Global and Local Path Planning Algorithms in ROS2: A Literature Review

Leonardo Augusto Antunes [a,b].

[a] Faculty of Engineering and Exact Sciences, State University of Western Paraná, Foz do Iguaçu, Brazil, leonardo.antunes@unioeste.br.

[b] Centro de Tecnologias Aplicadas, Diretoria de Tecnologias, Itaipu Parquetec. Foz do Iguaçu, Brazil, leonardo.aa@bolsista.pti.org.br.

Abstract. Autonomous Mobile Robots have a wide range of applications, although their development is not trivial. Tools such as the presented by the Robot Operating System (ROS) and the vast number of open-source packages available speed up the development. In order to navigate through an environment without a human operator, robots may rely on path planning algorithms to calculate its route, avoiding obstacles. A vast number of possible algorithms, ranging from the native to ROS packages, or the use of external technologies can be confusing when it comes to selecting algorithms for an application. This paper introduces the main components of path planning, present literature comparisons of ROS native Navigation2's algorithms along with different approaches observed in literature.

Keywords. Robot operating System, Navigation2, path planning, algorithm comparison.

1. Introduction

Mobile robots present a large set of applications, such as inspections in hazardous environments, security and surveillance, and self-driven vehicles [1].

In general, as states ISO 8373:2021 [2], mobile robots are "robots able to travel under their own control," independent of those being wheeled, legged, tracked, or even aerial or aquatic.

In many scenarios, it is possible and desired for a mobile robot to be controlled remotely by a human operator. Although, in some cases, an autonomous approach is more convenient or even the only viable option due to the high latency of communication.

Autonomous robots are defined as able to perform their tasks without a human operator, basing their actions on their own sensors, both proprioceptive and exteroceptive. In terms of mobile robots, being autonomous is often related to navigating through the environment on its own, which includes localization, mapping, path planning, and providing direction of travel [2].

When it comes to the implementation of robots, a common challenge is to integrate the various software and systems required for their functioning, as well as the lack of standardization in hardware from different manufacturers [3].

Different solutions to integrating robotics systems have been developed, but an outstanding one is the Robot Operating System (ROS), which aims to express each software component as an individual node, incentivizing a modular structure for the system. ROS also acts as middleware, providing communication through nodes in both publish-subscribe format and client-server [4].

ROS is an open source system that got a new version in 2017: ROS2, that among other improvements adopts the Data Distribution Service (DDS) communication standard [5,6]. In this work, the term ROS will be used to refer to ROS2 for simplicity.

Apart from the modularity, the vast number of open source packages developed by the ROS community is one of the major reasons to employ it, as states [4]. When it comes to navigation, some impactful tools are provided by the Navigation 2 (Nav2) packages.

Tools such as Simultaneous Localization and Mapping (SLAM) and multiple algorithms for path planning and motion control are provided by Nav2 [7]. This way, making a mobile robot autonomous can focus on selecting and configuring algorithms rather than writing and adapting an algorithm.

This work proposes to review bibliography in order to introduce the path planning and motion control algorithms present in Nav2 and how they perform in different combinations and scenarios.

2. Methodology

This work examines the functions of path planning and motion control, known as global and local path planning, respectively, and the performance of the algorithms in Nav2. It reviews four of the five publications referenced in the Nav2 documentation along with the documentation itself, as well as additional studies published in high-impact journals.

Using the advanced search tool of IEEE Xplore, searching for: "mobile robots", "path planning", "algorithm comparison", and "Navigation 2", in all metadata, 29 results are presented of publication from 1993 to 2024. Limiting the time period from 2020 to 2024, only 14 results persist, from which 12 are publications from conferences and 2 from journals.

3. Path Planning

The task of guiding a robot through an environment is divided into two main parts: finding a viable path from the origin pose to the desired pose and converting each step of the path to the robot's actuators, often considering the robot's movement limitations and kinematics [8].

The first part, called global path planning, addresses the problem of finding a viable path in the environment from a known map of this environment, avoiding static obstacles, and trying to minimize the total distance. Global path planners typically take into account only a few or none of the robot's kinematic limitations. Nav2 offers a variety of global path planners, including Dijkstra, A, Theta, Hybrid-A*, and State Lattice [9,10,11].

Once a global path has been established, the second part comes in, called local path planner or motion control. It takes the global path, which consists of a series of intermediate poses, and generates the signals for the actuators to drive the robot through those poses [5,9].

In the local path planner, the robot's kinematics, and also dynamics for some algorithms, are considered in order to calculate the robot's movement. The local path planner can also smooth the original global path, although some sources may separate the path smoothing task as a different algorithm, as in Nav2 [9].

4. Navigation2 Algorithms

Nav2 presents two types of global path planners: holonomic planners and kinematically feasible planners. Holonomic planners typically rely on conventional grid expansion algorithms, which disregard the robot's orientation during route generation [9].

To detect collisions, these algorithms evaluate the costs of traversed cells; for instance, if the cost of a location (taking into account an inflated cost map) is rising, it is probable that the path is nearing obstacles. Holonomic planners are generally more appropriate for omnidirectional or differential drive robots, particularly when the robot's footprint may be approximated as circular. The holonomic planners present in Nav2 are Dijkstra, A*, and Theta* [9].

However, robots with more complicated maneuverability, including Ackermann steering robots, legged robots, differential drive robots, or other holonomic bases whose footprint cannot be roughly represented as a circle, are better suited for kinematically feasible planners [9].

The kinematically feasible planners take into account the robot's kinematic characteristics, although dynamics are only found in local path planners. The kinematically feasible planners present in Nav2 are Hybrid-A* and State Lattice[9].

These algorithms, as states Zhang et al. [11], optimize their paths using cost functions, this way, inefficient cost functions can lead to increased time and space complexity, excessive path turns, and longer overall path length.

About the local path planners, Nav2 presents three type of algorithms: reactive, predictive, and geometrical.

Reactive planners adjust the global route in response to environmental changes, as the sensor data received may be more current or provide additional information compared to what was available during the initial computation of the global route. Instances of reactive planners include DWB and TEB [9].

Predictive planners adjust the global route utilizing fresh data, similar to reactive planners, while additionally considering temporal factors, optimizing the timing of each operation within the global trajectory, and frequently generating complex sequences of speed orders. An instance of a predictive planner in Nav2 is MPPI [9].

Geometric planners represent the most basic model, as they refrain from altering the global route upon the detection of obstacles. Robots employing these local planning algorithms will stop until the barrier is eliminated without altering their trajectory. An instance of a geometric planner is the RPP [9].

5. Comparison

Tüfekçi and Erdemir [12] run a simulation experiment on Gazebo in order to evaluate the performance of three global path planners in an unknown environment. The global path planner algorithms compared are Dijkstra, A*, and Carrot planner; all of them are paired with DWA as local path planner.

In Tüfekçi's and Erdemir's [12] experiments, the planners should calculate a path, but when faced with a new obstacle, they should recalculate a new path. Upon evaluating the total journey duration of the robot, from the initial to the final position across two distinct environments, both Dijkstra and A* algorithms exhibited comparable travel times; however, the Carrot planner demonstrated travel times exceeding double those of the other methods.

Gurevin et al. [13] performs a simulation comparison of three local path planners: DWA, TEB, and Trajectory planner. In the experiment, the robot travels through 6 stations, with all the cases considering the same global path planner.

In Gurevin et al. [13] experiments, DWA performed the shortest travel distance, while TEB performed the longest. The DWA local planner algorithm was 24.64% more successful than the TEB local planner and 2.39% more successful than the Trajectory local planner in terms of time it took to visit all stations.

In Macenski et al. [9], a robust comparison is presented; among the global path planners, all the algorithms present in Nav2 are evaluated through plan time and path length.

In terms of plan time, Hybrid-A* and State Lattice outperformed Dijkstra, Theta*, and A* by a notable margin, with Hybrid-A* presenting the shortest time and Theta* the longest. Although, in path length, the results produced similar lengths, being Dijkstra slightly longer and A* slightly shorter [9].

Macenski et al. [9] also presents a comparison for the local path planners, comparing DWB, TEB, Graceful, MPPI, and RPP according to the maximum frequency they can iterate; this can describe the algorithmic computational power required.

In this comparison, MPPI showed as the most computationally hungry algorithm, followed by TEB, DWB, Graceful, and RPP. Macenski et al. [9] also classify what types of robots each algorithm is compatible with, being DWB and Graceful not viable with Ackermann or Legged mobile robots, nor are the Dijkstra, Theta*, and A* global planners.

Antunes [14] produces a simulation experiment, combining the global path planners Dijkstra, A, *Theta*, and Hybrid-A*, along with the local path planners DWB, MPPI, and RPP. The experiment also considers the Rotation Shim combined with each of the local path planners, producing a total of 24 combinations.

In the experiments, Antunes [14] sets 11 poses which the robot should navigate through; the latter compare the combinations according to travel distance, travel time, and maneuvering time. The last is considered the total time the robot spent in without major dislocation, mostly adjusting its orientation.

In Antunes' [14] experiments, the combination of Hybrid-A* and DWB stood out with the shortest travel distance and shortest maneuvering time. The shortest travel time occurred in the Theta* and DWB combination, which presented a travel distance similar to the shortest.

6. Different Approaches

Apart from the native algorithms present in Nav2, it is also possible to implement different algorithms into the Nav2 ecosystem, using the rest of Nav2 environment with a different global path planner, for example.

Zhang et al. [11] proposes an improved JPS as global path planner. The original JPS algorithm is a modification of A*. In the simulation experiment, the proposed algorithm outperformed A* in 15.9% in terms of time efficiency and 36.3% in total length path.

Focused on human-robot interactions, Chu et al. [15] discusses the use of algorithms able to optimize paths in real time with active object capability. Chu et al. [15] technique considers an RRT* algorithm for providing a pre-planned path and further improves it considering risk-based constraints.

Manwal et al. [16] approach also consider environments shared by robots and humans. While most path planners seek to minimize travel distance, the [[Manwal2024DevelopmentOfUtilisingMR]] method combines potential fields method with the concept of risk map.

Finally, the Manwal et al. [16] algorithm uses Dijkstra for planning low-risk paths, respecting humans personal boundaries and social dynamics as conversing individuals.

Hosni, Kheiri and Najafi [17] propose a global planner based on an artificial neural network model. The model can obtain navigation policies from a skilled operator and then utilize this knowledge in new environments through appropriate training data. Hosni's, Kheiri's and Najafi's [17] study conducts comprehensive simulation tests to evaluate the model's capabilities, demonstrating its capacity for training with this data and application in novel contexts.

Liu et al. [18], on the other hand, proposes an indoor partitioning algorithm for processing indoor maps and improves the performance of global path planners. It effectively lowers the computational demands of the global path task by analyzing the global map, therefore enhancing the computational efficiency of the global path planning algorithm.

Liu's et al. [18] algorithm minimizes the computational workload of the global path planning algorithm by identifying the partitions containing the starting and ending points in the global path planning task, thereby disregarding irrelevant global map information when transitioning from the starting partition to the ending partition.

Wei et al. [19] propose the use of Deep Reinforced Learning for local path planners, investigating the applicability of real-life algorithms within simulated robotic environments. Through experimental tests, the Wei et al. [19] approach could reduce collision frequency and improve path smoothness.

Abandoning the separation of global path planners and local path planners, Gupta, Asha, and D'Souza [20] proposes the application of bug algorithms based on insect behaviors. Gupta, Asha, and D'Souza [20] claim that bug algorithms, such as Bug 0, Bug 1, Bug 2, DistBug, Tangent Bug, VisBug, and OneBug, are simpler to implement and require fewer computational resources.

7. Conclusions

The Robot Operating System combines a series of tools to facilitate robot development. Along with the tools native to ROS, there are a vast number of open-source, community-comtributed packages that can speed up development.

Nav2 provides many important tools for the development of autonomous mobile robots, including a number of path planning algorithms, both global and local, along with other navigation features.

Most of the planners available in Nav2 are based on classical algorithms, such as Dijkstra and A*. Although few authors have explored how the global path planner algorithms and local path planner algorithms interact with each other and how different combinations can be more suitable for different scenarios.

On the other hand, plenty of authors explore new areas, proposing new algorithms in order to outperform the classical ones.

Among the new technologies being applied to path planning, those based on artificial intelligence and neural networks got promising results.

8. References

[1] Ben-Ari M, Mondada F. Elements of Robotics. Springer; 2017.

[2] ISO 8373:2021 [Internet]. *International Organization for Standardization*. 2021. Available from: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en

[3] Quigley M. ROS: an open-source Robot Operating System. *International Conference on Robotics and Automation*. 2009 Jan 1.

[4] Takase H, Mori T, Takagi K, Takagi N. mROS: A Lightweight Runtime Environment for Robot Software Components onto Embedded Devices. Proceedings of the *10th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*. 2019 Jun 6.

[5] De Rose M. LiDAR-based Dynamic Path Planning of a mobile robot adopting a costmap layer approach in ROS2 [MA thesis]. *Politecnico de Torino*; 2021.

[6] Macenski S, Martín F, White R, Clavero JG. The Marathon 2: A Navigation System. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. pages: 2718-2725

[7] Macenski S, Jambrecic I. SLAM Toolbox: SLAM for the dynamic world. *The Journal of Open Source Software*. 2021 May 13;6(61):2783.

[8] Fahimi F. Autonomous Robots: Modeling, Path Planning, and Control. *Springer*; 2008.

[9] Macenski S, Moore T, Lu DV, Merzlyakov A, Ferguson M. From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2. *Robotics and Autonomous Systems*. 2023 Jul 27;168:104493.

[10] LaValle SM. Planning Algorithms. *Cambridge University Press*; 2006.

[11] Zhang Y, Yu Z, Shi Z, Zhou Z, Qi Y. A New Method of Motion Planning for Mobile Robots Based on Improved JPS and Polynomial Trajectory Planning. *2024 4th International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. 2024.

[12] Tüfekçi Z, Erdemir G. Experimental Comparison of Global Planners for Trajectory Planning of Mobile Robots in an Unknown Environment with Dynamic Obstacles. *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. 2023.

[13] Gurevin B, Gulturk F, Yildiz M, Pehlivan I, Nguyen TT, Caliskan F, et al. A Novel GUI Design for Comparison of ROS-Based Mobile Robot Local Planners. *IEEE Access*. 2023 Jan 1;11:125738–48.

[14] Antunes LA. Avaliação de Algoritmos de Planejamento de Rota para Navegação de Robóica Móvel [Undergraduate Thesis]. *State University of Western Paraná*; 2024.

[15] Chu J, Zhao F, Bakshi S, Yan Z, Chen D. Risk-Aware Path Planning with Uncertain Human Interactions. *2021 American Control Conference (ACC)*. 2021.

[16] Manwal M, Dhabliya D, Jweeg M, Habelalmateen MI, Jabbar KA, Jaafar AHM, et al. A Development of Utilising MR to get Dynamic Motion to Solve Applications in Space. Vol. 10, *2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 2024.

[17] Hosni MM, Kheiri A, Najafi E. Target-driven Navigation of a Mobile Robot using an End-to-end Deep Learning Approach. *2023 14th International Conference on Information and Knowledge Technology (IKT)*. 2023.

[18] Liu X, Wang L, Fan Y, Liang S. Research on Map partitioning and Preprocessing Algorithms for Global Path Planning. *2023 4th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*. 2023.

[19] Wei L, Lim KG, Tan MK, Liau CF, Wang T, Teo KTK. Autonomous Path Optimization in Unfamiliar Map Through Deep Reinforcement Learning. *2023 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*. 2023.

[20] Gupta S, Asha CS, D'Souza JM. Implementation and Comparison of BUG Algorithms on ROS. *2023 2nd*

International Conference for Innovation in Technology (INOCON). 2023.