Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 6 (2024)

Virtual Design and Simulation of Autonomous Tractor

Ganesh Y. [1], P Chandan [2], Harsha S.K.V [3], Akkipeddi Pranav [4], Sharanbassappa S Patil [5]

[1] *PES University, Bengaluru, KA 560085 India*
[2] *Professor, PES University, Bengaluru, KA 560085 India*

Abstract:- This project presents the design and simulation of an autonomous tractor that can navigate through an agricultural field using Robot Operating System (ROS) and Gazebo. The tractor model is constructed using Fusion 360 and imported into ROS. The tractor uses a LiDAR sensor to perform simultaneous localization and mapping (SLAM) using the GMapping algorithm. The tractor also uses the Dijkstra algorithm and the Dynamic Window Approach (DWA) for global and local path planning, respectively. The tractor's motion is controlled using a PID controller. The simulation results demonstrate the tractor's ability to map the environment, locate itself, and avoid obstacles while following a predefined path. The project also discusses the challenges and limitations of the proposed methodology, as well as the scope for future work.

Keywords*:* Autonomous Vehicles, Autonomous Tractors, Path Planning, GMapping, AMCL, ROS, Gazebo, RViz, Dijkstra's Algorithm, DWA Path Planning, PID Controller, Simulation.

1. Introduction

One of the most important roles of Farmers face challenges in sustainably producing large crop yields on limited land. Autonomous tractors, de- signed to perform various agricultural tasks with minimal human intervention, offer a solution. These tractors increase productivity and accuracy by performing tasks routinely and repetitively. As farmland expands for diverse crops, the number of tasks increases. Laborers may not perform tasks consistently or for extended hours, impacting crop production. Autonomous tractors, capable of prolonged, consistent work, can mitigate these issues. Autonomous tractors navigate com- plex environments using path planning, sensors, and con- trollers. Path planning finds efficient routes while avoiding obstacles and minimizing distance or time. Algorithms adjust the tractor's speed and path in response to uncertainties like weather changes or obstacles. Sensors, such as LiDAR, enable independent operation and navigation. Controllers, like the PID controller, adjust the tractor's control. A virtual design and simulation of an autonomous tractor have been created to test this theory. The tractor model, constructed using Fusion360, and farmland, created in Gazebo, facilitate efficient and robust navigation using the Dijkstra algorithm and Dynamic Window Approach (DWA) with ROS. The simulation uses RviZ and Gazebo interfaces, demonstrating the potential of autonomous tractors in modern agriculture.

2. Literature Survey

The summaries collectively detail a wide array of autonomous vehicle navigation techniques implemented in the project, with a primary focus on Simultaneous Localization and Mapping (SLAM) algorithms. These techniques encompass graphical SLAM, explore and return strategies, directed sonar navigation, and the integration of adaptive algorithms like Adaptive Monte Carlo Localization (AMCL). The incorporation of diverse sensors such as LiDAR and odometer is noteworthy, with an emphasis on fusing sensor data through particle filtering and optimal estimation method called Kalman filter. The project consistently integrates common elements across these methodologies, emphasizing the significance of correlations within SLAM algorithms, real-time implementation, and optimization for virtual deployment. The efficacy of SLAM algorithms is underscored by their ability to simultaneously map and localize, a crucial aspect for achieving autonomous navigation. Furthermore, the summaries delve into path planning strategies, including the use of the Dijkstra Algorithm, Dynamic Window Approach (DWA), and a fusion of A* and DWA to facilitate comprehensive navigation in dynamic environments.

The recurring theme of utilizing the ROS is evident, offering an effective framework for developing, testing, and deploying robotic applications. Overall, these papers help advance robotics by employing various approaches. They emphasize the significance of accurate sensor data, efficient mapping, and reliable localization as pivotal factors for achieving successful autonomous navigation across various scenarios in the project.

3. Methodology

The first step of this project was to pick the right software to run the simulation, making sure it was compatible with the available resources. Upon trying out different softwares like MATLAB and Simulink coupled with Unreal engine or Unity, ROS (Robot Operating System) met our requirements. This software is widely used to run robots and autonomous vehicles. It offers a large collection of reusable software components, called stacks, that implement common functionality, such as navigation, perception, manipulation, planning, and simulation. Stacks can be composed of multiple packages and depend on other stacks. It also provides a set of tools for debugging, testing, visualization, and introspection of the ROS system. An in-built package called Gazebo, which is a 3D simulator that can simulate complex and realistic environments is used to depict the farmland (virtual environment) and run the simulation as well.

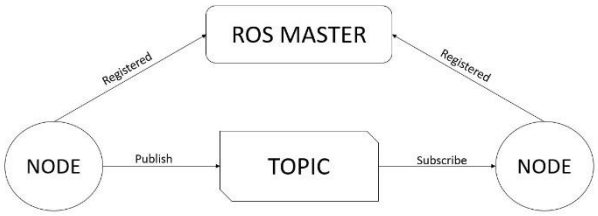

Fig. 1. ROS

3.1 Tractor and Farmland Model

The next step was to model a tractor. Modelling application

'Fusion 360' was used to construct the model as per the necessities of the project. Since a full-scale model could not be run in the software, a decision of scaling it down to the ratio of 1:3 was made. The new holland tractor model 90TL was used as reference in modelling the tractor. The dimensions of the tractor are listed below:

- Height 0.670 m
- Track width 0.660 m
- Front surface of body to end: 0.710 m
- Wheelbase 0.578 m

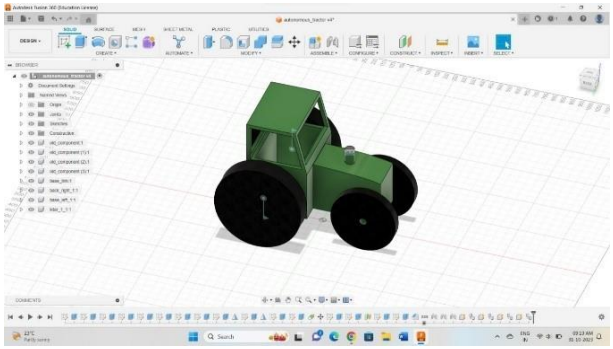

Fig. 2. Fusion 360 Model of Tractor

A farmland was modelled on which the simulation was run, using the Gazebo interface. The field dimensions are 30 m x 30 m.

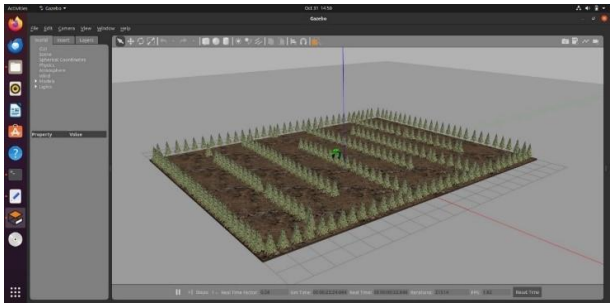

Fig. 3. Farmland Model

3.2 URDF

To implement these models, they had to be converted to the URDF format and certain parameters were to be specified. The URDF extension stands for URDF (Unified Robot Description Format). It is an XML file format for representing the physical properties of tractors in Robot Operating System (ROS). It is used to specify characteristics of robots like geometry of the robot, materials used, Colour, joints, type of joints and various other physical properties of the tractor, which are required in simulation, control, and path planning for tractors.

- URDF describes robots as a hierarchical tree, where relationships between different links are specified based on parent and child relation. In the sense each link either acts as a parent link or a child link.
- The geometric shapes of each link must be specified in URDF to get detailed visualizations and collision detection simulations.
- URDF defines the joints connecting different links, including their types- revolute, prismatic, etc. and axis of rotation. Which helps in the calculation of robot kinematics, such as the position and orientation of each link relative to other.
- Various physical properties like mass, inertia and various others of each link are specified in URDF. Allowing for dynamic simulations and realistic simulations.
- For this project multiple files related to URDF were created which include plugins file which included controller and lidar configurations, materials and physical properties used for the autonomous tractor.

3.3 PID Controller and LiDAR Configuration

A Proportional-Integral-Derivative (PID) controller is a common type of controller used in ROS. It's primarily located in the controller node, which is the central node in the package. Here's a simplified explanation of how it operates:

- Setpoint: This is the goal that the system aims to achieve. It's typically provided externally, either manually or by a higher-level control system.
- Process Variable: This is the parameter that we want to control. It could be various things like temperature, flow rate, pressure, rotation speed, etc. In our case, it's the tractor's speed.
- Error Calculation: The controller compares the measured process variable with the setpoint. The difference between these two values is used to compute a control signal. This signal is then sent to the actuation device, which adjusts the system to reach the desired value (setpoint). Essentially, this error value helps determine how much the output needs to be adjusted to bring the actual reading closer to the setpoint.

A LiDAR of the following characteristics was used.

- Field of view: 180 degrees
- Range: 20 meters
- Frequency: 100 Hz
- Accuracy: +-30 mm

In this way, LiDAR helps in environment mapping by creating detailed 2D grid-based maps and providing accurate data for digital elevation models and features like crop rows and obstacles like boulders/rocks. It also aids in location pinpointing by allowing each measurement in the point cloud to be georeferenced.

```
<gazebo>
  <plugin name="differential_drive_controller" filename="libgazebo_ros_diff_drive.so">
    <updateRate>10</updateRate>
    <leftJoint>Revolute1</leftJoint>
    <rightJoint>Revolute2</rightJoint>
    <wheelSeparation>0.280</wheelSeparation>
    <wheelDiameter>0.450</wheelDiameter>
    <wheelAcceleration>1.0</wheelAcceleration>
    <wheelTorque>20</wheelTorque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>base_link</robotBaseFrame>
    <odometrySource>1</odometrySource>
    <publishWheelTF>true</publishWheelTF>
    <publishOdom>true</publishOdom>
    <publishWheelJointState>true</publishWheelJointState>
    <rosDebugLevel>na</rosDebugLevel>
    <odometrySource>world</odometrySource>
    <publishTf>1</publishTf>
    <publishOdomTF>true</publishOdomTF>

    <!-- Set to true to swap right and left wheels, defaults to true -->
    <legacyMode>false</legacyMode>
  </plugin>
</gazebo>
```

Fig. 4. Controller Configuration 1

Fig. 5. Controller Configuration 2

3.4 Co-Simulation between Gazebo and Rviz

The meta operating system ROS was utilized for the simulation, incorporating two interfaces: RViz (Robot Visualization) and Gazebo. Gazebo serves as a simulator enabling users to construct a virtual world where the tractor is positioned, while RViz functions as a tool for presenting sensor data and state information from the tractor. RViz does not replicate the world or the tractor, but rather visualizes the tractor's perspective. The tractor model utilized in the simulation is a faithful reproduction of a real New Holland tractor. The virtual tractor model is designed using Fusion 360 and exported to ROS in URDF file format. The tractor is equipped with a lidar module (RP lidar) for self-directed movement. The URDF files encompass crucial details regarding the tractor such as track width, component masses, moment of inertia, as well as lidar specifications (coverage angle, range, frequency). A virtual world was established, containing the tractor model and a crop field situated on an irregular terrain. The co-simulation between RViz and Gazebo functions in a manner where Gazebo simulates the robot and its surroundings, while RViz visualizes the sensor data and state of the tractor. The left-hand window displays the Gazebo interface, showcasing LiDAR scanning and transmission of resampled data to the controller, aiding the DWA path planner in determining the most efficient path for the tractor. On the right-hand side, the RViz interface exhibits the visualization of tractor movement and the mapped environment. While these front-end operations unfold, in the background, ROS manages multiple ROS topics continuously, resampling data at regular intervals and devising new path plans for the tractor.

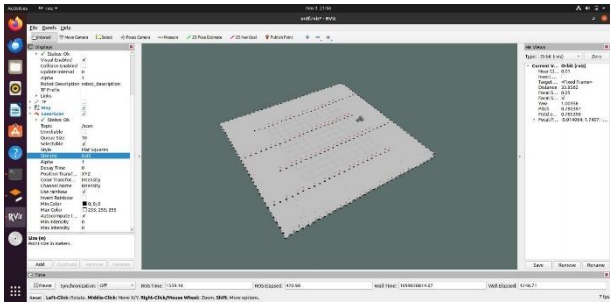

Fig. 6. LiDAR Scan

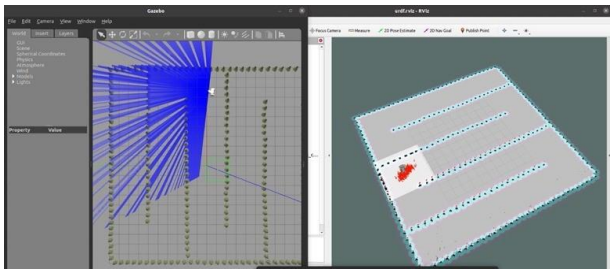

Fig. 7. Co-Simulation between Gazebo and RViz

3.5 GMapping

The GMapping algorithm is a filter-based optimization algorithm used in SLAM. It separates localization from mapping, it first locates the tractor, then mapping the surrounding environment. The process of GMapping has been explained as follows:

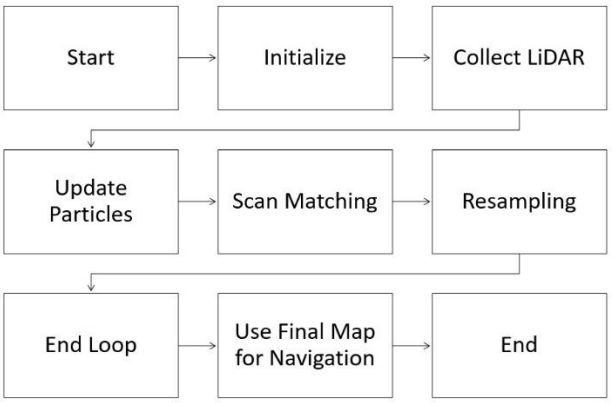

Fig. 8. Process of GMapping

- Initialize Map and Particles: At the start, GMapping initializes an empty map and a set of particles representing possible locations of the tractor. Each particle has a state consisting of our tractor (position and orientation) and the map.
- Collect Lidar Data: Lidar data is collected, which is a set of measurements of distances to nearby objects. These measurements were used to update the map and particles.
- Update Particles: For every particle, GMapping predicts the tractor's movement based on its current state and the control inputs (velocity and turning rate). This prediction is used to update the state of the particle.
- Scan Matching: After creating a predicted map, GMapping tries to align the Lidar data with it. This includes comparing the Lidar data with the areas of the map that are expected to match the measured distances. The outcome is a score that shows how accurately the Lidar data aligns with the map.

- Resampling: GMapping conducts resampling based on these scores, whereby particles with higher scores are more likely to survive, while those with lower scores may perish. This process involves updating the set of particles and their respective weights.
- End Loop: The iterations are repeated until a certain stopping condition is met, such as reaching a specific time threshold or maximum number of iterations.
- Use Final Map for Navigation: Ultimately, GMapping utilizes the final map and the particle state to steer the robot during navigation. The map furnishes the robot with information regarding its environment, while the particle state indicates the tractor's perceived location.

While the flow chart explains how the process of the GMapping works the math's behind it has been explained as follows:

- Particle Update: The tractor's motion is modelled as a random walk with Gaussian noise. The update equation for a particle xi is given by:

$x_i = x_i + v * \delta_t + w * sqrt(\delta_t) * N(0, Q)$

where v is the tractor's velocity, δ_t is the time interval, w is the turning rate, N (0, Q) is a normally distributed random variable with zero mean and covariance Q, and x_i represents the current state of the robot.

- Scan Matching: Given a new LIDAR scan, l problem is solved using a fast and accurate approach called FastSLAM. The equations for FastSLAM are complex and involve many steps, but they essentially involve updating the belief about the tractor's pose for each particle, given the observed LIDAR readings.
- Resampling: After each update, the set of particles is resampled according to their weights. The weight of a particle is proportional to the likelihood of the observed data given the particle's state. The resampling equation is given by:

  $w_new = w_old (- \beta * (log(w_old) - log(sum(w_old)))$ where w_new is the new weight of a particle, w_old is the old weight of the particle, beta is a parameter that controls the degree of resampling, and log is the natural logarithm function 1.

These equations form the basis of GMapping's operation. They allow the system to update its belief about the tractor's location and build a map of the environment over time.

3.6 Adaptive Mote Carlo Localization

In the domain of simulating autonomous tractors, precise localization is pivotal for effective navigation and task execution. AMCL, an extension of the Monte Carlo Localization (MCL) algorithm, has demonstrated significant advancements in this area. This subsection explores the mathematical foundations of AMCL and its relevance to the virtual simulation of autonomous tractors. The working behind the same has been explained as follows:

- Particle Representation: AMCL Utilizes a set of Particles
- (X_t) to represent the tractor's pose, with each particle (x_i) encapsulating the tractor's pose (x, y,) and a corresponding weight (w_i) denoting its likelihood
- Prediction Step: The prediction step updates the particle set based on the tractor's motion model: $x_t^i = motion - model(u_t, x_{(t-1)}^i)$

Here u_t is the control input at time t and $x^i{}_t$ signifies the predicted pose of particle i.

- Weight Update Step: Weights are recalibrated through the measurement model and sensor observations (Z_t) $wti = 1/zt.\ p\ (zx|xit,\ m).\ w-i(t-1)\ p(Z^x|x^i{}_t,m)$ represents the likelihood of the sensor measurement given the particle's pose and the map m, (Z_t) is a normalization constant.
- Resampling Step: To maintain diversity a resampling step is induced:

Resample ((X_t))

This step replicates particles with higher weights, ensuring a representative sample.

- Adaptation of Sample Size: A key feature of AMCL is the adaptive adjustment of sample size based on the Kullback-Leibler Divergence (KLD). It dynamically calculates the number of particles (n) needed to represent the posterior distribution, ensuring the error between true and sample-based distributions remains below a predefined threshold.

- Space Division through KD Tree: AMCL incorporates a KD tree for spatial distribution analysis, dynamically adjusting bin sizes in the adaptive sampling method.

In simulating autonomous tractors, AMCL emerges as a robust solution, offering a balance between approximation error and runtime efficiency, with dynamic sample size adaptation based on spatial distribution.

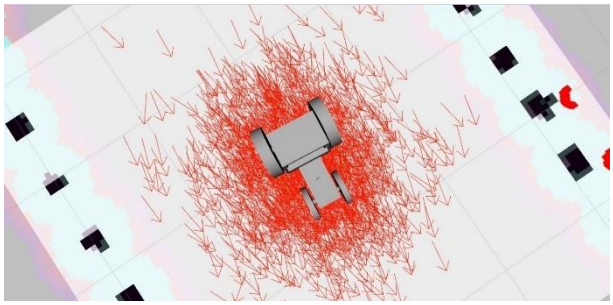

Fig. 9. AMCL Particles Visualization

3.7 Global Path Planning

For Global path planning Dijkstra algorithm was chosen after researching the different algorithms and their compatibility for large vehicles such as our tractor. Dijkstra algorithm is a type of path planning algorithm that is used to create shortest obstacle free path from start point to end point by taking vehicle configuration into consideration. It is basically used to create a static path on an occupancy grid map designed using sensors like lidar or a combination of Ultrasonic and IMU (Inertial Measurement Unit) sensors. This algorithm finds all possible paths from source to destination without neglecting any gird so has to find the most possible shortest path based on cost (time and distance). One of the reasons for using this algorithm is the customization it offers like different costs can be assigned to different terrains which is an added advantage to our project as it deals with different terrains. This algorithm doesn't need heuristics unlike other path planning algorithms like A* which makes it suitable for applications where functions cannot be defined accurately. It even comes with few disadvantages like it cannot be used for more complex or dynamic path planning and it cannot be used for very large maps which can be neglected for our use case. Various other global path planners like Carrot planner can be used for path planning. Carrot planner is the simplest path planning algorithm which creates path from start point to end point without considering any obstacles which is a major disadvantage of Carrot planner.

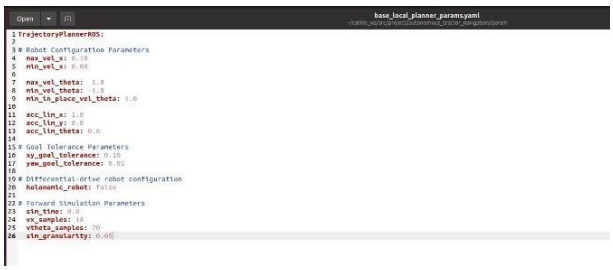

Fig. 10. Base Local Planner Parameters

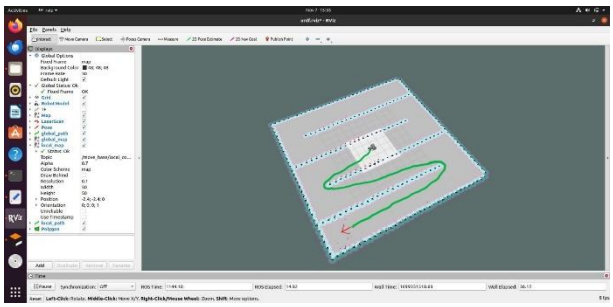

Fig. 11. Global Path

3.8 Local Path Planning

For local path planning DWA path planning algorithm has been used. It is a path planning strategy that can produce smooth and collision-free trajectories for the tractor in realtime. It operates by choosing the optimal velocity command from a set of permissible velocities that fall within the tractor's dynamic constraints and sensor range. The optimal velocity command is one that maximizes an objective function considering factors like distance to the target, obstacle clearance, and alignment with the global path. Since the DWA Planner is mostly used for mobile robots, it takes into account the tractor's kinematics. Robot kinematics is a simple mathematical model that is used to describe the position of the robot. For a general mobile robot, one can consider the case of 2D robot Kinematics.

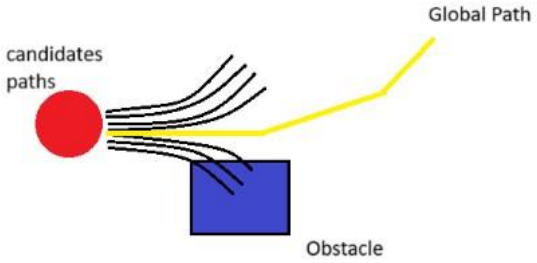

Fig. 12. DWA Path Planning

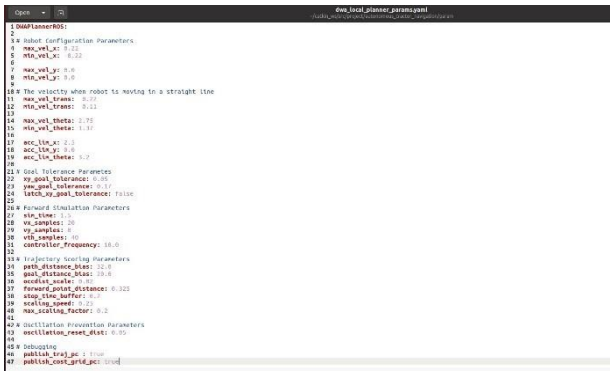

Fig. 13. DWA Local Planner Parameters

4. Results and Discussions

Results:

- Mapping of the environment and localization of the vehicle were successfully accomplished through the utilization of sensor data. The precise identification of the vehicle's position on the map provided a comprehensive understanding of its spatial relationship with its surroundings.
- For the purpose of charting a trajectory for the vehicle, sophisticated algorithms were employed. These algorithms factored in various constraints of the vehicle, including speed and turning radius, to generate an optimal and efficient path.
- To ensure safe maneuvering around obstacles, specific motion control techniques were developed. Through rigorous testing across different scenarios, these methods demonstrated effectiveness in averting collisions.
- The integration of RViz and Gazebo facilitated a co-simulation framework, enabling real-time visualization of the vehicle's motion and the surrounding environment in both platforms.
- Customization of algorithms was carried out to address specific limitations, enhancing the vehicle's performance in particular situations.

- Applications such as sowing, fertilizing, harvesting, and weed removal could be carried out using various attachments on the tractor, which constitutes a key element of the project aimed at benefiting farmers.

Discussion:

The project's outcomes suggest that the set goals were effectively accomplished. The creation of an environment map and the precise localization of the vehicle formed a strong basis for the subsequent stages of the project. These foundational elements played a crucial role in the development of path planning and motion control methods, which are key to the autonomous operation of the vehicle.

Path planning is a technique that involves determining an efficient route from the starting point to the destination while avoiding obstacles and minimizing distance or time. This is particularly important as autonomous vehicles need to navigate through complex environments, which can include uncertainties such as sudden changes in weather, fallen trees, or animals on the path. Algorithms are used to adjust the vehicle's speed and path in response to these uncertainties.

Sensors enable the autonomous vehicle to operate independently and navigate through a variety of environments. In this case, a LiDAR sensor is used. Controllers, on the other hand, adjust the lateral, longitudinal, and steering control in the autonomous vehicle. Among the various controllers used for autonomous agents, a PID controller has been found to be the most effective for this purpose.

A significant aspect of the project was the co-simulation between RViz and Gazebo. This setup allowed for real-time visualization of the vehicle's movements and the environment, proving to be an invaluable tool for testing and fine-tuning the vehicle's navigation capabilities. The ability to visualize the vehicle's movements in relation to its environment greatly facilitated the process of troubleshooting and optimizing the vehicle's performance.

Another noteworthy achievement was the personalization of the algorithms used in the project. By customizing these algorithms to tackle specific constraints, the performance of the vehicle was significantly improved. This personalization allowed for a more flexible and adaptable system, capable of handling a wider range of scenarios and conditions.

Looking ahead, there is potential for further work and improvements. Future efforts could focus on optimizing these algorithms even further, as well as exploring additional methods for obstacle avoidance and path planning. This approach demonstrates the potential of autonomous vehicles in modern agriculture. By creating a real-life model based on the virtual scenario, we can achieve even more validation.

References

[1] Saravanakumar D, Sakthivel G, Jegadeeshwaran R and Marshal Revanth, Simultaneous Localization and Mapping of Mobile Robot using GMapping Algorithm, International Symposium on Smart Electronic Systems, 2020.
[2] Dhruv Talwar and Seul Jung, Particle Filter-based Localization of a Mobile Robot by Using a Single LiDAR Sensor under SLAM in ROS Environment, International Conference on Control, Automation and Systems, 2019.
[3] Stephanie Bonadies and S. Andrew Gadsden, An overview of autonomous crop row navigation strategies for unmanned ground vehicles, Engineering in Agriculture, Environment and Food, 2019.
[4] Fabio Ugalde Pereira, Marco Antonio de Souza Leite Cuadros, Anselmo Rafael Cukla and Pedro Medeiros de Assis Brasil, A Study on Global Path Planners Algorithms for the Simulated TurtleBot 3 Robot in ROS, Institute of Electrical and Electronics Engineers, 2020.
[5] Jianxiang Liu, Xiangpeng Kong, Fuxiang Zhang, Yulu Wu, Ling Li and Yunfend Sun, Research on Path Planning of Mobile Robot by Fusisng A* and DWA Algorithms, International Conference on Intelligent Computing and Signal Processing, 2023.
[6] Shengmin Zhao and Seung-Hoon Hwang, Path Plannning of ROS autonomous Robot based on 2D LiDAR-based SLAM, International Conference on Information and Communication Technology Convergence, 2021.

[7] Marco C. De Simone, Domenico Guida and Zandra B Rivera, Unmanned Ground Vehicle Modelling in Gazebo/ROS-Based Environments, MDPI Machines, 2019.
[8] Anis Koubaa, Robot Operating System (ROS), Springer, 2021.
[9] Anandu Rajendraprasad, Sakthiprasad Kuttankulangara Manoharan and Rajesh Kannan Megalingam, Comparison of Planned Path and Travelled Path Using ROS Navigation Stack, International Conference for Emerging Technology, 2020.
[10] D. B. Ampratwum, Atsu S S Dorvolo and Umezuruike Linus Opara, Usage of Tractors and Field Machinery in Agriculture in Oman, Agricultural Engineering International: the CIGR Journal of Scientific Research and Development, 2004
[11] Sun Dihua, Qin Hao, Zhao Min, Cheng Senlin and Yang Liangyi, Adaptive KLD Sampling based Monte Carlo Localization, IEEE, 2018