ISBN: 978-628-95207-8-1. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2024.1.1.1968

Simulation and Modeling of Control and Sensorization Strategies for Autonomous Navigation with Differential Drive Robots: Unknown Environments Focused Approach

Luiz Gustavo de Lima[1], Vicente Idalberto Becerra Sablón[2]
[1,2]Universidade São Francisco, Brazil, lglima61@gmail.com, vicente.sablon@usf.edu.br

Abstract*– The objective of this article is to present the modeling and simulation of control and sensorization strategies applied to autonomous navigation of a differential drive robot, with the purpose of planning a trajectory in an unknown environment. It explores the growing impact of this technology in factory automation, particularly in tasks that are repetitive and hazardous. The evolution of mobile robotics is discussed, highlighting two popular solutions in the industry. Automated Guided Vehicle (AGV), initially guided by wires and later by reflective tapes, have evolved into highly automated systems and requires a training step prior to its use. As opposed, Autonomous Mobile Robots (AMR), are fully autonomous, embedded with recent developments on Simultaneous Localization and Mapping (SLAM) and Navigation algorithms to successfully map and plan a trajectory in an unknown environment in real-time. Using the Robot Operational System (ROS) as an open-source modeling and simulation tool, this article also addresses topics related to the implementation of such technology, highlighting challenges to overcome such as uncertainty management, positioning estimation drift mitigation and extraction of environmental features to better estimate the robot positioning in a global environment.

Keywords—robotics, autonomous navigation, robotics simulation, control modeling, SLAM.

I. Introduction

Autonomous robotics systems, whether for logistics transportation or as a robotic vacuum cleaner, are not merely a futuristic technological trend, they are an already-present reality that is reshaping how we engage with repetitive or potentially hazardous tasks. Their application is more prevalent in areas with a high repeatability of operations, as well as in tasks where there is an imminent risk to humans. This contributes to better efficiency, cost reduction, or enhanced well-being for humans.

In the current industrial landscape, autonomous robotics emerges as a promising solution to tackle the challenges associated with the non-standardization of factory floor plants where they will be deployed, introducing a significant level of uncertainty into robotic control.

These robots rely on advanced autonomous robotics technologies for recognizing the environment they inhabit, along with obstacles and potential risks, all without requiring specific environment programming or even prior learning step. This flexibility is a key advantage of this technology, albeit at the expense of its complexity.

This article presents model and simulation control and sensorization strategies applied to autonomous navigation, with the goal of planning the trajectory of a differential drive robot in an unknown environment. We will explore recent technological advancements that enable the integration of this technology into modern supply chains, along with the technical and operational challenges. By investigating deeply into this topic, our objective is not only to understand the current state-of-art but also to grasp which innovation fields are open for the future of the technology.

II. Related Work

As processes and services automation continually gain importance in the current market, mobile robotics, playing a significant role in this trend, is becoming an increasingly popular and cost-effective technology. This evolution brings awareness in the application of autonomous mobile robots, particularly for repetitive and unsafe tasks for humans [1]. The growing popularization of autonomous mobile robots in the field arises from a not addressed market requirement by automated guided vehicles, which were widely spread initially but have deficiencies in terms of flexibility and mobility [2]. The challenge in autonomous robot navigation involves developing methods and decision-making strategies applied to robotic control, enabling it to perform tasks in an unknown environment [3].

A. Mobile Robotics Introduction

For the industry, two concepts of mobile robotics have become popular over the years and have proven to meet the needs of the industrial environment in different ways. These are known as Automated Guided Vehicles (AGV) and Autonomous Mobile Robots (AMR).

Historically, AGVs were the first mobile robots made popular in the industry. Its main characteristic is the need to have environment knowledge, including their route, prior to execution. Consequently, the early AGVs used a wire-guided path planning system. Copper wires were positioned along the trajectory, and a signal generated by a dedicated controller was induced throughout its length, guiding the robot to its task. This system, as a characteristic, does not require any intelligence from the robot itself, as all control of the operation was done by the controller responsible for inducing the signal [4].

This system evolved to the use of reflective tapes along the route, bringing intelligence to the robot through sensors capable of identifying the path to be taken. In the system with reflective tapes, reference points are inserted where the robot should perform some kind of operation. For environments with constant layout changes, this system proved inefficient, requiring the user, under any change in the environment, to adapt and modify the robot trajectory.

As a way to mitigate this deficiency, a new technology was introduced using inertial sensors and distance sensors, eliminating the need for reflective tapes on the path [4]. Still, this technology requires the user to train the robot so that the route is known before its navigation. In this type of technology, the automated guided vehicle has the autonomy to decide, usually based on the shortest distance, the best path to take for a particular task. Although it has a certain degree of autonomy, this technology is unable to avoid obstacles during navigation. In summary, the technology became popular for its simplicity and cost-efficiency, but in expense of greater autonomy to adapt to changes in the environment.

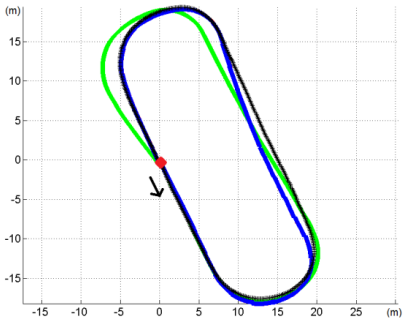


Fig. 1 Global position estimation drift example. In green: real trajectory. In black: trajectory calculated by the algorithm [5].

Unlike guided systems, autonomous mobile robots do not require any learning step or preparation of the environment in which they will be deployed. The aim is to enhance robotics by embedding sensorization and perceptual capabilities through sensors, actuators, and processing, thereby increasing autonomy [6]. Control and sensing strategies result from recent technological advances in sensors and processing, as well as the reduction in the costs of these technologies. In this concept, the robot creates a mapping of the environment based on its movement using a technique called Simultaneous Localization and Mapping (SLAM). This technique requires the robot to estimate its position and system orientation in the environment in real-time. The sensors applied to the system directly impact the outcome of the SLAM technique due to inherent measurement errors. Therefore, inertial sensors, distance sensors, and stereoscopic cameras are combined in a complex control system to achieve the lowest displacement error possible using sensor fusion techniques through the linear-quadratic problem theory (Kalman Filter).

B. Simulation and Modeling

For economic and technical reasons, the development of a robotic system is rarely done exclusively in a real environment. It is common to use mathematical models that represent the physical aspects of the robot and other conditions in a simulated environment with the assistance of software tools. Modeling is an essential process to understand and represent the physical and dynamic characteristics of a robotic control. The kinematic model is usually sufficient for the synthesis of control algorithms, especially in a simulated environment. Its concept ignores real world disturbances and variations, making it an efficient and fast strategy for testing different control strategies. However, a dynamic equation provides an assurance of real behavior in a simulated environment and vice versa.

In these circumstances, a preliminary alternative is to synthesize the controller based on the kinematic model and evaluate its performance and robustness against dynamic effects, considered in this case as disturbances [6]. With the need for the development of comprehensive models that accurately represent the dynamic characteristics of the mobile robot, and as an open-source software option, the Robot Operating System (ROS) is mentioned in this article.

The ROS framework has accelerated the adoption of standardization patterns for implementing interfaces between sensors and actuators, facilitating the modeling of robotic mechanisms through its abstraction and high-level language. For the modeling and simulation of robots of any nature, ROS implements modeling through a file format called Unified Robot Description Format (URDF). This format uses XML specification for robot description, exposing parameters related to kinematics, dynamics, visual representation, collision model, and sensor characteristics.

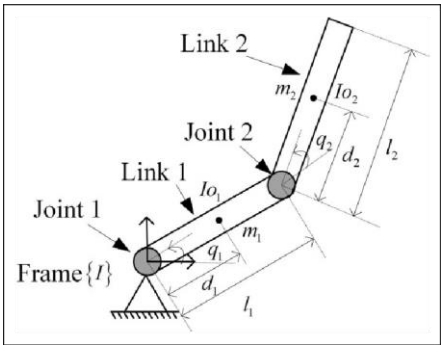


Fig. 2 Links and joints hierarchy [7].

In this concept, the robot is divided into smaller scopes called links and joints. The joints define the physical and dynamic characteristics of each part of the robotic assembly. The link hierarchically defines the relationships between the joints.

C. Navigation System

Navigation is one of the essential mechanisms for human survival. It involves the ability to move, interpret, and interact with the environment [8]. The most natural way to understand the principle of autonomous navigation is through the analysis

of human behavior. We can correlate capabilities such as sensory, visual, orientational, and cognitive with resources present in autonomous systems. This correlation is mentioned several times in the literature and is also addressed in reference [9].

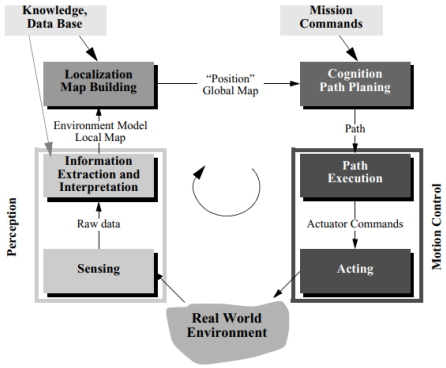


Fig. 3 Robotic control system schematic [9].

The main distinction between autonomous and automated lies in the ability of an agent, with knowledge of its physical and sensory limits, to self-regulate, generating its own rules of action based on information acquired from the environment. An autonomous agent is, above all, an automated system with its own ability to learn and adapt its behavior [10]. Although this is a correct definition, many aspects must be considered to achieve autonomy. Environmental settings, robot model, task list, and performance criteria are the main points to be evaluated, making the project work extremely complex due to its wide variation [11].

The robot interaction with the environment, through its sensors and actuators, is what will impact its execution of certain tasks since these components have limitations in their characteristics that ultimately limit the project's objective to be achieved [10]. An example of this characteristic is the choice of non-holonomic drive systems, which, in turn, limits the robot's freedom of movement in the plane, requiring control over its trajectory. Additionally, even the most precise sensors are not exempt from limitations, and the introduction of uncertainties into the system is inherent in their use. Thus, the efficient management of these uncertainties becomes a significant challenge in autonomous navigation.

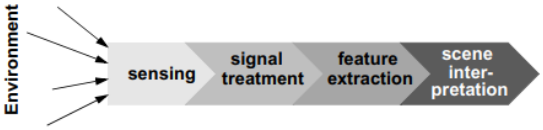


Fig. 4 Environment perception process [9].

One of the strategies found in the literature is called Feature Extraction. It involves extracting information from one or more sensors initially, generating a perception of the environment that influences the actions subsequently taken by the robot. In reference [9], feature extraction based on raw values measured by a LIDAR sensor is explored in detail, as well as using stereoscopic cameras.

Navigation is one of the most challenging competencies of an autonomous system. Its success is a consequence of the successful implementation of four foundational blocks, illustrated in Fig. 4, of the autonomous control system: perception, where the system extracts meaningful information from sensors; localization, where the system can locate itself in the environment; cognition, where the system has autonomy to decide how to act; and motion control, where the system has knowledge of its kinematics and dynamics to achieve the correct trajectory [9].

D. Differential Drive Strategy

The movement of a robot through a differential drive stands out for its simplicity and efficiency in motion, still providing enough freedom for the robot to perform many small-scale applications. This mechanism consists of two axes mounted on the same plane, where each can be controlled independently forward or backward. Robots using this mechanism depend on one or more swivel casters that support and prevent the robot from tipping.

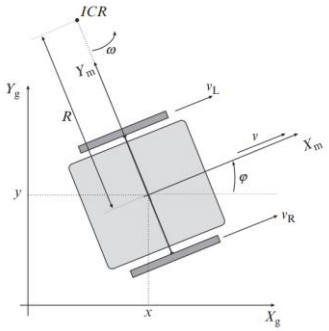


Fig. 5 Differential drive kinematics modelling [12].

The control strategy applied to this mechanism uses a kinematic modeling, making it possible to use relative global coordinates (x, z, φ) representing the estimated position and angle of the robot in the horizontal plane. This control results in the variation of the axes speed, altering the robot's trajectory in a way that respects the non-holonomic nature of the mechanism. Non-holonomic robots have constraints in their mechanism that require the system to follow a specific trajectory to reach a desired position. At the moment when the axes have different speeds, the robot moves along a curvature centered at some point to its left or right, called *Instantaneous Center Radius (ICR)*.

Considering Fig. 5, we can define the linear velocity v, in m/s, on both axes in reference to a ICR point by:

$$v_R = \omega * \left(R + \frac{l}{2}\right)$$
$$v_L = \omega * \left(R - \frac{l}{2}\right) \quad (1)$$

Where l is the distance between the center of the axes in meters, v_R and v_L are the linear velocities of the axes with respect to the plane in meters per second, and R is the distance from the ICR to the center of the robot chassis in meters. Therefore, to define R or ω, we have:

$$R = \frac{l}{2} * \frac{v_L + v_R}{v_R - v_L}$$
$$\omega = \frac{v_R - v_L}{l} \quad (2)$$

In order to simplify the inverse kinematics, motion rules can be defined to limit the behavior of this mechanism:

1. If $v_R = v_L$, then $\omega = 0$, meaning a straight-line drive;
2. If $v_R = -v_L$ or $-v_R = v_L$, and $R = 0$, then $v = 0$, meaning a rotation on robot center point.

In terms of control strategy, these definitions restrict the robot's motion to just two scenarios: linear movement or angular movement. For trajectory planning, motion should be planned in phases defined by a period T_s in seconds, and during this period, only one of the two scenarios will be executed. According to reference [12], the robot's position at a time instant t, defined in seconds, known as odometry, can be determined by integrating its kinematic model. Based on the control variables $v(t)$ and $\omega(t)$, the final position in global coordinates is defined by direct kinematics represented by the matrix:

$$q(t) = \begin{bmatrix} x(t+1) \\ z(t+1) \\ \varphi(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ sen(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \quad (3)$$

$$x(t) = \int_0^t v(t)\, cos(\varphi(t))\, dt$$
$$z(t) = \int_0^t v(t) sen(\varphi(t))\, dt \quad (4)$$
$$\varphi(t) = \int_0^t \omega(t)\, dt$$

Where x and z are coordinates in the plane in meters, φ is the angle of the robot's face in radians, v is the linear velocity of the robot in meters per second, and ω is the angular velocity of the robot in radians per second. The direct kinematics can also be represented by (4).

Assuming that the velocities v and ω are constant at the instant of time represented by $t = kT_s$, $k = 0,1,2, \ldots$, it is possible to numerically express the direct kinematics using the Euler method:

$$x(k+1) = x(k) + v(k)cos(\varphi(k))T_s$$
$$z(k+1) = z(k) + v(k)sen(\varphi(k))T_s \quad (5)$$
$$\varphi(k+1) = \varphi(k) + \omega(k)T_s$$

For linear motion ($v_R = v_L$), considering the rules applied to the kinematic model we have:

$$x(k+1) = x(k) + v(k)cos(\varphi(k))T_s$$
$$z(k+1) = z(k) + v(k)sen(\varphi(k))T_s \quad (6)$$
$$\varphi(k+1) = \varphi(k)$$

And for angular motion ($v_R = -v_L,\ -v_R = v_L$):

$$x(k+1) = x(k)$$
$$z(k+1) = z(k) \quad (7)$$
$$\varphi(k+1) = \varphi(k) + \omega(k)T_s$$

Keeping in mind that, to ensure the second condition, there is:

$$\omega(k) = \frac{2v(k)}{l} \quad (8)$$

With the inverse kinematics, it is possible to determine the control variables, defined by $v(t)$ and $\omega(t)$, in a way that an estimated position $q(t)$ is reached considering the current position. Thus, for the control variable of linear velocity v, we have:

$$v(k) = \pm\sqrt{x^2(k+1)T_s + z^2(k+1)T_s} \quad (9)$$

Where the sign depends on the desired direction – positive for forward and negative for backward. Finally, the control variable for angular velocity can be defined by:

$$\omega(k) = \frac{x(k+1)z(k+2)T_s - z(k+1)x(k+2)T_s}{x^2(k+1)T_s + z^2(k+1)T_s} \quad (10)$$

E. 2D LIDAR Sensor

A popular starting point for implementing mapping and localization algorithms in autonomous robots is Light Detection and Ranging (LIDAR) sensor. The technology is based on the Time of Flight (ToF) principle, where a beam of light is emitted and the time it takes to reflect off an object and return to the receiver is measured. Thus, the time it takes for the beam to return to the receiver determines the distance to the object, considering the constant speed of light. This same principle is applied in variations of the sensor that effectively measure one or more dimensions [13].

The sensor that measures at least two dimensions (r, θ) is the most commonly cited sensor for autonomous robotic sensorization. In this variant, the emitter and receiver are rotated at a constant speed, which can range from 1 Hz to 100 Hz. The distance to objects is measured in the horizontal plane surrounding the sensor. These sensors have a range that allows for measuring distances between 0.2 m and 25 m, with an error rate of 0.5% to 2%, which varies according to the distance. In other words, the greater the distance, the greater the error.

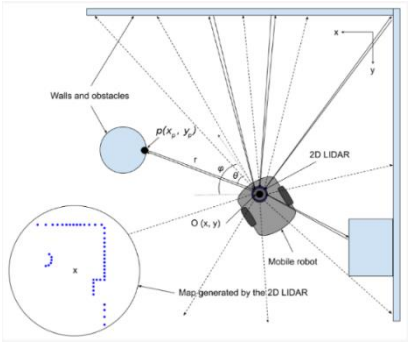


Fig. 6 LIDAR measurement concept diagram on a differential drive robot [13].

F. *Simulation Software*

Open-source software has a great impact on autonomous robotics. The community effort, with many academic contributions, brings significant progress through dedicated software platforms for robotics, breaking the barrier of the cost of developing new applications. Despite the decrease in costs with hardware used in robotics, software remains a determining role in the success of robotic control [14].

Due to its modularity, ROS implements mechanisms for interprocess communication, ensuring standardized transmission of data between different software modules within the same system. The concept of communication between modules is parallel to the MQTT protocol concept. Modules that need to expose data to be used by other modules use the *publish* method to a broker. To consume data exposed by one of the modules in the system, the *subscribe* method is used. This concept makes communication between different modules highly efficient by its event-oriented concept. In ROS, this concept is referred to as middleware [15].

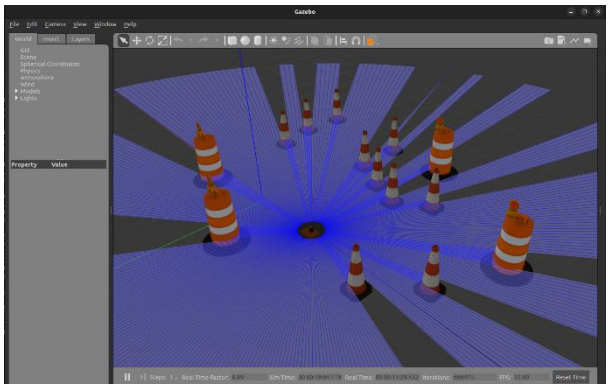


Fig. 7 Gazebo simulation tool environment.

To complement ROS functionality, especially during robot simulation implementation, there are several open-source tools that integrate with the modularity and abstraction architecture that ROS provides and add on top of those simulation features. One example of a simulation tool is the software called Gazebo, and for robot control system visualization, the software called RVIZ2. Through the integration of the control, visualization, and simulation tools, it is possible to create a highly accurate model of a robot even in a simulated environment.

III. MATERIALS AND METHODS

The simulation presented in this article used only open-source tools. The implementation was based on the ROS2 software, and consequently, the Ubuntu Desktop operating system is required to reproduce the achieved results. An introduction to the concepts of ROS2 can be found in Open Robotics official documentation.

A. *Environment Preparation*

The instructions below handle all open-source tools installation steps required in the implementation part:

1) Install Ubuntu Desktop 22 LTS, available at https://ubuntu.com/download/desktop.
2) After OS installation complete, install all updates:
   ```
   $  sudo apt update && apt upgrade
   ```
3) Install ROS2 Iron software, available at https://docs.ros.org/en/iron/Installation/Ubuntu-Install-Debians.html.
4) Install Gazebo Fortress software, available as a ROS2 module:
   ```
   $  sudo apt install ros-iron-ros-gz
   ```
5) Install SLAM Toolbox, available as a ROS2 module:
   $ sudo apt install ros-iron-slam-toolbox
6) Install Navigation2 Toolbox, available as a ROS2 module:
   ```
   $  sudo apt install ros-iron-navigation2 ros-iron-nav2-bringup
   ```
7) Setup environment variables of the current open terminal with the command below, or, to automatically setup when the terminal is opened, edit the *.bashrc* file.
   ```
   $  source /opt/ros/iron/setup.bash
   ```
8) Environment variables setup can be validated using the command below:
   ```
   $  printenv | grep -I ROS
   ```
9) The command output must be similar to following excerpt, this ensures that the paths to the ROS2 and ROS2 modules executables and libraries are known:
   ```
   ...
   ROS_DISTRO=iron
   ROS_PYTHON_VERSION=3
   ROS_VERSION=2
   ...
   ```
10) Create a ROS2 workspace within the current user home folder:
    ```
    $  mkdir -p ~/ros2/workspace/src && cd ~/ros2/workspace/src
    ```
11) Install *git* tool and clone the repository of the implementation:
    ```
    $  sudo apt install git && git clone https://github.com/lg-lima1/godot.git
    ```
12) Go to the ROS2 workspace folder:
    ```
    $ cd ~/ros2/workspace
    ```
13) Finally, all the implementation files can be compiled using the command below:
    ```
    $  colcon build –symlink–instal
    ```

14) Check if prior step has finished successfully with no errors, and if so, setup workspace environment variables:
```
$  source install/setup.bash
```

The environment preparation is complete with the successful compilation message. The implementation development was based on a repository that uses a different version of ROS2, which is available at https://github.com/joshnewans/articubot_one. The cloned repository contains all files related to the development of this simulation. The source code repository is available at https://github.com/lg-lima1/godot. To edit source code files, it is recommended to install the Visual Studio Code IDE, available at https://code.visualstudio.com/.

B. Robot Modeling using Unified Robot Description Format standard

For the simulation presented in this article, a total of seven joints were used, each with its own responsibility within the robotic control. The standardized modeling description eases the developer's visualization of the system's control architecture and the responsibility of each joint. Responsibility can range from visual impact only, affecting just the simulation display, to specific sensor acquisition parameters, such as the sampling rate in Hertz and the maximum distance of the LIDAR sensor, even in a simulated environment. The files for this section are located in the directory `./godot/description`.

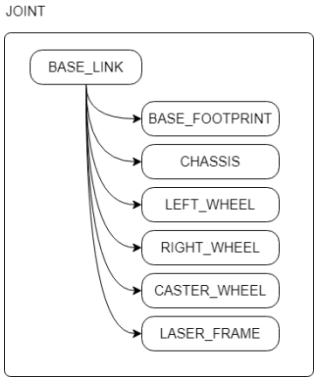


Fig. 8 Project's URDF file hierarchy, exposing the relationship between joints and links.

The purpose of the *BASE_LINK* joint is to combine all other joints, representing the robot as a whole. It does not expose any parameters or configurations and serves only as a reference for the robot's coordinate system for trajectory control systems. The joints *BASE_FOOTPRINT* and *CHASSIS* represent the physical dimensions of the robot's base in the system. The joints *LEFT_WHEEL* and *RIGHT_WHEEL* represent the left and right wheels, respectively. The term *CASTER_WHEEL* represents the features of a swivel caster. The definition of these joints can be found in the file `./godot/description/robot_core.xacro`.

The *LASER_FRAME* joint represents the characteristics of the LIDAR sensor. It defines parameters related to its geometry and dynamics, as well as configuration parameters such as angle, scanning distance, sampling rate, and samples per revolution. The definition of this joint can be found in the file `./godot/description/lidar.xacro`.

The purpose of the URDF file is to compile all joints into a single coordinate system, which is represented by the *BASE_LINK* joint. Fig. 9 visually represents the content of the URDF file, showing the relationship between the coordinate systems of each joint, it is evident that each joint has its own independent coordinate system.

Now, it is possible to start the RVIZ2 visualization tool to check the robot modeling interpreted from the URDF file:
```
$  ros2 launch godot launch_sim.launch.py
```

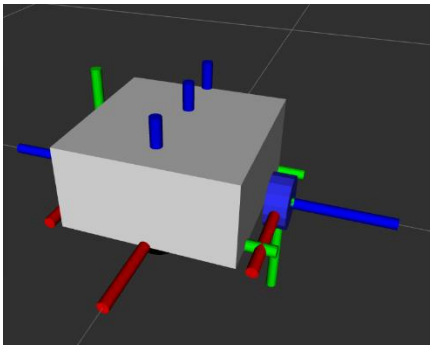


Fig. 9 Robot modeling visualization in RVIZ2.

C. Control Strategy

In practical terms, for the ROS2 framework, motion control is achieved by publishing velocity commands to the */cmd_vel* topic. This topic has an associated data type called *Twist*, which defines linear and angular velocity in all planes using six degrees of freedom of motion. However, it is up to the control strategy to determine which degrees of freedom are actually used. Until this moment, ROS2 does not have yet all the necessary information to execute the appropriate control strategy. As previously mentioned, the differential drive system is used, for that, the simulation tool Gazebo implements the control strategy through the *libgazebo_ros_diff_drive.so* plugin. The control system definitions can be found in the file `./godot/description/gazebo_control.xacro`.

Following this procedure, an efficient motion control system is effectively executed in the simulation. The following procedure illustrates how to control the robot manually in the simulation environment:

1) Start Gazebo:
```
$  ros2 launch godot launch_sim.launch.py
world:=./src/godot/worlds/obstacles.world
```
2) In a separate terminal, check and setup the ROS2 and workspace environment variables:
```
$  source /opt/ros/iron/setup.bash
$  source install/setup.bash
```
3) Run the remote *Twist* operation application. With this application is possible to jog the robot in the horizontal plane while visualizing it in the Gazebo simulation tool:

```
$ ros2 run teleop_twist_keyboard
teleop_twist_keyboard
```

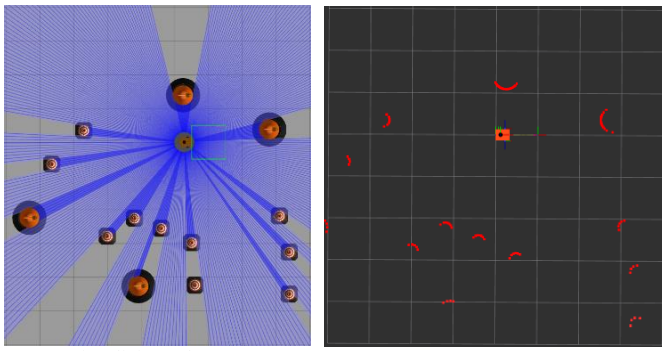


Fig. 10 Visualizing robot's movement at Gazebo and RVIZ2 using the *Twist* remote operation application to control the robot target coordinates.

D. *Autonomous Navigation System*

Two tools from the ROS2 software toolbox catalog were used to implement the autonomous navigation system. Firstly, the SLAM Toolbox was configured, which implements simultaneous localization and mapping algorithms. The parameters and configurations can be found in the file `./config/mapper_params_online_async.yaml`. To replicate the achieved results, follow the procedure below:

1) Open a new terminal and setup the ROS2 and workspace environment variables.
2) The algorithm-built map RVIZ2 visualization can be initiated using the command below. Until this moment, the map is empty because the robot did not had any environment information scanning yet.
   ```
   $  rviz2 -d src/godot/config/map_view.rviz
   ```
3) Run the SLAM Toolbox:
   ```
   $  ros2 launch slam_toolbox
   online_async_launch.py
   slam_params_file:=./src/godot/config/mapper_
   params_online_async.yaml use_sim_time:=true
   ```
4) Using the *Twist* remote operation application, jog the robot through the map as the map it is been built over the robot trajectory.
5) Again, open a new terminal and setup the ROS2 and workspace environment variables.
6) Run the Navigation2 Toolbox:
   ```
   $  ros2 launch nav2_bringup
   navigation_launch.py use_sim_time:=true
   ```

The Navigation2 tool provides various resources for generating robot trajectories. It mainly consumes data generated by the SLAM Toolbox, particularly regarding the map generated from LIDAR measurements. At this moment, we have a fully functional autonomous system.

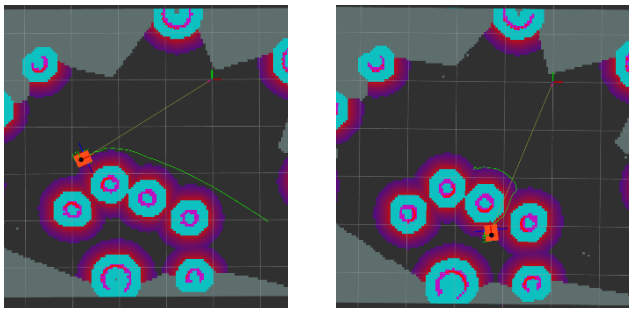
Through the RVIZ interface, target positions can be inserted for the system (x, z, φ), and the calculated trajectory can be observed. To do so, some configurations are necessary in the RVIZ tool for proper visualization:

1) Disable the visualization of the map generated by the SLAM algorithm.
2) Enable the visualization of the map generated by the navigation algorithm called heat map. This map represents the regions where the algorithm encountered obstacles, and from this, the trajectory is generated.
3) Enable the visualization of the navigation trajectory.

IV. RESULTS AND DISCUSSION

The experiments described earlier demonstrate the consistent construction of a horizontal plane map of the simulated environment without losing track of the robot's position. The localization method proved to be efficient in eliminating odometry errors by combining them with measurements taken by the LIDAR sensor.

Fig. 11 shows the navigation algorithm's results, based on the previously illustrated map. The navigation often estimated a correct trajectory to the defined target, but sometimes, especially when close to an obstacle, it was unable to generate a deviation trajectory, resulting in a collision with the nearby obstacle. Further advancement to this project is expected when using multiple LIDAR sensors in different positions.



a) b)

Fig. 11 Green line is the planned trajectory a) Visualization of the heatmap created by the algorithm identifying environment obstacles. b) Moment where a trajectory was planned but a collision with one of the obstacles happened.

The field of kinematic modeling of robots enables control systems to be more effective. The concept of odometry assumes that the kinematic behavior of the robot is known and can therefore satisfy some simpler control systems, acting as feedback for the algorithm. Once a proficient system for estimating the robot's position in the environment is achieved, it becomes possible and feasible to complement it with algorithms for localization, mapping, and navigation. In more complex cases, where odometry is combined with sensor measurements, a much more robust system is achieved, where errors from one system or the other are linearized and do not impact the overall system.

The results demonstrate the potential of open-source platforms while implementing these systems. The simulations and development presented in this article address these concepts and illustrate, from a practical perspective, the conditions for effective implementation in a simulation environment. However, it should be noted that there is still much to be explored in this area. One deficiency of the navigation algorithm was observed during the obstacle's avoidance, especially when they are very close. It is known that

better specifications of the LIDAR sensor can have a positive impact on this behavior, but the literature mentions stereoscopic camera as a possible solution.

ACKNOWLEDGMENT

Here we express our appreciation to Josh Newans, creator of Articulated Robotics blog, whose work has positively influenced the overall achievement of the result of this article.

Our gratitude extends to our mentor Dr. Vicente Idalberto Becerra Sablón, for his guidance and constructive feedback throughout the research and writing process. His expertise has been fundamental in shaping the ideas presented.

Lastly, we would like to express our deepest appreciation to families and friends for their understanding, encouragement, and patience during the demanding phases of this project.

REFERENCES

[1] J.E. Jácobo, "Development of a Versatile Mobile Autonomous Robot using Subsumption Architecture," M.S. thesis, Dept. Mechanical Eng., UNICAMP, Campinas, São Paulo, 2001, doi: 10.47749/T/UNICAMP.2001.225368. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.2001.225368.

[2] A.S. Mainardi, "Mobile robotic devices simulation with emphasis in trajectory planning for navigation," M.S. thesis, Dept. Mechanical Eng., UNICAMP, Campinas, São Paulo, 2010, doi: 10.47749/T/UNICAMP.2010.773390. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.2010.773390.

[3] L.C. Diogenes, "Use of Kalman filter and computational vision for the correction of uncertainties in the navigation of autonomous robots," Ph.D. thesis, Dept. Mechanical Engineering, UNICAMP, Campinas, São Paulo, 2008, doi: 10.47749/T/UNICAMP.2008.439669. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.2008.439669.

[4] H. Martínez-Barberá, D. Herrero-Pérez, "Autonomous navigation of an automated guided vehicle in industrial environments," Robotics and Computer-Integrated Manufacturing, vol. 26, no. 4, pp. 296-311, Aug. 2010, doi: 10.1016/j.rcim.2009.10.003.

[5] G. Bresson, R. Aufrère, R. Chapuis, "Improving SLAM with Drift Integration," IEEE 18th International Conference on Intelligent Transportation Systems, pp. 2700-2706, 2015, doi: 10.1109/ITSC.2015.434.

[6] R.A. Cordeiro, "Modeling and path tracking control of an outdoor robotic ground vehicle," M.S. thesis, Dept. Mechanical Eng., UNICAMP, Campinas, São Paulo, 2013, doi: 10.47749/T/UNICAMP.2013.909540. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.2013.909540.

[7] H. Liu, S. Li, B. Wang, "Virtual decomposition controller for flexible-joint robot manipulators with non-full-state feedback," International Journal of Advanced Robotic Systems, p. 14, 2017, doi: 10.1177/1729881417745676.

[8] A.A. Salazar, "Monocular visual navigation and sensor fusion for mobile robotics and hearing aid sensors," Ph.D. thesis, Dept. Mechanical Eng., UNICAMP, Campinas, São Paulo, 2019, doi: 10.47749/T/UNICAMP.2019.1127027. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.2019.1127027.

[9] R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza, *Introduction to Autonomous Mobile Robots: Intelligent Robotics and Autonomous Agents*, Cambridge, MA, USA: The MIT Press, 2011.

[10] R.R. Cazangi, "Uma proposta evolutiva para controle inteligente em navegação autônoma de robôs," M.S. thesis, Dept. Mechanical Eng., UNICAMP, Campinas, São Paulo, 2004, doi: 10.47749/T/UNICAMP.2004.307250. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.2004.307250.

[11] M.F. Figueiredo, "Redes neurais nebulosas aplicadas em problemas de modelagem e controle autônomo," Ph.D. theses, Dept. Elet. Engineering, UNICAMP, Campinas, São Paulo, 1997, doi: 10.47749/T/UNICAMP.1997.118281. [Online]. Available: https://doi.org/10.47749/T/UNICAMP.1997.118281.

[12] G. Klancar, A. Zdesar, S. Blazic, I. Skrjanc, *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*, Oxford, Oxfordshire, ENG: Elsevier, 2017.

[13] M. Bouazizi, A.L. Mora, T. Ohtsuki, "A 2D-Lidar-Equipped Unmanned Robot-Based Approach for Indoor Human Activity Detection," Sensors, vol. 23, no. 5, pp. 2534, 2023, doi: 10.3390/s23052534.

[14] A. Koubaa, *Robot Operating System (ROS): The Complete Reference*, vol. 1, Cham: Springer, 2016, doi: 10.1007/978-3-319-26054-9.

[15] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics, vol. 7, May 2022, doi: 10.1126/scirobotics.abm6074.