman Filter algorithm is still the most basic and popular solution for discrete and low-precision signals such as GPS, IPS,...Robot systems for the Kalman algorithm require IPS, encoding wheel, IMU. We used to embed EKF in the Central Processing Unit on the robot. The input data is a 2D data format that includes: Coordinates from IPS (MarvelMind Beacon Indoor Position System), Velocity from Wheel Encoder, Revs and Angular Velocity as measured with a gyro sensor Gyroscope, Accelerometer measured by sensor Accelerometer, 2 sensors are integrated into one 6DOF - Sensor is called IMU-MPU6050. The EKF algorithm is divided into 3 parts: Initialization and linearization, Prediction, and Update. Assume that the robot has x y coordinates from the IPS, linear velocity Vx vy from the Wheel Encoder, linear acceleration ax ay yaw vyaw, orientation, and angular velocity IMU. Our goal is to predict, update, and process 2D coordinates for the robot.

D. Plan a Path for the Robot

Route or plan the robot's path from current location to target location using Dijkstra's shortest path detection algorithm as in navigation system using IPS. From the location coordinates of the robot specified on the Costmap2D map, the routing program creates a path from the current position to the target location. Dijkstra algorithm is used to find the shortest path to the target position.

- Calling the node we are starting is called the initial node. Call the distance of node Y the distance from the original node to Y. The Dijkstra algorithm will assign some initial distance values and will try to improve them step by step.
- Mark all buttons that are not used. Creates a set of all unapproved nodes called the unapproved set.
- Assign all nodes an expected distance value: set it to 0 for our original node and infinity for all other nodes. Set the original node to the current node.

- For the current node, consider all of its non-visited neighbors and calculate their expected distance across the current node. Compare the newly calculated expected distance with the currently assigned value, and specify a smaller value.
- When we have examined all of the current node's unaccessed neighbors, mark the current node as accessed and delete it from the un-accessed set. An accessed node will never be checked again.
- If the destination node has been marked as visited (when planning a route between two specific nodes) or if the minimum expected distance between nodes in an unreachable set is infinity (when planning complete transmission plan; occurs when there is no connection between the original node and the remaining un-accessed nodes), then stops. The algorithm has ended.
- If not, select the unexpected button marked with the minimum expected distance, set it as the new "current node," and go back to step 3.

When planning a route, there is no need to wait until the destination node is "accessed" as above: the algorithm can stop when the destination node has the smallest expected distance among all the nodes " Unvisited "(and thus can be selected as the next" current "). The algorithm flowchart is described below.

Dijsktra route-finding algorithm

```
                function dijkstra(G, S)
    for each vertex V in G
        distance[V] ← infinite
        previous[V] ← NULL
        if V != S, add V to Priority Queue Q
    distance[S] ← 0
                while Q is not empty
        U ← Extract MIN from Q
        for each unvisited neighbor V of U
tempDistance ← distance[U] + edge_weight(U, V)
                if tempDistance < distance[V]
                    distance[V] ← tempDistance
                    previous[V] ← U
    return distance[], previous[]
```

III. POSITIONING AND NAVIGATION SYSTEM FOR AUTONOMOUS ROBOTS IN AN INDOOR ENVIRONMENT

The software system overview shown in Fig. 7 below is the proposed navigation and navigation system model.

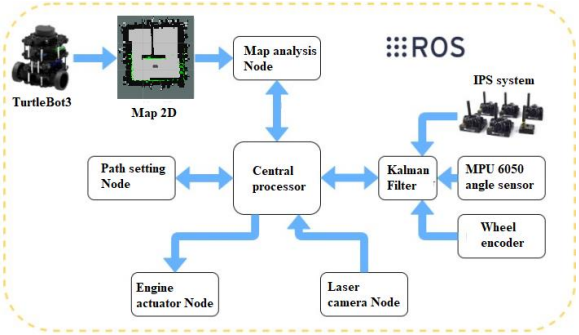

Fig. 7 General system diagram

The robot uses an open-source operating system ROS as the foundation for its processing and communication flows. The central processor is considered the robot's brain, which receives all sensor data, including laser data from the camera button, coordinate data from the IPS system, and rotation angle data from the MPU angle sensor. The central processor receives the user's request about the target position the robot needs to reach, then sends the request with the map data, robot's current coordinates, and destination coordinates to the set node. Create the path and get the array of coordinates of the robot's path point on the map. From the path point coordinate array, the central processor proposes a control algorithm to the motor actuator button, controls the robot to follow the set path, and approaches the destination.

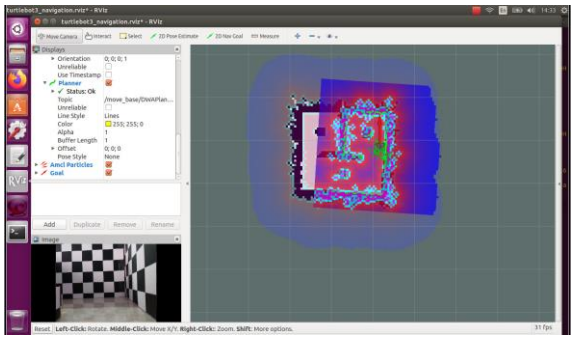

Fig. 8 Robot navigation results on Rviz

IV. CONCLUSION

Navigation and navigation system for Robot using IPS indoor positioning system has been proposed, developed, and surveyed. Experimental results show that the operating system is stable in an indoor environment with an area of about 20 m^2. The system also shows the advantages of processing speed that increase the flexibility of the robot's movement, and at the same time, it also overcomes the disadvantages of popular systems in the world that SLAM uses Lidar sensors. The positioning and navigation system for robots used in the indoor environment is an open module that can be expanded and integrated for various applications, especially for indoor tour-guide robots.

ACKNOWLEDGMENT

This study was supported by the University of Economics - Technology for Industries, Viet Nam; http://www.uneti.edu.vn/.

REFERENCES

[1] Chen, Z., Birchfield, S.T.,Qualitative Vision-Based Mobile Robot Navigation, In Proc. IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida (May 2006).

[2] R. L. e. a. Guimarães,ROS navigation: Concepts and tutorial", Springer, Cham, (2016) 121-160.

[3] A. a. P. A. Pajaziti,SLAM–map building and navigation via ROS, International Journal of Intelligent Systems and Applications in Engineering, 2(4)(2014) 71-75.

[4] Z. e. a. An, "Development of Mobile Robot SLAM Based on ROS," International Journal of Mechanical Engineering and Robotics Research, (2016) 47-51.

[5] R. K. e. a. Megalingam,ROS based autonomous indoor navigation simulation using SLAM algorithm, Int. J. Pure Appl. Math, (2018) 199-205.

[6] F. e. a. Albers,Online Trajectory Optimization and Navigation in Dynamic Environments in ROS, Robot Operating System (ROS). Springer, Cham, (2019), p241-274.
[7] S. Thrun.,Probabilistic Robotics, Communications of the ACM, (2002) 52-57.
[8] R. e. a. Giubilato.,An evaluation of ROS-compatible stereo visual SLAM methods on an Nvidia Jetson TX2, Measurement, (2019) 161-170.
[9] R. N. a. M. K. B. Darmanin.,Autonomous Exploration and Mapping using a Mobile Robot Running ROS, ICINCO, (2016).
[10] Chatterjee, A., Rakshit, A., & Singh, N. N.,Mobile Robot Navigation., Studies in Computational Intelligence, (2013) 1–20.
[11] Roan Van Hoa, L. K. Lai, Le Thi Hoan.,Mobile Robot Navigation Using Deep Reinforcement Learning in Unknown Environments., SSRG International Journal of Electrical and Electronics Engineering (SSRG-IJEEE), 7(8)(2020) 15-20.
[12] Pham Ngoc Sam, Tran Duc Chuyen.,Research and Designing a Positioning System, Timeline Chemical Mapping for Multi-Direction Mobile Robot., SSRG International Journal of Electronics and Communication Engineering, 7(11)(2020) 7-12.
[13] Roan Van Hoa, Dinh Thi Hang, Tran Quoc Dat, Tran Dong, Tran Thi Huong,Autonomous Navigation for Mobile Robots Based on Reinforcement Learning., SSRG International Journal of Electronics and Communication Engineering, 8(1) (2021) 1-5.
[14] https://en.wikipedia.org/wiki/Robot navigation.
[15] Dijkstra's Algorithm, Gass, Saul; Fu, Michael. Gass, Saul I; Fu, Michael C., (2013).
[16] https://emanual.robotis.com/docs/en/platform/turtlebot3/ove view/.