**DESIGN AND DEVELOPMENT OF SEMI-AUTONOMOUS MOBILE ROBOT FOR RAKING OF CASHEW NUTS DURING SUN DRYING PROCESS**

*A Project Report submitted to Manipal Academy of Higher Education in partial fulfilment of the requirements for the course*

**ROS LAB MTE**

*Submitted by*

**N.DHANYASHREE (220972006)**

**ADITHYA GOLWALKAR.S.R (220972012)**

**NANDAN UMRIKAR (220972004)**

**2nd SEMESTER - M. TECH**

**INDUSTRIAL AUTOMATION AND ROBOTICS**

DEPARTMENT OF MECHATRONICS ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY, MANIPAL-576104

2023 MAY

Diagram

Description automatically generated with medium confidence

**ACKNOWLEDGEMENT**

It has been an honour and privilege to have pursued our **M.Tech Industrial Automation and Robotics** degree in the esteemed **Manipal Institute of Technology, Manipal**. We express our profound gratitude to the Department of Mechatronics for providing us with all the necessary support and facilities to meet our project requirements.

We are grateful to our Head of Department Dr.D.V.Kamath, Ph.D. for offering us the entitlement to experiment and practise the concepts in laboratories. Our heartfelt thanks to our lecturers , Dr.Pooja Nag, Associate Professor , Mr. Sivayazi Kappagantula, Assistant Professor, Assistant Professor who took time out of their schedule to attend our queries and aid us to complete our project. Last but not the least our heart full thanks to our classmates for supporting during the journey.

We are extremely thankful and fortunate to all the concerned people without whom this project would not have been a success. We are most obliged.

Thank you, Manipal Institute of Technology.

Thank you all.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO.** |
| **1. Introduction** | **2** |
| **2. Literature review** | **3** |
| **3. Problem statement** | **6** |
| **4. Objectives** | **6** |
| **5. Methodology** | **6** |
| **6. Algorithms Used** | **9** |
| **7. Conclusion** | **9** |
| **8. References** | **10** |

**1. Introduction**

The cashew seed is commonly considered a snack nut (cashew nut) eaten on its own, used in recipes, or processed into cashew cheese or cashew butter. Like the tree, the nut is often simply called a cashew. Cashew allergies are triggered by the proteins found in tree nuts, and cooking often does not remove or change these proteins. In 2019, four million tonnes of cashew nuts were produced globally, with Ivory Coast and India as the leading producers. As well as the nut and fruit, the plant has several other uses.

Before the cashew nuts can be collected, they must first undergo the sun-drying process. This involves cleaning and sorting the raw cashew nuts to remove any debris or damaged nuts. The nuts are then spread out in a thin layer and placed in an open, sunny area with good air circulation. The drying process can take several days to a week, depending on the weather and the desired moisture content of the nuts. The nuts are considered fully dried when they reach a moisture content of around 8%.

Sun drying is a common method for drying raw cashew nuts, which are the edible seeds of the cashew tree. The process involves exposing the nuts to sunlight and air for a period to reduce their moisture content. This involves the raking or ploughing of the dried cashew nuts in the ground after they have been sun-dried. The raking process is a tedious and time-consuming task that can benefit from automation. To this end, we propose the use of ROS2, a robot operating system, to simulate the cashew nut raking process using SLAM and Nav2.

Once the cashew nuts have been dried, they are ready to be collected. The traditional method of collecting the nuts involves manual labour, which can be time-consuming and labour-intensive. To automate this process, we propose the use of mobile robots equipped with SLAM and Nav2.

SLAM (Simultaneous Localization and Mapping) is a technique used by mobile robots to map their environment and determine their location within that environment. It involves the use of sensors such as LiDAR, cameras, and IMU (inertial measurement unit) to create a map of the robot's surroundings. Nav2 is a navigation system for mobile robots that uses the maps created by SLAM to plan the robot's path and avoid obstacles.[1]

By using SLAM and Nav2, we can simulate the cashew nut raking process in a virtual environment using ROS2. The mobile robot would be equipped with sensors and actuators to rake the dried cashew nuts in the ground. The robot would also be able to navigate through the environment using the map created by SLAM and avoid obstacles using the path planning system of Nav2.[2]

Simulation of the cashew nut raking process using ROS2 can help to reduce labour costs and increase productivity on real time application. The virtual environment would allow us to test different robot designs and algorithms without the need for physical prototypes. Additionally, the simulation would provide a safe and controlled environment for testing, which would reduce the risk of damage to the robot or surrounding objects.

**2. Literature review**

Farmers must sun dry the raw cashews for at least 2 days, turning them frequently to ensure uniform drying and keep moisture content between 7 and 10 percent. Handling raw cashews can be dangerous because they release a toxic oil-like substance. Lack of sorting, drying, and farm management can affect the quality of the nuts and lead to mycotoxin contamination. The shell of the cashew contains the toxic oil urushiol, which can cause dermatitis on contact.

**2.1 Slam toolbox**

With the help of mobile Intel CPUs, which are frequently found on robots larger than 100,000 f t2, SLAM Toolbox can map spaces effectively. Typically, it is simple to complete employing untrained technicians contracted to install robot solutions or monitor systems remotely. Applications have been developed that use the SLAM Toolbox and exploration planners to automatically map a space.

To continue enhancing or expanding an existing map, it is also possible to serialise a current mapping session and deserialize it later. Instead of saving submaps like in Cartographer, this serialisation preserves the entire raw data and pose-graph, enabling the creation of a number of unique tools and more precise multi-session mapping.

These tools also allow for manual posegraph manipulation, as seen in Figure 2, where a user can rotate a map or help with a difficult loop closure by manually manipulating the pose-graph nodes and data. The process of combining numerous serialised maps into a composite map is known as kinematic map merging. Additionally, a 3D visualizer plugin was developed to help with the use of these tools and the basic SLAM library features. This representation also allows for the development of numerous more tools and utilities.[3]

***2.1.1 Operational modes***

Synchronous mapping, asynchronous mapping, and pure localization are the three main operational modes and executables offered. Synchronous mapping makes it possible to map and localise in a space while maintaining a measurement buffer to add to the SLAM issue. When undertaking offline processing or when the map's quality is particularly important, this can be beneficial. The asynchronous mode, in contrast, will handle new measurements after the previous measurement is finished and the new update requirements are satisfied. As a result, while executing complicated loop closures, this will never lag behind real-time. However, if processing the final measurement takes too long, the map might not contain all legitimate measurements. When the accuracy of real-time localisation is crucial, this option is advantageous.

It is possible to utilise either of these modes for multi-session SLAM, which involves loading an earlier session and continuing to hone the pose-graph. A map of a sizable office complex is shown in Figure 3 after being partially drawn in one session and finished in another. The pure-localization method was later utilised with this map, which contains numerous loop closures between the two datasets, to enable autonomous navigation.[4]

**2.2 Navigation2**

***2.2.1 Action servers***

Long-running processes like navigation are frequently managed by action servers. Like a canonical service server, they let clients call long-running tasks executing in different processes or threads and return a future to the task's outcome. Feedback and outcomes can be acquired asynchronously by asynchronously requesting information from the shared future objects or synchronously by registering call-backs with the action client. In this stack, action servers are used to calculate plans, control efforts, and recoveries by communicating with the highest-level BT navigator using a NavigateToPose action message and with the BT navigator's following smaller action servers. For communication with the servers, each action server has a specific unique action type in nav2\_msgs.

***2.2.2 Lifecycle Nodes and Bond***

The ROS 2 specific Lifecycle Nodes and Bond contain state machine transitions for the startup and shutdown of ROS 2 servers. A node is first started in an unconfigured state, processing simply the constructor. The nodes must be configured to go inactive by the launch system or the provided lifecycle management. After that, the node can be activated by moving through the activating stage. Deactivating, tidying up, shutting down, and ending in the finalised state are the steps leading up to shut down. It also has a bond link for the lifecycle manager to make sure that a server stays active after transitioning up.

This project makes substantial use of the Lifecycle Node framework, which is used by all servers. To simplify the complexity of Lifecycle Nodes for common applications, Nav2 uses a wrapper called nav2\_util Lifecycle Node.

***2.2.3 Behaviour trees***

Complex robotics jobs are increasingly using Behaviour Trees (BT). They define a more scalable and intelligible framework for defining multi-step or many state applications since they are a tree structure of activities that need to be accomplished. They offer a formal framework for navigation logic that can be utilised to build sophisticated systems as well as be checked and verified using cutting-edge techniques to ensure its accuracy. We utilise BehaviorTree CPP V3 as the behaviour tree library for this project. With the help of cutting-edge technologies, BTs offer a formal structure for navigation logic that can be utilised to build sophisticated systems as well as be verified and validated as being true.

***2.2.4 Navigation servers***

* Planners

Planners are responsible for computing a path to complete a certain objective function. The objective could be to move the robot from its current position to a goal position or to cover all the free space in the environment. Planners have access to a global environmental representation and sensor data. They can be designed to compute various types of paths, such as the shortest path, complete coverage path, or predefined routes.

* Controllers

Controllers, also known as local planners, are responsible for following the path computed by the global planner or completing a local task. They have access to a local environmental representation to compute feasible control efforts for the robot's base to follow. Controllers can be designed for various tasks, such as following a path, docking with a charging station, boarding an elevator, or interfacing with a tool.

* Smoothers

Smoothers are used for additional quality improvements of the planned path. They can be designed to optimize the path to make it smoother and easier for the robot to follow. Smoothers operate on the planned path computed by the planner and can be used to improve the path's optimality.

* Recovery Servers

Recovery servers are used to get the robot out of a bad situation or attempt to deal with various forms of issues to make the system fault tolerant. Recovery behaviours are a mainstay of fault-tolerant systems. The goal of recoveries is to deal with unknown or failure conditions of the system and autonomously handle them. Examples may include faults in the perception system resulting in the environmental representation being full of fake obstacles or if the robot was stuck due to dynamic obstacles or poor control. Backing up or spinning in place, if permissible, allows the robot to move from a poor location into free space it may navigate successfully. In the case of a total failure, a recovery may be implemented to call an operator's attention for help.

**3. Problem statement**

Due to uneven and untimely turning of raw cashew nuts during its sun drying process, cashew nuts are not being dried evenly resulting in its contamination due to presence of excess humidity. This contamination leads to losses for the farmers and needs to be mitigated. Manually raking of cashew nut needs to be done under hot sun and the nuts release toxic oil like substance it is a very tiresome and dangerous job. The same needs to be simulated in virtual model where the robot will move from set point to the goal point, covering the nuts along the path.

**4. Objectives**

* Design CAD model of mobile robot to rake the nuts on the ground.
* Implement the path planning algorithm using Nav2.
* Simulate the robot using SLAM and Nav2 toolbox in ROS2.

**5. Methodology**

Following process is followed for the design and manufacturing methods:

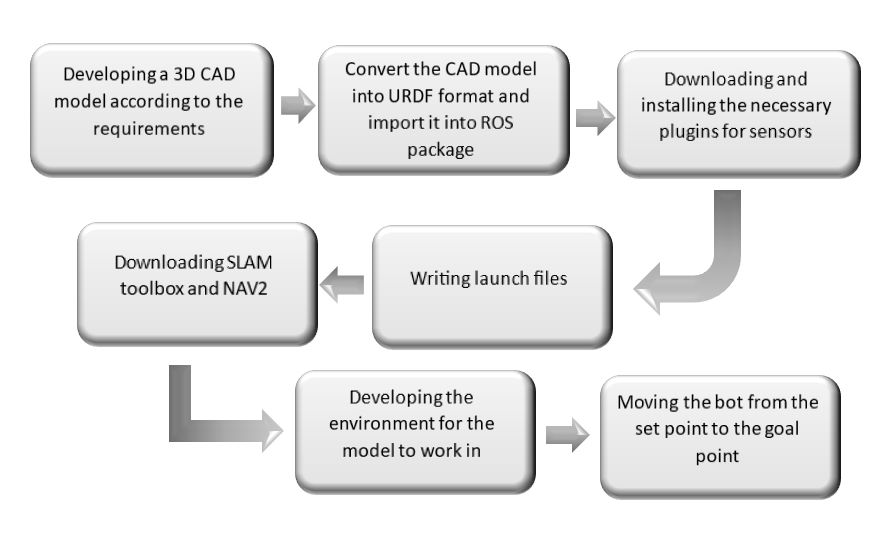
****

Fig Flow chart of the simulation process

1. **Developing CAD model**

Analysing various design requirements for the mobile robot and creating a blue print of the design, various components and parameters are included and designed by following method:

a. Wheels

Wheel design can’t be an existing one since the application demands specially designed wheels which are constructed by connecting two circular discs with rods at their circumference. It enables the robot to plough through the cashew nuts which would be about 4cm thick layer on the ground, without damaging the nuts. it also prevents the slippage of tires due to nuts.

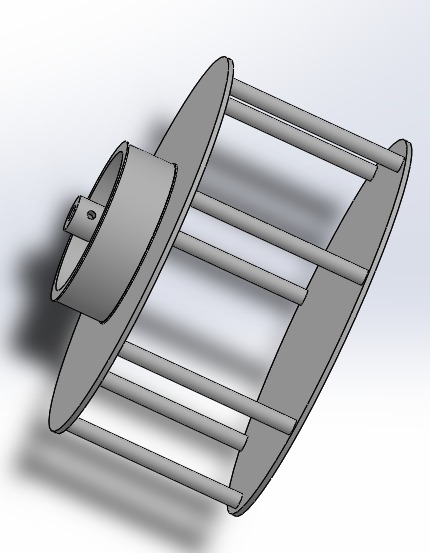


Fig Cad model of the wheel

b. Rake design

The gap between the tooth and the size of each teeth is decided after thoroughly studying about cashew sizes and average length and breadth. Width of the rake is obtained by studying speed of the bot and the area that has to be covered in the given stipulated time.

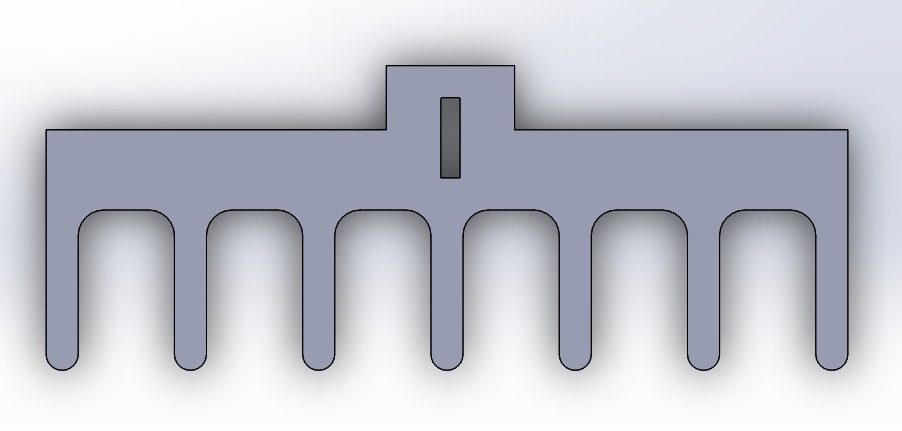


Fig Cad model of rake

c. Body dimension

After considering the various components that go into the bot and studying the required width of the rake, length and breadth of the robot is decided. Plexi glass is used for the construction of the body due its light weight compared to metals and their ability to be machined easily.

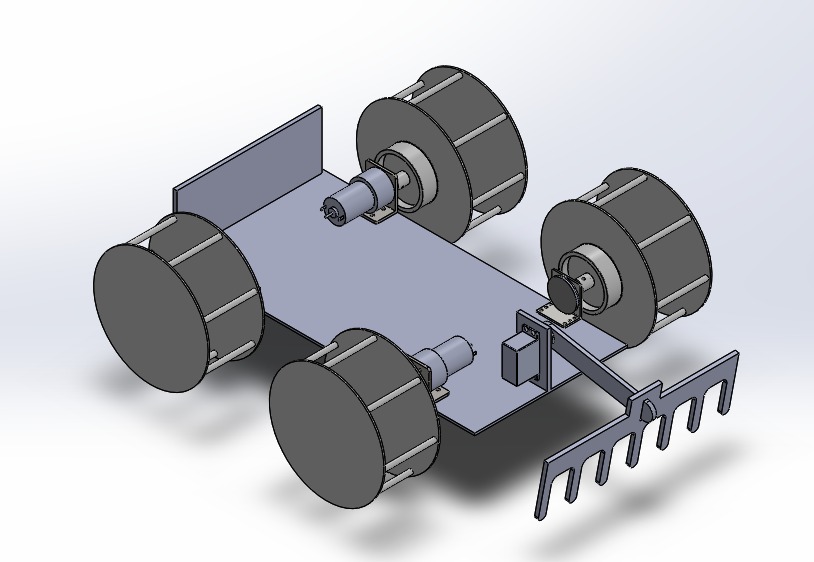
****

Fig CAD Model of prototype

d. Ground clearance

Since the nuts are dried in a way that the thickness of cashew layer is around 4cm the ground clearance is given 7.5cm to be on safe side.

1. **Exporting the CAD design of mobile robot:**

The URDF file of the CAD model is exported to the ROS2 workspace, where it will be used to create the simulation package. The package will include all the necessary files and dependencies required to simulate the robot.

1. **Downloading and installing necessary plugins for sensors:**

The robot will be equipped with various sensors, such as LiDAR, IMU and a camera, which will be used to map the environment and detect the moisture level of the cashews. Necessary plugins for these sensors will be downloaded and installed in the ROS2 workspace.

1. **Writing launch files:**

Launch files will be written to start the simulation environment and launch the required sensors and nodes. The launch file will also include the robot model and its controllers.

1. **Downloading the SLAM toolbox and Nav2:**

To perform the task of ploughing the drying cashews, the mobile robot needs to navigate around the environment and avoid obstacles while simultaneously building a map of the area. This is where the SLAM toolbox and Nav2 come in.

The SLAM toolbox in ROS2 provides several algorithms for simultaneous localization and mapping (SLAM), which is the process of building a map of an unknown environment while simultaneously localizing the robot within that map. The specific algorithm used by the SLAM toolbox in ROS2 depends on the selected package and configuration.[3]

Nav2, on the other hand, is a navigation stack in ROS2 that provides a set of navigation-related functionalities, such as localization, path planning, and obstacle avoidance. Nav2 is primarily focused on navigation and path planning, rather than map building, but it can use SLAM-related functionalities, such as localization, to improve navigation accuracy. SLAM toolbox and Nav2 is installed for simulation.

1. **Environment creation:**

The environment is be created using RVIZ, a 3D modelling software. A typical village ground will be modelled, with trees and bushes as obstacles. The ground will be modelled to resemble a cashew farm, and the unevenness in the bare ground is embedded into the design. The environment will then be exported as a URDF file, which will be used in the simulation package.

Graphical user interface, application

Description automatically generated

Fig Map in RVIZ

1. **Generating the map:**

The robot will be moved around the environment from a set point to generate a map using the SLAM toolbox. The robot will use its sensors to detect obstacles and the moisture level of the cashews. The generated map will be saved and used by the Nav2 package to plan the robot's path.

Graphical user interface

Description automatically generated

Fig Gazeboworld

1. **Moving the bot from the set point to a goal point:**

Once the map is generated, the robot will be moved from the set point to a goal point. The Nav2 package will be used to plan the robot's path and avoid obstacles. The robot will use its ploughing mechanism to reduce the moisture content of the cashews as it moves along its path.

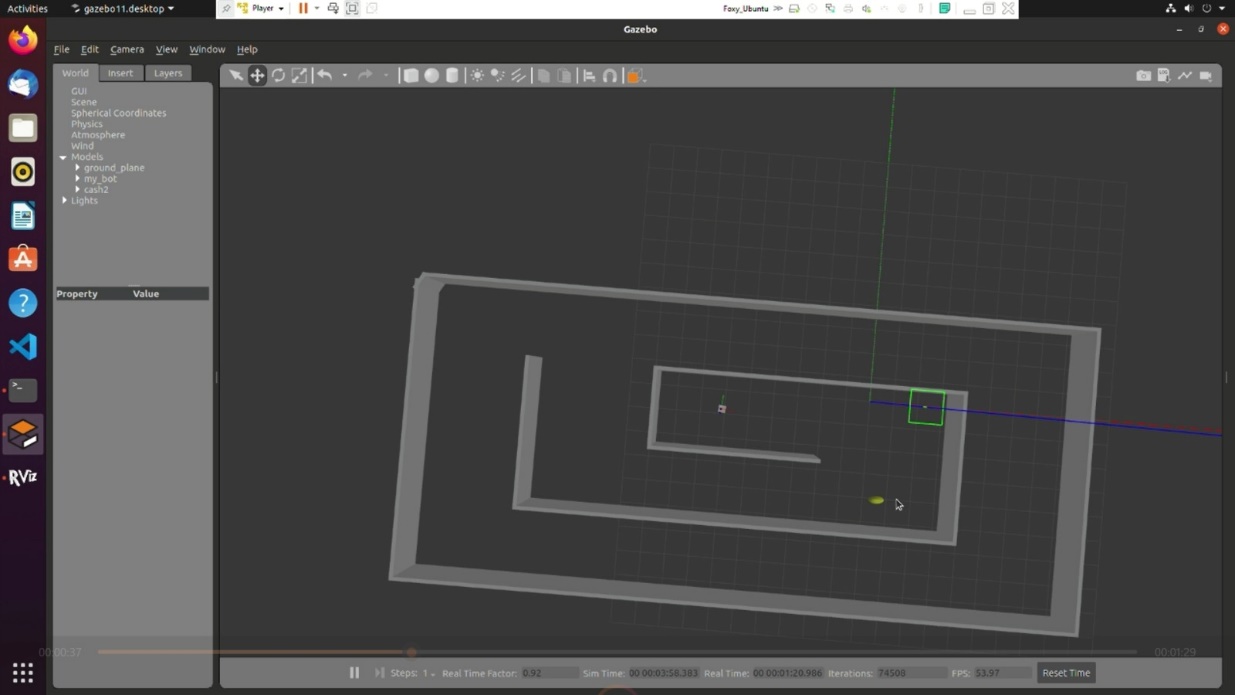


Fig Moving towards the goal point

**6. Algorithms Used**

***6.1 Google ceres***

Ceres Solver is a powerful optimization library that has been used in production at Google for more than four years. It is known for its clean, extensively tested, and well-documented code that is actively developed and supported. The library's modeling API is designed to make it easy for users to build and modify the objective function. The API allows the user to create the objective function one term at a time, without worrying about how the solver will deal with the resulting changes in the sparsity or structure of the underlying problem.

One of the most tedious and error-prone parts of using an optimization library is supplying derivatives. Ceres Solver addresses this problem by shipping with automatic and numeric differentiation, allowing users to compute derivatives without doing it by hand. Additionally, Ceres allows users to mix automatic, numeric, and analytical derivatives in any combination that they want.

Ceres Solver also provides robust loss functions that allow users to reduce the influence of outliers in the data. Most non-linear least squares problems involve data, and where there is data, there will be outliers. Ceres allows the user to shape their residuals using a loss function, which reduces the influence of outliers.

In many cases, some parameters lie on a manifold other than Euclidean space, such as rotation matrices. In such cases, Ceres allows users to specify the geometry of the local tangent space by specifying a Manifold object. This is an important feature for many real-world optimization problems.

Ceres Solver comes with a variety of optimization algorithms that suit different needs. Depending on the size, sparsity structure, time, and memory budgets, and solution quality requirements, different optimization algorithms will be appropriate. Ceres Solver supports Trust Region Solvers such as Levenberg-Marquardt, Powell's Dogleg, and Subspace dogleg methods. It also supports Line Search Solvers like Non-linear Conjugate Gradients, BFGS, and LBFGS. The library has been extensively optimized with C++ templating, hand-written linear algebra routines, and OpenMP or modern C++ threads-based multithreading of the Jacobian evaluation and the linear solvers. Additionally, if the system supports CUDA, then Ceres Solver can use the Nvidia GPU on the system to speed up the solver.[5]

Ceres Solver is the best-performing solver on the NIST problem set used by Mondragon and Borchers for benchmarking non-linear least squares solvers. Its speed, flexibility, and ease of use make it an excellent choice for solving optimization problems in various fields, including computer vision, robotics, and machine learning.

***6.2 Elastic pose-graph deformation***

Changes to the environment cannot be persistently implemented using the pure localization approach. Instead, it compares data from the current session to a rolling buffer of measurements from the initial session(s) and the pose-graph. The pose-graph will be updated using the measurements from the current session along with additional restrictions and nodes. This enables adaptation to environmental changes based on newly added features or relocated objects to improve localization quality. The measurements in the rolling buffer will eventually "expire" and be eliminated from the pose-graph and localization issue, returning the pose-graph for that area to its initial condition. Elastic pose-graph deformation is how the authors describe this action. The fact that the pure-localization mode, when utilised with no previous mapping session data, can be used for efficient lidar odometry is an intriguing side effect. Lidar odometry can scale to spaces of any size since it will only compare data to its local buffer and retain just current views of the environment.[4]

***6.3 Extended Kalman Filter SLAM***

The EKF-SLAM method represents the robot motion and observation models as non-linear functions that are linearized around the current estimates of the robot pose and map. The robot motion is described by a function that models the kinematics of the robot and adds a zero-mean Gaussian noise with covariance to account for uncertainties in the robot motion. The observation model is described by a function h that relates the current robot pose and the observed landmarks' positions and adds a zero-mean Gaussian noise with covariance to account for uncertainties in the sensor measurements.

The EKF-SLAM algorithm performs two steps iteratively to estimate the joint posterior distribution of the robot pose and map. The time update step uses the motion model to predict the new robot pose and the map's covariance, given the previous estimates and the control inputs. The observation update step updates the joint posterior distribution, given the new sensor measurements and the predicted robot pose and map. The update involves computing the Kalman gain, which balances the relative weight of the motion model and the sensor measurements in the estimation process.[3]

The EKF-SLAM method has several issues that need to be considered. Convergence of the map estimate is not guaranteed, and the algorithm may suffer from inconsistency due to non-linearities in the motion and observation models. The computational complexity of the algorithm increases with the number of landmarks, which makes it challenging to scale up to large environments. Data association, i.e., the problem of associating measurements with landmarks, is a significant issue, especially in environments with similar or ambiguous landmarks. Several variants of EKF-SLAM have been proposed to address these issues, and ongoing research continues to improve its performance and scalability.

**7. Conclusion**

The successful simulation of a mobile robot in ROS2 using SLAM and Nav2 toolbox is presented . The physical dimensions of the robot were first designed using SolidWorks software, and the resulting CAD model was exported to a URDF file. Necessary sensor plugins were installed, and launch files were written to control the robot's behaviour. The SLAM toolbox and Nav2 were downloaded and utilized for localization, path planning, and obstacle avoidance.

In addition, a virtual environment was created using Blender, which included trees and bushes as obstacles with uneven ground. A map of the environment was generated. The robot was able to move from a set point to generate a goal point successfully.

The simulation process demonstrated the capabilities of ROS2 in creating a realistic simulation of a mobile robot and the effectiveness of SLAM and Nav2 in providing accurate localization and path planning. With this simulation, it is possible to test the robot's behaviour in different scenarios, which can help improve its performance in the real world. Overall, the success of the procedures highlights the potential of ROS2 in designing and simulating robots.

1. **References**

[1] N. T. Tran, T. D. Ngo, D. K. Nguyen, P. X. Son, and N. H. Thai, *Mapping and Path Planning for the Differential Drive Wheeled Mobile Robot in Unknown Indoor Environments Using the Rapidly Exploring Random Tree Method*, vol. 1. Springer Nature Singapore, 2022. doi: 10.1007/978-981-19-1968-8\_43.

[2] G. Rajendran, U. V, and B. O’Brien, “Unified robot task and motion planning with extended planner using ROS simulator,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7468–7481, 2022, doi: 10.1016/j.jksuci.2021.07.002.

[3] S. Macenski and I. Jambrecic, “SLAM Toolbox: SLAM for the dynamic world,” *J. Open Source Softw.*, vol. 6, no. 61, p. 2783, 2021, doi: 10.21105/joss.02783.

[4] T. Bailey and H. Durrant-whyte, “Simultaneous Localisation and Mapping ( SLAM ): Part II State of the Art,” pp. 1–10.

[5] A. Merzlyakov and S. MacEnski, “A Comparison of Modern General-Purpose Visual SLAM Approaches,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2, pp. 9190–9197, 2021, doi: 10.1109/IROS51168.2021.9636615.