Autonomous navigation and indoor mapping for a service robot

Navegación autónoma y mapeo de interiores para un robot de servicio

Julián López Velásquez **Gustavo Acosta Amaya**

Politécnico Colombiano Jaime Isaza Cadavid, Colombia

Jovani Jiménez Builes

Universidad Nacional de Colombia

Recibido: 17/05/2023
Aceptado: 11/07/2023
Publicado: 22/09/2023

Correspondencia de autores:
julian_lopez91132@elpoli.edu.co

Abstract

Objective: Teach the operation of the Summit platform and the Powerball robot manipulator. Simultaneous Localization and Mapping (SLAM) is a quite common and interesting problem in mobile robotics. It is the basis of safe autonomous navigation of mobile robots and the entrance to new combined applications with a manipulator for instance. **Methodology:** In order to find a solution to the SLAM problem, the ROS middleware and the MRPT were selected. Autonomous navigation was tested using two methods, the MRPT navigation ROS package, which is a reactive navigation method based on Trajectory Parameter Space (TP-Space) transformations, and the ROS navigation stack, a standard for differential drive and holonomic wheeled robots. **Results:** To validate the advantages and disadvantages of both approaches, a mobile robot with strong kinematic constraints (Ackermann-steering-type) known as Summit was used. As an additional work, an application using the mobile robot Summit and a robotic manipulator (Powerball) was carried out, with the intention of picking and placing objects of the mobile robot, a widely spread application among service robotics, especially, in the area of industrial logistics. **Conclusions:** Finally, it is concluded that with the tests carried out with the robot, it was possible to demonstrate autonomous navigation, using the two mentioned methods.

Palabras clave: SLAM, ROS, MRPT, TP-Space, Robotic, Artificial intelligence.

Resumen

Objetivo: Enseñar el funcionamiento de la plataforma Summit y del robot manipulador Powerball. La localización y el mapeo simultáneos (SLAM) es un problema bastante común e interesante en la robótica móvil. Es la base de la navegación autónoma segura de robots móviles y la entrada a nuevas aplicaciones combinadas, como por ejemplo un manipulador **Metodología:** con el fin de encontrar una solución al problema de SLAM, se seleccionaros los middlewares ROS y MRPT. La navegación autónoma se probó utilizando dos métodos, a saber: el paquete ROS de navegación MRPT que es un procedimiento de navegación reactivo basado en las transformaciones del espacio de parámetros de trayectoria (TP-Space), y la pila de navegación ROS, un estándar para accionamiento diferencial y robots con ruedas holonómicas. **Resultados:** Para validar las ventajas y desventajas de ambos enfoques, se utilizó un robot móvil con fuertes restricciones cinemáticas (de tipo dirección Ackermann) conocidas como Summit. Como trabajo adicional se realizó una aplicación utilizando el robot móvil Summit y un manipulador robótico (Powerball), con la intención de recoger y colocar objetos desde el robot móvil hacia otros espacios, especialmente, en el área de logística industrial. **Conclusiones:** finalmente se concluye que con las pruebas realizadas con el robot se logró demostrar la navegación autónoma, utilizando los dos métodos mencionados.

Keywords: SLAM, ROS, MRPT, Espacio-TP, Robótica, Inteligencia artificial.

Como citar (IEEE): J. López-Velásquez., G. Acosta-Amaya., y J. Jiménez-Builes “Autonomous navigation and indoor mapping for a service robot". *Investigación e Innovación en Ingenierías,* vol. 11, n°2, 28-38, 2023. DOI: https://doi.org/10.17081/invinno.11.2.6459

Introduction

The applications of robotics have generally focused on the industrial areas, mainly mass production. These machines perform repetitive, tedious, and dangerous tasks for humans [1]. Since the last decade, the industrial robots stopped being of exclusive employment inside the production plants, focusing their applications in domestic areas, entering in a new and beneficial field known as service robotics [2]. The concept of service robot, according to the International Organization for Standardization (ISO 8373), can be defined as a robot that performs useful tasks for humans or equipment excluding industrial automation applications. However, it is worth mentioning the fact that those robots tend to operate partially or completely autonomously and the tendency is to reduce human intervention to the minimum. A simple task that any service robot must perform is to navigate in most of the cases through unstructured (unknown) environments; therefore, maps are not available, and the robot must build one on its own. The effectiveness and efficiency of navigation depends on having a map without large errors. However, this process is only one of the challenges any service robot must face in order to achieve a decent autonomous navigation.

Autonomous navigation is one of the most important objectives in mobile robotics [3]. It has three fundamental objectives: (1) autonomous exploration of known or unknown environments, (2) mapping, and (3) localization. In addition, the preparation of maps or models of the environment is a fundamental task in mobile and service robotics. Service robots can implement two types of navigation: (1) Navigation based on maps and (2) navigation without models (without maps). The map-based navigation requires a priory representation of the environment in order to plan trajectories that allow the robot to go from one point to another. However, it is not always possible to have a map, when the robot must carry out tasks in an unknown environment, then it must perform an exploration of the place in which is located, with the purpose of building a model or map of the environment. This work becomes complex because it is necessary for the robot to estimate its location on a map that is hardly in the process of being built. It means that the robot must solve simultaneously the problem of mapping and localization, both highly interrelated. This problem is known as Simultaneous Localization and Mapping (SLAM) [4]. The main objective of the present project is to implement and validate different algorithms for the autonomous navigation and mapping of indoor environments in a service robot, including solving the SLAM problem. The algorithms shall be implemented in an Ackermann-steering-type mobile robot known as Summit. In [5] an affordable solution for navigation and localization to operate in indoor environments, involving ROS, the TurtleBot2 robot, a Kinect sensor, a laptop and a pre-built map of an unknown environment is presented. In [6] a differential drive non-holonomic robot equipped with various onboard sensors (ultrasonic, Kinect, encoder) was developed and described how a ROS based control system and the gmapping package were used to achieve localization, autonomous navigation and mapping in an office like environment. [7] described a ROS-based control system applied to the Pioneer 3-DX robot used for the mapping, localization and autonomous navigation. [8] implemented and evaluated different methods of autonomous navigation: Pure Pursuit (which ended up discarded since the route followed by the robot was determined by the user while the others, the path was calculated by the robot itself), the ROS navigation stack and MRPT [9] pure reactive navigation.

What remains of this article is organized as follows: in section two the materials and methods used for the mapping, autonomous navigation and the pick & place application are presented. The algorithms used for mapping and autonomous navigation are discussed in section three. In section four, an application that combines a mobile platform (Summit) and a robot manipulator (Powerball) is explained. Results and discussion are presented in section five. Finally, the conclusions and the bibliography are shown.

Materials and methods

Never as before, robotic systems have had to perform the kind of tasks that are currently required [10, 11]. In this research, the approach has been to implement a solution for mapping and autonomous navigation on a service robot. Furthermore, a pick and place application combining a mobile robot and a robotic manipulator [12] suited for logistics or service robotics was carried out:

Robot Operating System (ROS). Is an open-source robotics middleware. ROS provides a distributed programming environment, for both, real and simulated robots, as well as hardware abstraction, low-level control, message-passing between processes, software package management and tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS has witnessed a huge community with increasing number of users and developers from academia and industry, as well as hobbyists. Nowadays, ROS has become the de-facto standard for robotics research [13].

Mobile Robot Programming Toolkit (MRPT). It is an open-source C++ library which provides developers with portable and well-tested applications covering data structures and algorithms employed in common robotics research areas [14].

Robot Summit. It is an Ackermann-steering-type mobile robot with a 4x4 traction system. It has an embedded PC (Intel DN2800MT, 4GB of RAM and a 2.5 SATA HDD). The robot was modified to be able to use the Hokuyo UST-20LX and the Asus Xtion pro live as the main data acquisition sensors [15].

Schunk LWA4P (Powerball). It is a lightweight robot arm, especially suited for mobile applications due to its internal controllers. The low current input makes it possible to operate the lightweight arm either via a power supply or battery. The arm is controlled via CAN bus interface with CANopen protocol. It is essentially composed of three double-axis rotary modules in different sizes and a coupling flange for an end effector/ gripper or a sensor [16].

Workstation. Lenovo Yoga 510 series with Intel Core i3-6100U CPU @ 2.30GHz x 4 processor, 7.7 GB of RAM and 114.5 GB of HDD.

Mapping & autonomous navigation algorithms

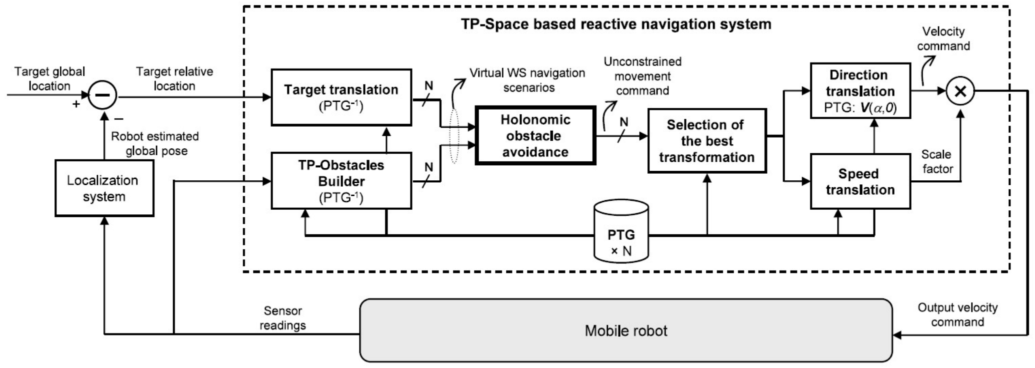
Mapping. In order to set autonomous navigation, a map shall be created [17], in this particular case, we used the MRPT. It should be noted the type of format it uses, which is known as Rawlog format. It stores the datasets and are the input of many MRPT applications for off-line processing. Since the use of MRPTROS is fundamental regarding this project, therefore, to obtain a map that works without complications, data will be acquired using rosbags and then transformed to MRPT format. It should be mentioned that the gmapping package is also a well-suited option.

MRPT reactive navigation. The MRPT navigation is a pure reactive navigation method based on Trajectory Parameterized Space (TP-Space) (See Figure 1) and Parameterized Trajectory Generator (PTGs). It is not map-based unlike ROS navigation. It means, the system generates a real-time “inner” obstacle-map and traces an optimal path based on the free-space [18].

- Trajectory parameterized space. TP-Space is defined as a two-dimensional space where each point corresponds to a robot pose on C-Space sampling surface. The components of this space are an angular component α and a distance d [19].

- Parameterized trajectory generator. The MRPT reactive navigation well-performance lies in the implementation of a set of path models to measure the distance to obstacles known as Parameterized Trajectory Generator (PTGs). They are a mapping of TP-Space points (α, d) into C-Space poses ((x, y), θ) so the straight path from origin becomes compatible with C-Space.

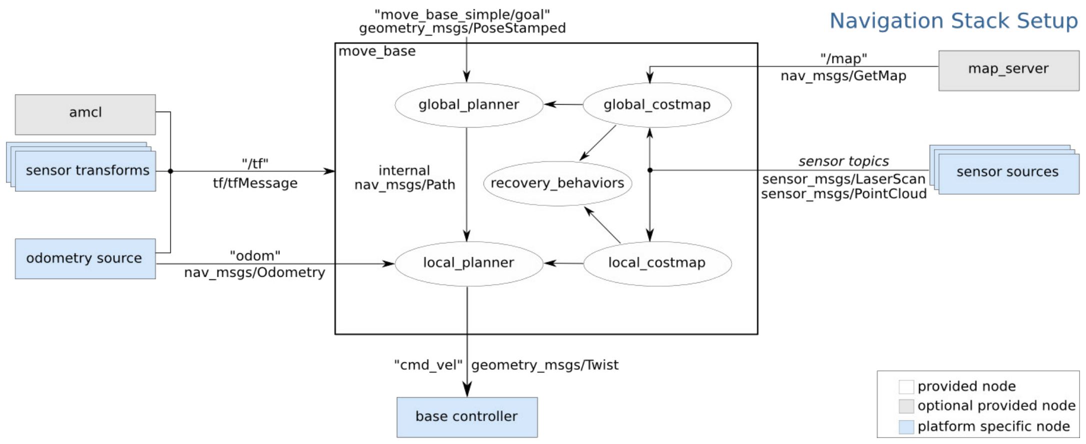
Figure 1. A complete reactive navigation system based on TP-Space involves a variety of PTGs that simultaneously translate obstacles and the target location into the TP-Space. A holonomic reactive navigation method selects a desired motion in TP-Space. The most advantageous PTG is selected and translates the holonomic motion into a non-holonomic one, generating the velocity commands for the real robot.


TP-Space based reactive navigation system
Target global location
Target relative location
Robot estimated global pose
Localization system
Target translation (PTG⁻¹)
TP-Obstacles Builder (PTG⁻¹)
N
Virtual WS navigation scenarios
Holonomic obstacle avoidance
Unconstrained movement command
Selection of the best transformation
Direction translation PTG: V(α,0)
Speed translation
Velocity command
Scale factor
PTG × N
Sensor readings
Mobile robot
Output velocity command

Source: [19].

RPT reactive navigation. The ROS Navigation stack (See Figure 2) contains two different planners. The global_planner which only calculates the shortest path from the robot estimated starting point on the map to its destination [20], not taking into consideration unexplored obstacles, and the local_planner which reacts to the local environment and those uncharted obstacles, replanning the trajectory if needed. This method is mainly meant for differential drive and holonomic wheeled robots; therefore, it does not care about the Ackermann steering of the mobile robot and its minimum turning radius. In order to fix it, the teb_local_planner package tries to overcome this limitation. This one shall run as the local planner, and also will compute the velocity commands based on its path. It requires a lot of information as input, which ultimate affects the result and the performance of the calculation.

Figure 2. ROS Navigation stack



Source: Own elaboration

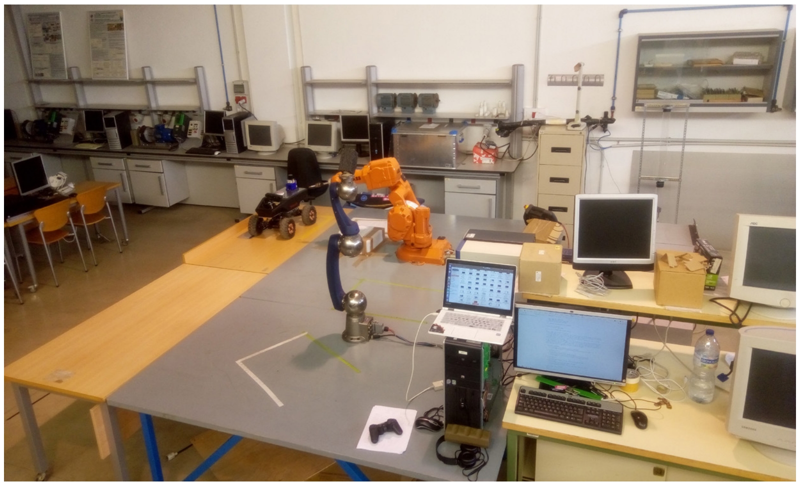
Localization. In order to accomplish a well performed navigation, the autonomous driving car needs to get localized within the map correctly. From the start until the very end, the robots odometry is calculated based on velocity commands such as the speed and the steering angle, however, since the robots initial position begins on an estimated point, its odometry is not exact at all. Therefore, a localization algorithm is required to get a more precise position of the robot within map. ROS navigation uses the Adaptive Monte Carlo Localization (AMCL) [21], which is an algorithm that uses a particle-filter to track the pose of a robot against a known (laser-based) map. The MRPT reactive navigation uses the MRPT localization package, which is a similar interface to AMCL, however, its advantage is that it supports different particle-filter algorithms.

Mobile platform Summit & robot manipulator Powerball

The Summit robot was controlled using a joystick in order to move it around to some specific areas (See Figure 3) due to the limited space in which the autonomous navigation was not very accurate, at the same time its localization was determined with the MRPT localization package. Once the Summit robot reached one of the areas, a motion planning (MoveIt) run, trying to pose the Powerball end effector above an additional fixed link (to gripper link) which was added 15 cm above the top of the Summit URDF model, its purpose was to act as a reference for the Powerball end-effector. It means, that was the position the end-effector had to reach in order to start the pick and place actions.

The end-effector of the Powerball robot was not used to avoid possible collisions with the Summit or the surroundings, a black sponge-like material with similar dimensions was used instead. It should be noted that is fully functional within the simulation, also the end-effector was added to the motion planner (MoveIt) in order to recreate a more rigorous movement trajectory.

Figure 3. Powerball-Summit Work-Area



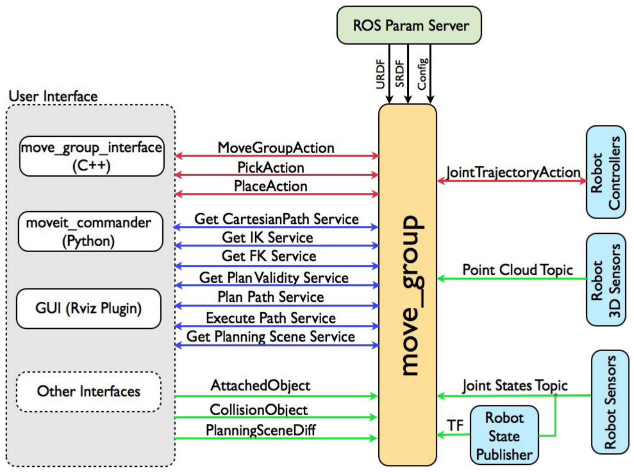
Source: Own elaboration

MoveIt! It is a high-level open-source package originally written thoroughly in C++ and later optimized for real-time performance. Earned its place as the most popular motion planning platform mainly due to the fact that its setup assistant GUI allowed a large community of non-experts and robotics hobbyists to access the many capabilities of the framework such as, motion planning, inverse kinematics, 3D perception, collision checking, into a variety of robots or even to test their own, which ultimately lead into the growth, contribution to the software's development and maintenance [22]. The core and integrator of MoveIt! high-level system is the ROS node move_group (See Figure 4) which in brief, it allows the user to interact with ROS actions and services and command the robot using one of the following methods (C++ / Python API, or GUI plugin to Rviz). This node gathers the information regarding the robot and its surroundings (JointState, TF, Point Cloud) through ROS topics [23]. The robot description (kinematics, joint limits) included in the

URDF and SRDF files, and obtained through the ROS param server. Noteworthy, the configuration files are automatically generated by MoveIt!'s assistant and stored inside the resulting ROS package.

Motion planning. Motion planning is "the technique to find an optimum path that moves the robot gradually from the start pose to the goal pose, while never touching any obstacles in the world and without colliding with the robot links" [24]. In order to guarantee the extensibility within MoveIt! components and impulsing the framework to become successful, its plugin-based architecture was a game-changing which allowed the software to be re-usable and provided the tools to create custom solutions [22].

Figure 4. MoveIt architecture diagram



Source: Own elaboration

This takes us back to the core node move_group, which through a couple of key components (ROS messaging system and the lightweight plugin interface [22]) allow the user to run the default planner OMPL (Open Motion Planning Library) [25] or any other. To start the motion planning, it all begins by using the ROS actions/services to send a request to the motion planner. That call specifies the planning requirements (i.e. to modify the current position of the actuator and/or end-effector) and, through ROS topics the node comprehends the robot's global pose and the behavior of each joint, taking into consideration that at the same time a previously defined collision check is making sure the inbuilt kinematic (position, orientation, visibility) and user-specified constraints are under control. Finally, the result trajectory will be returned by the node complying with the default and modified constraints, as well as using the specified joint velocities [26].

Kinematics. The setup assistant uses by definition the KDL numerical jacobian-based solver, however, as It has already been established, due to MoveIt! highly relies on the use of plugin infrastructure versatility, it permits the user to implement its own inverse kinematic (IK) solver by using the package IKFast which generates the C++ files for later use [27]. It should be noted that the IKFast plugin generator tool does not work with manipulators that poses more than 7 DOF [26].

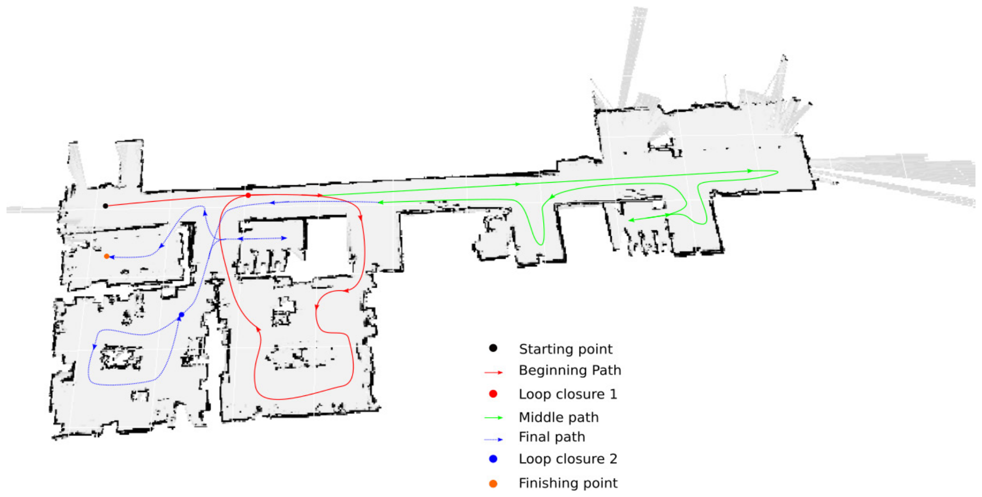
Collision checking. It deeply relies on the FCL library [28] Despite being a costly computational operation, it is mostly transparent to the user; fortunately, there are some strategies that the user can implement in order to reduce the complexity. During MoveIt! setup assistant an automatic self-collision matrix is created [22], depending on the complexity of the manipulator, the resulting matrix can turn out very decent, though, once again, the user can modify it in order to disable or enable collision checking between links or the environment.

Results and discussion

The main results obtained during the research are described below.

Mapping. In order to generate the maps, a dataset containing the odometry, laser measurements, tf, etc., was created completing the circuit in Figure 5. All obstacles are presented as black pixels, and free-spaces are the light gray areas, where the Summit is able to go. Glass doors are not detected as obstacles by the laser.

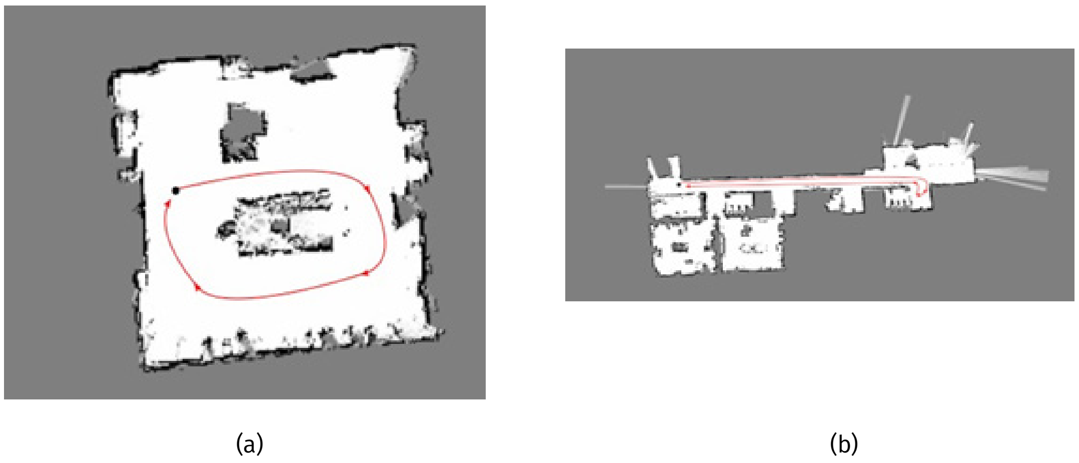
Figure 5. Circuit covered by the Summit robot to generate a map



Source: Own elaboration

Autonomous navigation. Countless experiments of exploration of the workplace with remote teleoperation were conducted in order to obtain the necessary datasets to generate the best map to set the autonomous navigation with the methods described in chapter three. The objective of the autonomous navigation was to reveal the advantages and disadvantages offered by each one of the methods, taking into consideration the rectangular shape and Ackerman configuration of the robot.

Figure 6. SLAM maps of (a) lab 1.10 CITE IV and (b) CITE IV floor 1



Source: Own elaboration

The tests were performed in the CITE IV building of Universidad de Almería. In the area that corresponds to Figure 6a, the robot had to clear one lap, however, during the second try with MRPT, the robot was able to complete two laps, not being the case with ROS navigation. In the area that corresponds to figure 6b the robot had to go to the end of the corridor and returned. In both cases, the linear and angular speeds were set to 0.5 m/s and 0.6 rad/s respectively, in order to guarantee fair conditions. The sensorial system is based in a low-cost LIDAR, a depth sensor and encoders. A representation of the maps is shown in figure 6. Among the many criteria to compare both methods of navigation (computational expenses, safest path), the execution time was selected, that is, the amount of time the Summit robot needed to clear the path. Results are summarized in Table 1.

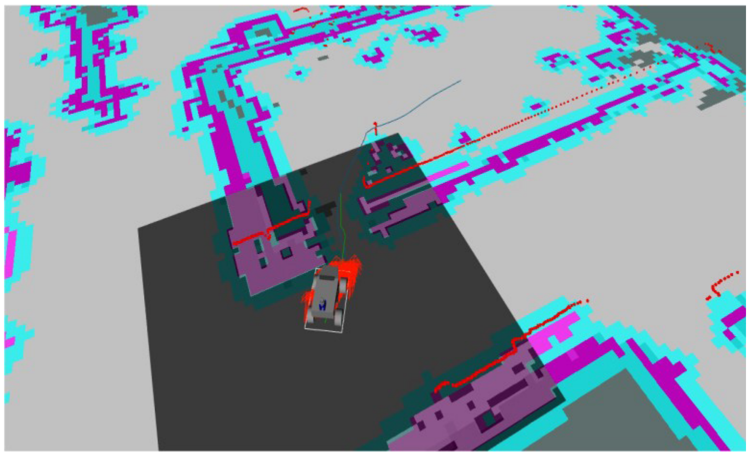
Table 1. Autonomous navigation results

MRPT navigation		ROS navigation	
Lab 1.10	Corridor	Lab 1.10	Corridor
70 s	303 s	598 s	973 s
80 s (Lap 1)	–	434 s	–
70 s (Lap 2)	–	628 s	–

Source: Own elaboration

In both navigation packages a localization system based on an adaptive particle filter [29, 30] has been used. The ROS navigation included the AMCL package, whereas the MRPT reactive navigation used the mrpt_localization_package, which is a similar interface to AMCL but supporting different particle filter algorithms and sensors. In both cases, the estimated robot position in the map is shown as a cloud-point of red arrows, as displayed in Figure 7.

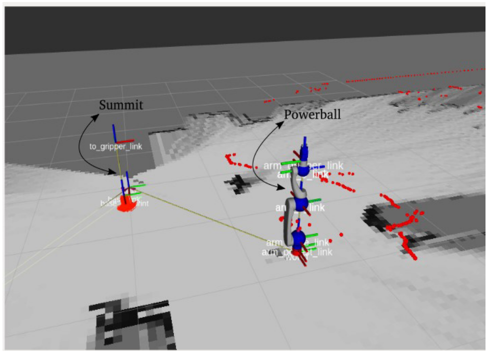
Figure 7. Local (green line) and Global (blue line) paths generated by the Local and Global planners of ROS navigation



Source: Own elaboration

A Powerball & Summit. As mentioned in section four, the Summit robot was controlled with a joystick due to the autonomous navigation ineffectiveness in the small work-area; on the other hand, the Powerball robot was working autonomously with the help of the motion planner MoveIt.

Figure 8. Powerball - Summit ROS TF



Source: Own elaboration

Once the Summit robot arrived to a specific area, the Powerball tried to pose its end-effector above the additional fixed link (to gripper link) added in the Summit's URDF model. All this with the intention of simulate a pick and place environment, a widely spread application among service robotics, especially, in the area of logistics.

Conclusions

Autonomous navigation was tested using two methods. MRPT reactive navigation and ROS navigation stack, whereas this one implements planed navigation using a global planner to recreate a path from the starting point to the goal, it should be noted that the trajectory covered with the MRPT reactive navigation was similar, keeping some distance from obstacles. The teb_local_planner worked better than using the general configuration of ROS navigation which is mainly meant for differential drive and holonomic wheeled robots, because it takes into consideration the kinematics constrains and limitations of robots with Ackermann configuration, however, it was not still enough to compete against the MRPT reactive navigation, that means, the parameters inside the local planner still can be better optimized.

During the Summit-Powerball application, there were any collisions between robots nor the Powerball with the ground, due to the well-performance of the motion planner, even though, the end-effector was not used in the real robot, it was configured within the motion planner to obtain a better result during the collision checking.

References

1. R. Megalingam, V. Naick, M. Motheram, J. Yannam, N. Gutlapalli & V. Sivanantham. "Robot operating system based autonomous navigation platform with human robot interaction". *Telkomnika (Telecommunication Computing Electronics and Control)*, 21 (3), pp. 675 - 683, 2023. DOI: 10.12928/TELKOMNIKA.v21i3.24257.
2. O. Chi, C. Chi, D. Gursoy & R. Nunkoo. "Customers' acceptance of artificially intelligent service robots: The influence of trust and culture". *International Journal of Information Management*, 70, 2023. DOI: 10.1016/j.ijinfomgt.2023.102623.
3. R. Megalingam, B. Tanmayi, G. Reddy, I. Krishna, G. Sree & S. Pai. "ROS-based GUI controlled robot for indoor mapping and navigation". Lecture Notes on Data Engineering and Communications Technologies, 58, pp. 283 - 294, 2021. DOI: 10.1007/978-981-15-9647-6_22.

4. H. Durrant-Whyte & T. Bailey “Simultaneous localization and mapping (SLAM): Part I the essential algorithms”. *IEEE Robotics Automation Magazine*, 13(2):99110, 2006. DOI: 10.1109/MRA.2006.1638022.
5. O. Hamzeh & A. Elnagar. “Localization and navigation of autonomous indoor mobile robots”. *International Journal of Computing, Communications & Instrumentation Engg.*, vol. 2, pp. 228-233, 2015.
6. N. Mohan, T. Shreekanth & B. Sandeep “Autonomous robot based on robot operating system (ROS) for mapping and navigation”. *International Journal of Computer Science & Engineering Technology*, vol. 6(7), pp. 449-457, 2015.
7. S. Zaman, W. Slany & G. Steinbauer “ROS-based mapping, localization and autonomous navigation using a pioneer 3-DX robot and their relevant issues”. In: *Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, pp. 15, 2011. DOI: 10.1109/SIECPC.2011.5876943.
8. E. Rodríguez. “Navegación autónoma de un robot móvil Summit”. Doctoral thesis. Universidad de Almería, España, 2015.
9. J. Blanco. “Mobile Robot Programming Toolkit (MRPT)”. [Online]. Available: https://w3.ual.es/~jlblanco/software/ [Last access: 03 May 2023].
10. A. Moniz & B. Krings Robots working with humans or humans working with robots? Searching for social dimensions in new human robot interaction in industry societies. *Societies*, vol. 6(3), pp. 123, 2016. DOI: 10.3390/soc6030023.
11. H. Rasheed, Y. He, H. Khizar & H. Abbas. "Exploring consumer-robot interaction in the hospitality sector: Unpacking the reasons for adoption (or resistance) to artificial intelligence". *Technological Forecasting and Social Change*, 192, 2023. DOI: 10.1016/j.techfore.2023.122555
12. G. Schunk. “Assembly and Operating Manual LWA 4P”. Schunk GmbH & Co. KG. Assembly and Operating Manual LWA 4P. [Online]. Available: https://schunk.com [Last access: 3 May 2023].
13. T. Itsuka, M. Song, A. Kawamura & R. Kurazume “Development of ROS2-TMS: New software platform for informationally structured environment”. *ROBOMECH Journal*, 9(1), 2022. DOI: 10.1186/s40648-021-00216-2.
14. H. Zhang, L. Yu & S. Fei. “Design of dual-LiDAR high precision natural navigation system”. *IEEE Sensors Journal*, 22(7), pp. 7231-7239, 2022. DOI: 10.1109/JSEN.2022.3153900.
15. L. Contreras, T. Yamamoto, Y. Matsusaka & H. Okada “Towards general purpose service robots: World Robot Summit – Partner Robot Challenge”. *Advanced Robotics*, 2022. DOI: 10.1080/01691864.2022.2109428.
16. Z. Wang, J. Peng, M. Dong, S. Song, S. Ding & Y. Liu. “Novel trajectory planning method based on double quaternion and Tau theory”. *Lecture Notes in Electrical Engineering*, pp. 482-491, 2022. DOI: 10.1007/978-981-16-6372-7_54.
17. V. Raja, D. Talwar, A. Manchikanti & S. Jha. “Autonomous navigation for mobile robots with sensor fusion technology”. *Lecture Notes in Mechanical Engineering*, pp. 13-23, 2023. DOI: 10.1007/978-981-19-0561-2_2.
18. [18] N. Baslan, S. Heerklotz, S. Weber, A. Heerklotz, B. Hofig & J. Abu-Khalaf. “Navigation and vision system of a mobile robot”. In: *Proceedings of the 19th International Conference on Research and Education in Mechatronics*, Nro. 8421777, pp. 99-104, 2018. DOI: 10.1109/REM.2018.8421777.
19. J. Blanco, J. González & J. Fernández. “Extending obstacle avoidance methods through multiple parameter-space transformations”. *Autonomous Robots*, vol. 24, pp. 2948, 2008. DOI: 10.1007/s10514-007-9062-7.

20. H. Cheon, T. Kim, B. Kim, J. Moon & H. Kim. "Online waypoint path refinement for mobile robots using spatial definition and classification based on collision probability". *IEEE Transactions on Industrial Electronics*, 70 (7), pp. 7004 - 7013, 2023. DOI: 10.1109/TIE.2022.3203684.

21. F. Dellaert, D. Fox, W. Burgard & S. Thrun. “Monte Carlo localization for mobile robots”. In: *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1322-1328, 1999. DOI: 10.1109/ROBOT.1999.772544.

22. D. Coleman, I. Sucan, S. Chitta & N. Correll. “Reducing the barrier to entry of complex robotic software: A MoveIt! case study”. *Journal of Software Engineering for Robotics*, vol. 1(1), 2014.

23. L. Yin, J. Liu, F. Zhou, M. Gao & M. Li. "Cost-based hierarchy genetic algorithm for service scheduling in robot cloud platform". *Journal of Cloud Computing*, 12 (1), Nro. 35, 2023. DOI: 10.1186/s13677-023-00395-w.

24. L. Joseph. “Mastering ROS for robotics programming”. Second edition. Packt Publishing Ltd., USA, 2015.

25. OMPL “The open motion planning library”. Technical report. Kavraki Lab, Department of Computer Science. [Online]. Available: https://ompl.kavrakilab.org/ [Last access: 03 May 2023].

26. MoveIt "Concepts". Technical report. PickNik Robotics. [Online]. Available: https://moveit.ros.org/documentation/concepts/ [Last access: 03 May 2023].

27. R. Smits. “KDL: Kinematics and dynamics library”. Technical report. Orocos Kinematics and Dynamics. [Online]. Available: http://www.orocos.org/kdl [Last access: 03 May 2023].

28. J. Pan, S. Chitta & D. Manocha. "FCL: A general purpose library for collision and proximity queries," *2012 IEEE International Conference on Robotics and Automation*, Saint Paul, USA, pp. 3859-3866, 2012, DOI: 10.1109/ICRA.2012.6225337.

29. [29] J. Martínez Garcés y J. Barreto Fereira, “Modelo de planeación para la inversión tecnológica en centros de investigación universitarios”, *Investigación e Innovación en Ingenierías*, vol. 7, n.º 2, jul. 2019. DOI:10.17081/invinno.7.2.3448

30. D. Fox. “KLD-sampling: Adaptive particle filters”. *Advances in Neural Information Processing Systems*, pp. 713-720, 2001.