MediAssist: Robotic Medication Companion for Elderly Care

Nikhilesh Gorrepati[1], Sai Shivani Vala[2], and Vyshnavi Ryali[3]

***Abstract*— Robot applications are at an all-time high, and the medical industry is one such area where these robots could be deployed. This project offers a novel solution to the problems related to older patients' medicine administration. This project presents an autonomous robotic system intended to give individualized support and help older people with their drug regimens. It serves as a friend for the elderly, making sure the right doses are given when needed. By bridging the gap in medicine management, the project aims to promote independence, safety, and well-being among the aged population. It seeks to provide family members and carers with peace of mind. The project aims to enhance the quality of life for senior citizens by developing and implementing MediAssist. It also seeks to improve health outcomes and create a sense of empowerment and support among the aging population.**

I. INTRODUCTION

Semi-humanoid robots are becoming more and more common, and soon we'll see them in every aspect of our daily life. These robots will someday take the place of humans in some minimalist jobs or hazardous circumstances since they are designed to move and behave like humans. Due to physical limits or cognitive problems, elderly people sometimes struggle to conduct daily duties alone. There is an immediate need for an Assistive Robot for Elderly Care[1] that can physically aid with chores like fetching medicines, and assisting to enhance their quality of life and guarantee their well-being. In the context of this project, the Fetch robot serves as a central element in a comprehensive task involving autonomous navigation, manipulation, and object recognition. The overarching goal is to leverage the capabilities of the Fetch robot to perform a specific task: picking up a medicine box from a table and autonomously navigating to a designated room to place the box. This project showcases the integration of complex robotic functionalities to solve a practical problem in a simulated environment. The robot is simulated in a house environment using a gazebo to view how the robot interacts in it.

II. RELATED WORK

The use of robotic systems in the healthcare industry has gained popularity in recent years. This section provides an overview of the relevant tools or packages that have been used to implement projects in the area of medical robotics, specifically using the Robot Operating System (ROS), and compares them with our proposed model. In research paper [2], the artificial potential field method has been used to determine the path. However, this method has limitations because it provides a longer path when compared to other navigation and path-planning algorithms. In our model, we have used the ROS Navigation Stack for autonomous navigation which provides a shorter path to reach the required goal. In research paper [3], a vacuum gripper has been used for object manipulation. However, this method has limitations because when handling objects that require grasping in different ways (e.g., pinch grip, parallel grip, etc.), vacuum grippers may not be as adaptable. We used two-fingered grippers in our model because it has several benefits, including versatility in handling objects in ROS Gazebo simulations with effectiveness.

III. METHOD

This project is application-oriented. We will develop a robotic system where the robot will be outfitted with a manipulator arm capable of picking up medication containers and delivering them to the user. There are two modes for the robot. Pick and Place, and Navigation. The robotic arm is utilized in the Pick and Place mode to pick items like medication. In this mode, the user has to tell the robot where the object is located. After that, the robot arm determines the optimum path to take in order to pick the object as instructed. The robot's second mode is navigation.

A. Software Tools/Packages

- Fusion 360 is a 3D designing software. Using this software, all of the robot's 3D designs are created. After that, these designs are exported as STL files so that the linkages' collision and visual features may be used in URDF.
- A Robotic Operating System (ROS) is a meta-operating system. It is an assortment of software tools and libraries designed specifically for robots. ROS is simple to use since its nodes support Python and C++, the two most popular programming languages. This project's experiments were conducted in ROS Noetic. The operating system for this project is Ubuntu 20.04, a Linux distribution built on the Debian operating system.
- MoveIt is an open-source platform used to manipulate the arms of a robot. It is primarily designed to function in ROS. MoveIt is compatible with ROS as a library. It uses the user-provided final end-effector configuration as a starting point and returns joint states for every robot joint to get to that configuration without colliding.
- Rviz is a visualization tool available as a library in ROS. It can be used to view various topics in ROS. Rviz is a fantastic tool for tracking the present status of the robot because it allows for the visualization of the entire robot. It can facilitate quick URDF model debugging. MoveIt is compatible with Rviz as well. The robot arm

can have a goal state assigned to it, and we can see the path it takes to get there.

- Gazebo[4][5] is a widely used open-source robot simulation software that could be used alongside ROS. Also, it supports various robot models, sensors, and controllers, and is best for simulating the physical aspects of our robotic system, including manipulation and navigation making it suitable for this project. Many real-world environments can be simulated in Gazebo. The robot can be added to the simulated environment to check the interaction between the robot and the replicated world.
- Hector SLAM (Simultaneous Localization and Mapping) is a popular open-source SLAM package in the field of robotics and autonomous navigation. It's used for mapping environments and localizing a robot within that mapped environment. This package is particularly known for its robustness and efficiency in mapping large-scale environments using LIDAR data.
- ROS Navigation Stack is a collection of software packages that assist your robot in moving from a beginning place to a goal location.
- OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. More than 2,500 optimised algorithms are available in the library, including a wide range of both traditional and cutting-edge computer vision and machine learning techniques. In addition to detecting and recognising faces, these algorithms can identify items, track moving objects, and do a lot more.

IV. EXPERIMENTS

Below are the results obtained:

A. Robot Design

The Fetch robot's key capabilities that contribute to the success of this project include:

- Mobile Base: The robot is equipped with a robust mobile base that enables it to navigate through indoor environments efficiently. The base is capable of autonomous motion and can follow predefined trajectories or respond dynamically to the surrounding environment.
- Manipulation Arm: A sophisticated robotic arm, equipped with an end-effector (gripper), allows the robot to grasp, lift, and manipulate objects. This arm is crucial for the successful execution of tasks involving object interaction and placement.
- Sensors and Perception: The Fetch robot is equipped with a set of sensors, including cameras and depth sensors, enabling it to perceive and understand its surroundings. This perception capability is essential for recognizing objects, identifying obstacles, and making informed decisions during navigation.

B. Simulation World

According to our use case where the robot will be assisting elderly people with fetching medicines regularly, we have created a world that simulates a small house with two rooms - a living room with a table and a bedroom.

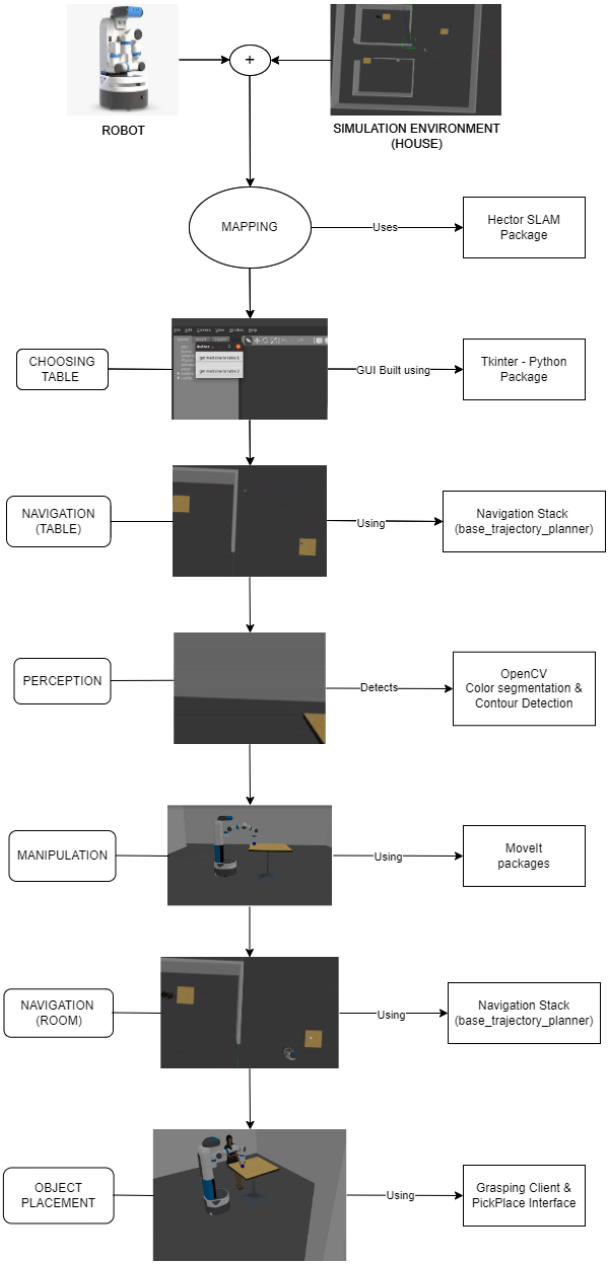

Fig. 1. Methodology

C. Adding a LIDAR

After setting up the simulation environment, we need to add a LIDAR to our robot so that it can perform SLAM (Simultaneous Localization And Mapping) within the house world. For this, we added a laser link and a Gazebo plugin (gazebo_ros_head_hokuyo_controller) to the existing robot urdf file.

D. Creating and Saving a Map

To facilitate navigation, path planning, and localization, we must now create a map. We mapped the robot's environment using Hector SLAM, a ROS package used in Gazebo

Fig. 2. Robot Design

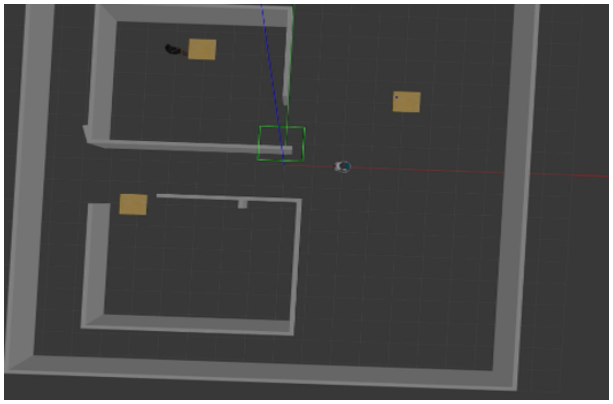

Fig. 3. Simulation Environment

simulations. Below are the steps we followed to generate a map,

- Initially, we set up Qt4, a program for creating graphical user interfaces.
- After that, we cloned the Hector-SLAM package into our workspace on ROS.
- For the Hector-SLAM launch file to properly integrate with the base and odometry frames of our robot, we set up the frame parameters.
- We first built the packages, and then we started the Gazebo simulation and the Hector-SLAM mapping process.
- Lastly, we used the rqt_robot_steering tool to control the robot's movement.
- This was done in a precise order to ensure accurate mapping results by moving the robot slowly during the mapping process.

Once the map is created, it must be saved for later use. We utilized the map-server package, which facilitates the saving of maps as yaml and pgm formatted files. Maps that have been saved can be loaded and viewed at any time.

E. Robot Navigation

After setting up the world, equipping our robot with a LIDAR, and creating a map, we can now focus on the robot's autonomous navigation within the virtual environment. We did that by utilizing the ROS Navigation Stack, a collection of software tools and libraries in ROS. The following are the procedures we used to set up the ROS Navigation Stack parameters:

- Installed the ROS Navigation Stack.
- Configured RViz settings for visualization.
- Defined parameters for the base local planner to compute velocity commands.
- Set up global and local costmaps using YAML files for obstacle representation.
- Adjusted AMCL parameters for precise robot localization and mapping.
- These actions collectively enable the robot to navigate autonomously in a Gazebo simulation.

In the project, the base trajectory planner is responsible for planning and executing trajectories for the mobile base of the Fetch robot during navigation. This planner employs a global path planning strategy to determine the optimal route from the current robot position to the desired goal location. The trajectory is then executed to facilitate smooth and collision-free movement of the robot. Our robot will now be able to plan paths, avoid obstacles, and move independently in the simulated environment by utilizing the ROS Navigation Stack for autonomous navigation.

F. Sending Simulated Robot to Goal Locations

Now that our robot can navigate autonomously, we need to work on sending it to a specified goal location within the simulated environment for picking up medicines and giving them to the elderly. To achieve this target, below is the solution we will be following:

- Number the main goal locations we would like to send our robot. In our case, there are three rooms - the living room (to pick up medicines from the table) and bedrooms (to deliver to the elderly) Let us assume, 1-Room-1 2- Room-2.
- Make a note of the X and Y coordinates of each desired goal location using the Rviz Point Publish button.
- Write a program that takes goal location as input and reaches there based on X and Y coordinates we get from Rviz.
- This helps our robot to navigate between the rooms for picking up medicines and delivering them to the elderly.

G. Object Detection using OpenCV

We used OpenCV for object detection. The general steps to be used for object detection with OpenCV are mentioned below:

- Capture Simulation Images: With the help of ROS packages such as gazebo_ros_camera or

gazebo_ros_snapshot, you can build a ROS node that records pictures from the Gazebo simulation.

- Implement OpenCV Object Detection: Create a custom ROS node that recognizes medications using OpenCV for object detection. We assume that the medicine box is rectangular. Whenever an object with specified properties comes into our robot's sensor, the robot will detect it using OpenCV and pick the box.
- Publish Detection Results: Publish the detection results as ROS messages, such as sensor_msgs/Image to represent detected objects, as soon as medicines are detected in the images.
- Visualization and Integration: We can visualize object detection in RViz or Gazebo, by creating custom markers that show bounding boxes or labels around detected medications. To ensure proper communication with the simulation environment, integrate the object detection node with the launch files and ROS workspace.

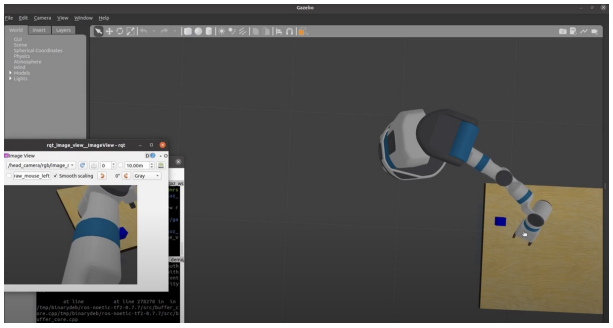

Fig. 4. Detecting Medicine Box

H. Object Manipulation with MoveIt and Two-Fingered Gripper

Using MoveIt with a two-fingered gripper allows robots to plan and execute precise grasping and manipulation tasks, enhancing their object-handling capabilities. The steps listed below can be used to accomplish fetch manipulation.

- Fit a two-fingered gripper to the robot arm so it can pick up and handle objects.
- Include the gripper in the MoveIt setup as an end effector, specifying its grasp points and directions of approach.
- Define the target poses and collision characteristics of our object in the MoveIt environment.
- To generate collision-free trajectories for the robot arm and vacuum gripper to grasp, lift, and move objects, create motion planning and control scripts.
- To carry out your manipulation tasks, connect MoveIt to our Gazebo simulation and use it to simulate the Fetch robot's actions in a virtual world.

I. Object Placement at desired location

Once the medicine box is grasped, the robot navigates to the room that was initially selected and places it on the table. Since the table coordinates were pre-calculated and provided to the robot during the room selection process, the robot moves to the selected room and places the medicine box on the table.

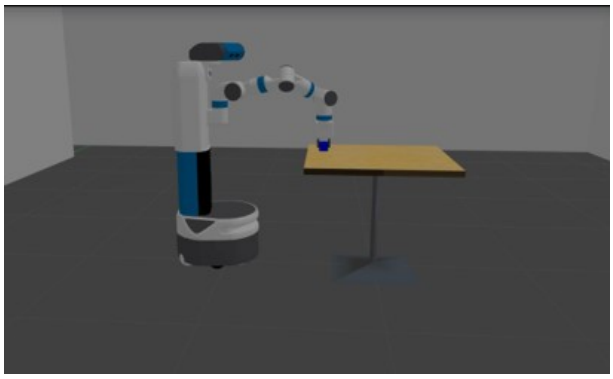

Fig. 5. Object Manipulation

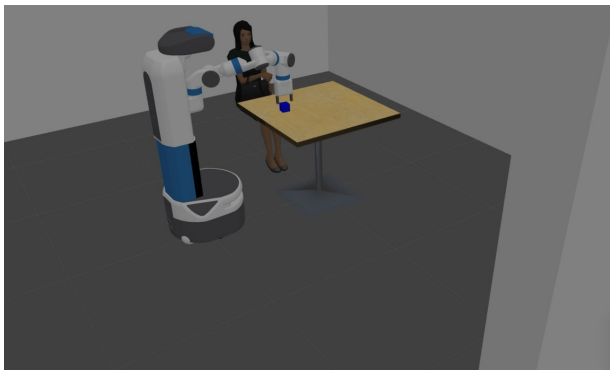

Fig. 6. Object Placement

V. CONCLUSION

REFERENCES

[1] M. Andtfolk, L. Nyholm, H. Eide, and L. Fagerström, "Humanoid robots in the care of older persons: A scoping review," 2021, pp. 518–526.

[2] Y. Y. C. L. Yang Li, Bin Tian, "Path planning of robot based on artificial potential field method." IEEE, 2022.

[3] M. A. H. S. S. S. Y. S. Yukihisa Karako, Toshihiro Moriya, "A practical simulation method for pick-and-place with vacuum gripper." IEEE, 2017.

[4] I. Peake, J. L. Delfa, R. Bejarano, and J. O. Blech, "Simulation components in gazebo." IEEE, 2021.

[5] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator." IEEE, 2004.