A review of research about automatic navigation system of mobile robots based on ROS

Qiuyang Yu

Mechanical Design manufacture and Automation Major, Southwest Jiaotong University, Chengdu, China

2395078841@qq.com

Abstract. With the rapid development of technology, robotic technology is continuously updated and iterated. In this process, mobile robots are gradually entering different fields and playing important roles in people's daily lives. In this context, research on path planning for mobile robots has become a priority in order to meet the various tasks that mobile robots need to handle in different situations. This paper proposes a ROS-based mobile robot autonomous navigation system, which uses the gampping algorithm to implement SLAM technology for building environment maps. It addresses the accuracy issues found in traditional map construction methods and improves localization accuracy by combining IMU with laser odometry. The paper focuses on the problem of path planning for mobile robots and mainly compares and analyzes the advantages and disadvantages of the A* algorithm and Dijkstra algorithm in path planning algorithms, aiming to find a more suitable path planning algorithm. This paper provides support for the development of the modern mobile robot field.

Keywords: component, ROS, algorithm, localization.

1. Introduction

The mobile robot is an integrated system which is composed of environmental perception, dynamic planning, behavior control and other functions. It is the highest achievement in mechatronics, and it is one of the most active areas of scientific and technology development [1]. With the improvement of robot performance, the field of application of mobile robots has been expanded, not only in industry, agriculture, medical care, services and other industries are widely used, but also in urban security, national defense and space exploration fields and other harmful and dangerous occasions are well applied. As a result, mobile robotics has received widespread attention from countries around the world.

In the field of robotics, mobile robots have an irreplaceable role to play, with a broader range of applications and greater practicality than fixed robots. Mobile robots are a comprehensive system combining computer, communication, microelectronics, information and other specialised fields and are capable of sensing information about their environment, controlling their own movement and dynamic obstacle avoidance, and have been successfully used in military support, space exploration, catering and logistics. Depending on the movement of the base, mobile robots can be divided into several categories such as wheeled, tracked, walking and crawling robots [2], which can meet the functional requirements of a variety of application scenarios.

Today's mobile robots are becoming increasingly intelligent and diverse and are a popular research area in the robotics industry. Autonomous navigation technology is a core part of mobile robots and is an essential guarantee that robots can truly move autonomously and respond intelligently to their environment. Autonomous navigation of mobile robots refers to the use of environmental information collected by sensors to determine the position of a mobile robot in an environmental area and to drive itself towards a target position. The study of the autonomous navigation technology is divided into three basic questions [3]: (1) Where am I? (2) Where is the goal? It means that the robot has to determine the location of the goal point based on the environmental information collected: (3) How should I go to the goal? It refers to how the robot moves from the initial point to the goal point. These three fundamental questions of autonomous navigation systems are concerned with the relationship. These three fundamental issues are mainly related to SLAM and Path Planning technologies, which have been the key points and hot spots in the field of autonomous navigation research.

Foreign study about robots is earlier with mature theories and more advanced techniques. The Stanford Research Institute developed a mobile robot called Shakey in the 1970s, but shakey usually took several hours to finish the path planning due to technical limitations at the time [4]. Although the robotics research in China is only 30 years from the beginning to the present, the country has made excellent progress in this area. For example, SLAMTEC has created a relatively perfect positioning and navigation system which uses the RPLIDAR series radar as its core sensor. At the same time, matching the self-developed high-performance modular positioning and navigation system SLAMWARE, the robot can achieve autonomous positioning, automatic map building, path planning and automatic obstacle avoidance to help to solve the problem of autonomous robot walking [5].

Developed in 2010, Ros is a Linux-based meta-operating system responsible for linking different parts of a robot together and collecting data to draw a realistic map, with powerful tools and libraries in ROS to achieve different user requirements for robot models and running code for use across computers. It is widely used in the design of control systems in various objects such as self-driving cars, and autonomous robots and has now become an important part of modern robotics [6,7]. On the one hand, it provides a platform for sensing the environment, control of motion and visualisation of operational functions, and on the other hand, it has an essential meaning in terms of hardware, software and functional verification of intelligent robots. At the same time, localisation and mapping algorithms play an equally integral role in robot navigation and are becoming a popular direction in the field of robotics applications. By collecting data from IMUs or encoders via ROS [8], the position of a robot can be successfully estimated without the need for expensive sensors, enabling mobile robots to generate incremental maps of their environment. In addition, Gazebo and Rviz plug-ins are included in Ros to simulate and monitor the robot's operations. They are also important tools for implementing navigation planning for autonomous robots [9].

Therefore, this paper's research on mobile robot navigation chose to design a good and functional automatic navigation system based on the ROS platform, which can be used to draw a more accurate raster map and determine an obstacle-free route. This system provides good guidance for the robot's operation, meets the requirements of mobile robot development and avoids the low efficiency of existing robot path planning exploration and the high cost of the optimal solution.

2. Structure

This research focuses on autonomous navigation systems for mobile robots, with the main components being organic. The key issues of environment mapping, real-time autonomous positioning and path planning for mobile robots. The research focuses on the autonomous navigation system of mobile robots. The research results are difficult to be extended. In order to improve the autonomy of mobile robots, the scalability of navigation functions and navigation tasks The goal is to improve the autonomy of mobile robots, the scalability of navigation functions and the accuracy of navigation tasks. The ROS system framework is used to design the architecture of an autonomous navigation system for mobile robots, and to build a general mobile platform on which autonomous positioning, map building, and path planning algorithms are based on this platform. The study of autonomous localisation, mapping and path planning algorithms is based on this.

The development of domestic mobile robots has been relatively mature, but the traditional A* algorithm plans paths have many turning points, which can affect the overall balance of the robot during the turning process. Therefore, this paper solves the problem by replacing the traditional A* algorithm with Dijkstra's algorithm to avoid the low efficiency of existing robot path planning exploration and the high cost of the optimal solution.

2.1. Robot models

In terms of hardware, it consists of two main modules. One module is a mobile chassis with a pair of independent drive wheels installed at the front end of the chassis and a pair of small castors at the rear end of the chassis, which are used to control the robot's direction by adjusting the speed of the wheels. This method is more convenient and relatively low-cost. Another module combines radar, motor and odometer.

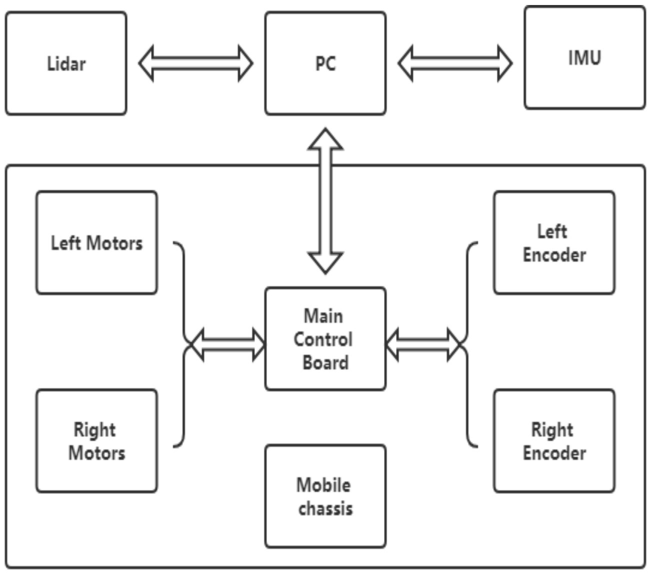

Figure 1. Overall hardware framework for mobile robots

2.2. SLAM Survey

SLAM which named Simultaneous Localization and Mapping usually are divided several different parts to renewal the information of predicted mobile robot position at all times. However, there is a big difference between the predicted position information and the current accurate position during the process of robot movement. As a result, the information of robot position can't be obtained solely from the robot motion. The position of the robot can be corrected by the information detected in the surroundings obtained from the distance measurement unit. The core of SLAM is the EKF, which is used to combine this information to estimate the exact position of the robot. The features selected above are typically called landmarks and the EKF continuously estimates the robot position and the position of the landmarks in the surrounding environment.

2.2.1. Positioning solutions. In this part, the system chooses Gmapping, which is a function pack and is exploited by using RBPF particle probability algorithm. For this package, it just has a low information requirement. Therefore, only a radar of common accuracy is required for the LIDAR, without the need for a high investment. When gmapping is running, firstly collected various initial information about the device by using the scan tool. In the actual test process, the odometer measurement results are prone to errors for areas with few feature points. To improve the positioning accuracy, this paper uses IMU with laser odometer for positioning. Finally, a raster map is drawn and displayed using rivz.

2.2.2. Algorithm analysis. This paper uses Gmapping's map building scheme. The idea of its RBPF-SLAM algorithm is that the posterior probability $p(x1:t, m|z1:t, u1:t-1)$ of the map m and robot's trajectory $x1:t$ is decomposed into the product of the two posterior probabilities of the map estimation and robot's trajectory estimation, as shown in equation (1).

$$p(x1:t, m|z1:t, u1:t-1) = p(x1:t|z1:t, u1:t-1) \cdot p(m|x1:t, z1:t) \tag{1}$$

where $p(x1:t|z1:t, u1:t-1)$ is the posterior probability of the robot trajectory $x1:t$, derived from $z1:t\ vs.\ u1:t$. $p(m|x1:t, z1:t)$ is the posterior probability of the map m, which is derived from $x1:t\ vs.\ z1:t$.

Using the RBPF-SLAM algorithm, incremental maps of the environment are constructed with known sensor observations and odometer data. As particles are continuously iterated, high weight particles are gradually retained while low weight particles are discarded. In the resampling stage, if the effective particle number Neff is less than the set threshold Nth, resampling is performed. As shown in equation (2), w(i) denotes the normalized weight.

$$N_{eff} = 1/\sum_{i=1}^{N} (w^{(i)})^2 \tag{2}$$

Finally, the corresponding $p(m|x1:t, z1:t)$ is calculated using each particle's trajectory $x1:t$ and the observation information $z1:t$, in order to achieve the update of the map.

The content above mentioned is just a special fundamental computational process. In addition, there are a lot of research directions that have not been mentioned. So, as the SLAM are mentioned above, it still can have plenty of improvement. An example of this is the problem of closing the loop of a robot, i.e., how to bring the robot back to the place where it previously appeared. The robot should be able to recognize reappearing landmarks and update the landmark information with the first information that appears. In addition, the robot should update the landmark information before the robot returns to a known place. Alternatively, we can combine SLAM with a grid approach to map the map into a human-readable format.

2.3. Navigation Survey

Navigation consists of two main components, which are positioning and path planning. The robot deduces its position in an arbitrary state on the basis of the information uploaded by the lidar and the odometer, and the auxiliary robot realizes the optimal path planning in navigation [4].

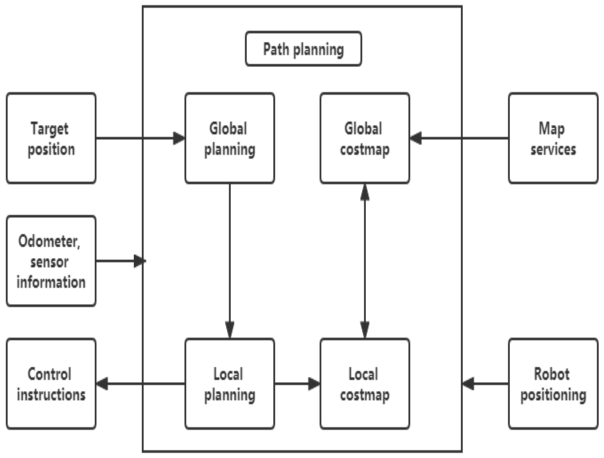

Figure 2. Navigation function frame diagram

The specific process is that a multi-objective point navigation algorithm is first used to calculate the shortest path between the starting point and the end point in order to pass through the target region according to the requirements. A global path planner is then used to plan a collision-free path between the regions.

For global path planning between target regions, experimental tests use the A* algorithm and Dijkstra's algorithm.

The A* algorithm is the most efficient direct search method for solving shortest paths in static road networks and is an effective algorithm for solving many search problems. The closer the distance estimate in the algorithm is to the actual value, The closer the distance estimate is to the actual value, the faster the search will be.

It is a graph search algorithm which is widely used. By knowing the location of the target point, cit is easy to get success in global planning. The valuation equation has a fundamental modality, which is

$$f(n) = g(n) + h(n) \tag{3}$$

$f(n)$ is the minimum path cost for the start position to reach the end position. $g(n)$ is the determined path cost for the start position to reach the intermediate node n; $h(n)$ is the estimated path cost for the intermediate node n to reach the end position.

The calculation of h(n) is the key to the heuristic function, and there are three expressions for the distance.

Manhattan distance refers to the sum of the difference between the horizontal and vertical coordinates between the intermediate and target points.

$$h_1(n) = |x_1 - x_2| + |y_1 - y_2| \tag{4}$$

Euclidean distance refers to the linear distance between two nodes in the coordinate system.

$$h_2(n) = \sqrt{}(x_1 - x_2)^2 + (y_1 - y_2)^2 \tag{5}$$

The Chebyshev distance is the maximum of the difference of the components of the two nodes.

$$h_3(n) = max(|x_1 - x_2|, |y_1 - y_2|) \tag{6}$$

Final, Euclidean distance is chosen for the heuristic function distance according to the comparison of the robot's trajectory.

In simple terms, the predicted value comparing with the actual value of the current peak to the end point is closer, the efficient of finding the shortest path with A* algorithm is more. On the contrary, if the predicted value of the distance is dramatically different from the actual value of the endpoint, then the algorithm probably has a less efficient than Dykstra's algorithm. The correct answer may not even be obtained if the difference is even more significant. However, when the distance estimate is smaller than the actual distance, it is certain that the correct answer will be obtained (it is just that if a suitable distance estimate is not set, the efficiency will become worse).

The A* algorithm is often used in game programming, for example, to compute a plan of action while an adversary is catching up with the player, but because the algorithm has such a computational calculated amount, it can delay the game as a whole. Therefore, when programming in practice, it needs to be considered in combination with other algorithms or adapted accordingly to the specific application scenario.

Dijkstra's algorithm is the typical minimum path planning algorithm used to calculate the shortest path from one node to other nodes. Its primary characteristic is centered on the starting point and expands outwards in layers until it expands to the end point. It was first proposed by the Dutch scientist Dijkstra in 1959 [5] and is described as a three-step process of initialization, finding the label p and modifying the label t.

$$G = \{U, S, Q\} \quad Q = U - S \tag{7}$$

U is the set of all points in the directed graph. S is the set of points for which the shortest path has been found, and in the initial state, let there be only starting points $\in S$ starting points $\in S$. Q is the set of points for which the shortest path has not yet been found.

Let Lk be the shortest distance from the starting point through a number of points in S to point k in the current case ($k \in U$), with the initial L starting point $= 0$ and all others equal to $+\infty$.

The algorithm uses a heuristic function to make the generated paths keep approaching the target point. After starting from the starting point, find a neighbor n of the starting point along a particular arc (set the weights to arcs).

Set

$$L_n = min\{L_n, L_{start} + arcs\} \tag{8}$$

Find the smallest L_k which is point v in the set Q. Then L_v is the shortest path length from the starting point to v. Remove the point v from Q and add it to S, repeat all the above operations for the point v until $S = U$, $Q = 0$ and L_k is the shortest path length from the starting point to each point at the end.

In essence, when comparing the two algorithms, it is easy to see that the first Dijkstra algorithm calculates the length of the shortest path from the source point to all other points, while the A* algorithm focuses on the shortest path from point to point (including specific paths).

The second Dijkstra algorithm is based on a more abstract level of graph theory, and the A* algorithm can be used more easily in, for example, game map pathfinding.

Secondly, Dijkstra's algorithm is essentially a kind of broad-first search, a divergent search, and therefore it has a high space complexity and time complexity. For these current points which is located on the road, the A* algorithm records their cost about moving to the origin, and it also computes the predicted cost from the present point to the goal, which can also be regarded as a depth-first algorithm.

Finally, if target points are far more than normal condition, the A* algorithm will lead to a large amount of redundant data and a complex evaluation function. As a result, if just need to compare road lengths insted of getting a particular path, A* algorithm will become a better choice.

3. Result Survey

In order to address the inefficiency and high cost of current mobile robot path planning and to verify the feasibility and effectiveness of the ROS-based Dijkstra algorithm for solving path planning problems, both the D and A* algorithms are studied and compared in this paper, and the simulation results are shown in Figure 3 and Figure 4.

According to the figure, there is a collision-free path from the start position to the end position of both algorithms. However, Dijkstra's algorithm will find the shortest path from the start point to each vertex in order, starting with the vertices that are close to the start point. This means that the shortest paths to some of the vertices further away from the endpoint will also be calculated, but this part is actually useless. In contrast, A* estimates a value in advance and uses this value to eliminate some useless calculations [10].

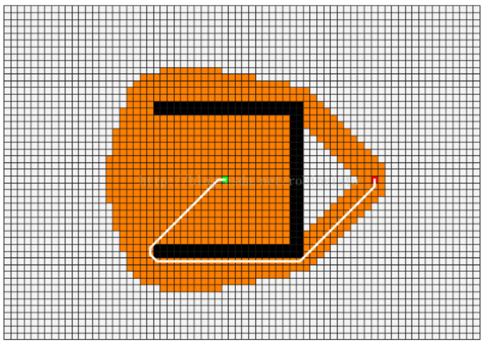
Figure 3. Simulation results of A* algorithm [11]

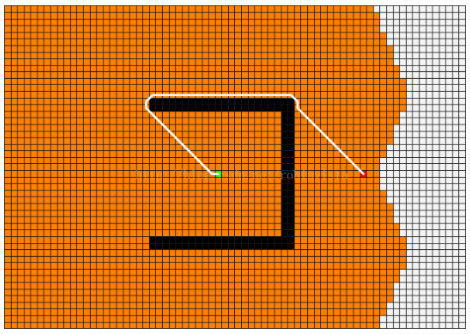

Figure 4. Simulation results of Dijkstra's algorithm [11]

4. Conclusion

This paper achieves more accurate positioning and demonstrates the good efficiency of Slam in constructing 2D maps using Gmap stacks through the fusion of laser odometry and IMU. Furthermore, the Dijkstra algorithm is compared with the A* algorithm to derive a more efficient path planning. Simulation results in Mathematica show that the A* algorithm is very efficient in terms of stability and applicability. In contrast, the Dijkstra algorithm starts from the starting point and expands in layers of computation around it, reaching the target point after computing a large number of nodes. This ultimately leads to slow and inefficient speeds.

References

[1] Ya Xu, “Research on Path Planning for Mobile Robot Based on Reinforcement Learning”, Shandong University Journal, 2013.

[2] Qiangqiang Ye, et.al, “Realization of indoor autonomous navigation mobile robot system based on ROS”, Transducer and Microsystem Technologies, no. 2, vol. 41, pp. 90-93, 2022.

[3] Guangzhou Guangya Frankfurt Exhibition Co, Development history and current situation of mobile robot, messe frankfurt, 2019, https://siaf.gymf.com.cn/newslist/industrynews/37018,Accessed 3 April 2022.

[4] LAURI. M, RITALA. R, “Planning for robotic exploration based on forward simulation”, Robotics and Autonomous Systems, vol. 83, pp. 15-31.

[5] Dijkstra E W, A note on two problems in connexion with graphs, Numerische Mathematik, Vol. 1, Issue. 1, 1959. PP. 269-271.

[6] 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.

[7] 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.

[8] 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.

[9] Quang, H.D., Manh, T.N., Manh, C.N., Tien, D.P., Van, M.T., Tien, K.N. and Duc, D.N. (2020). An Approach to Design Navigation System for Omnidirectional Mobile Robot Based on ROS. International Journal of Mechanical Engineering and Robotics Research, pp.1502–1508. doi:10.18178/ijmerr.9.11.1502-1508.

[10] Di Wang, et.al , “Research and simulation of navigation system of mobile robot based on ROS”, Journal of North China Institute of Science and Technology, vol. 18, no. 04, pp. 54-60, 2021.

[11] blog.csdn.net. (n.d.). CSDN - Professional IT publishing platform. [online] Available at: https://blog.csdn.net/.