Autonomous Warehouse Pick-and-Deliver Robot

Shubham Derhgawen
University of California
Riverside,USA
sderh001@ucr.edu

Harshini Murugadoss
University of California
Riverside,USA
hmuru002@ucr.edu

***Abstract*— This paper introduces an autonomous warehouse pick-and-deliver robot designed to address the limitations of current warehouse automation systems, which often lack flexibility and affordability. Built on the TurtleBot 3 Waffle Pi platform with an OpenManipulator arm, the system integrates LiDAR-based SLAM, RRT* path planning, and inverse kinematics (IK) for object manipulation. Implemented in the Robot Operating System (ROS) and simulated in Gazebo, the robot demonstrates effective navigation and preliminary object-handling capabilities in a warehouse environment. Experimental results highlight successful mapping, path planning, and waypoint navigation, laying the foundation for a low-cost, adaptable warehouse solution.**

Keywords*— *Autonomous Robots, Warehouse Automation, ROS, SLAM, Path Planning, Inverse Kinematics, Robotic Manipulation

I. Introduction

Warehouse automation has transformed logistics by improving efficiency, yet many existing robotic systems are constrained by high costs, rigid infrastructure requirements, and limited adaptability to varied tasks. Current solutions excel at fixed operations but struggle with generalizability, leaving a gap for affordable, flexible alternatives suited to dynamic warehouse environments.

This project presents an autonomous pick-and-deliver robot aimed at overcoming these challenges. By integrating a mobile base with a robotic manipulator, the system can navigate a warehouse, map its surroundings, and handle objects autonomously. The robot leverages the TurtleBot 3 Waffle Pi and OpenManipulator arm, with simulation in Gazebo and control via ROS.

II. Robotic System

A. Hardware Overview

- TurtleBot 3 Waffle Pi: A differential-drive platform equipped with LiDAR, an IMU, and wheel encoders for robust mobility and localization.
- OpenManipulator Arm: A lightweight manipulator for pick-and-place operations, suitable for small warehouse items.
- Sensors:
  - 2D 360° LiDAR for mapping and obstacle detection.
  - Camera for future dynamic object detection.
  - IMU for orientation tracking, fused with wheel encoder data for enhanced localization.

B. Software Stack

- *ROS GMapping:* SLAM algorithm for generating a 2D occupancy grid map.
- *RRT Algorithm*:* Path planning to compute collision-free trajectories.
- *Gazebo:* Simulation environment replicating a warehouse with obstacles.
- *Inverse Kinematics (IK):* Calculates joint angles for the OpenManipulator arm to handle objects.
- *PID Controller:* Closed-loop control for precise waypoint navigation.

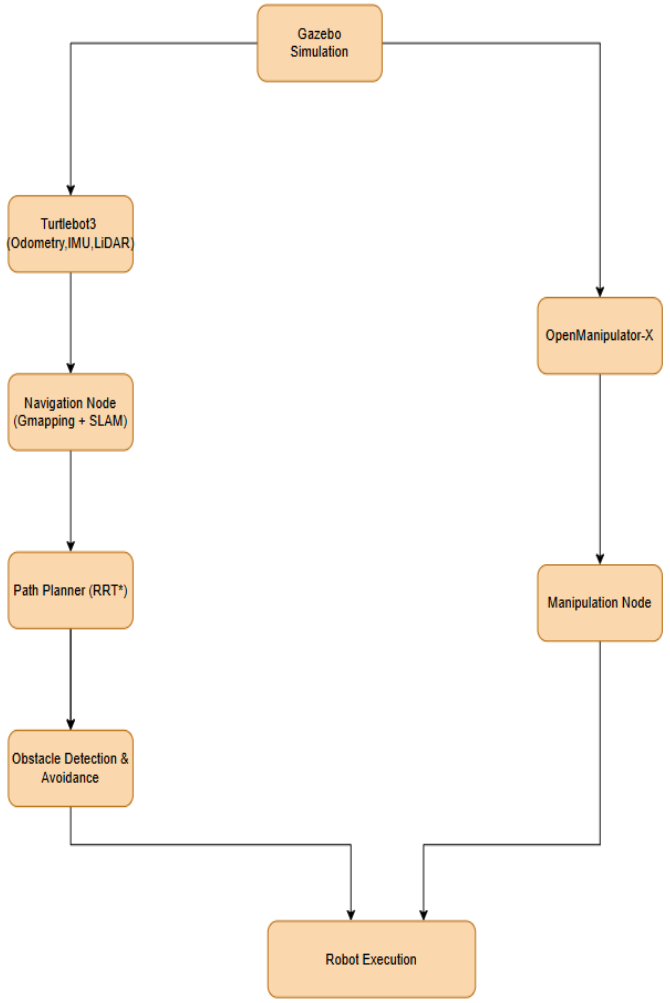

Fig. 1. Node Level Architecture

III. Methodology

We begin by selecting an appropriate mobile robot equipped with a manipulator arm suitable for warehouse operations. Next, we develop a simulated warehouse environment to replicate realistic operational scenarios. Initially, the robot is teleoperated to navigate the warehouse and perform Simultaneous Localization and Mapping (SLAM), generating a precise occupancy grid map (.pgm file) using the GMapping algorithm. With the generated map, we employ the RRT* path planning algorithm to compute collision-free paths and generate navigation waypoints. These waypoints are subsequently fed into a closed-loop PID controller that guides the robot accurately along the planned trajectory.

To mimic realistic warehouse conditions, dynamic cylindrical models representing humans are introduced into the simulation environment. The robot actively detects these moving cylinders, employing a Time-to-Collision (TTC) method for robust collision avoidance. Upon detecting an obstacle, the robot dynamically adjusts and re-plans its trajectory to maintain safe navigation. Finally, upon successfully reaching its destination, the robot performs precise object detection and manipulation tasks. Utilizing vision-based systems and inverse kinematics, it efficiently picks up the target object and proceeds along the pre-calculated route to complete the placement task.

A. Mapping and Localization

The GMapping SLAM algorithm fuses LiDAR scans with IMU and wheel encoder data to construct a 2D occupancy grid of the warehouse. This map is exported as a .pgm file for use in path planning..

B. Path Planning and Navigation

The RRT* (Rapidly-exploring Random Tree Star) algorithm generates optimal, collision-free paths by creating waypoints that adapt to the known warehouse layout. These waypoints form a navigable route tailored to the robot's specifications and operational constraints. A closed-loop PID controller manages the TurtleBot's movement along this predefined path, continuously adjusting velocity and orientation based on real-time sensor feedback to ensure accurate navigation and waypoint adherence.

C. Collision Avoidance

Real-time obstacle detection employs LiDAR sensor data integrated with a time-to-collision estimation method to dynamically identify potential collisions during navigation. Upon detection of obstacles or moving objects within its trajectory, the robot promptly recalculates its route to maintain safety. This adaptive system continually monitors and updates the robot's path, enabling effective collision avoidance and uninterrupted navigation through dynamically changing environments.

D. Object detection and Manipulation

Upon reaching the designated pickup location, the robot engages its camera system combined with image processing techniques to identify and classify the target object precisely. Inverse kinematics algorithms calculate the optimal arm trajectories necessary to approach and grasp the object efficiently and safely. Following successful object manipulation, the robot securely transports the object along the path defined by the RRT* planner toward the designated placement location, demonstrating integrated navigation and manipulation capabilities.

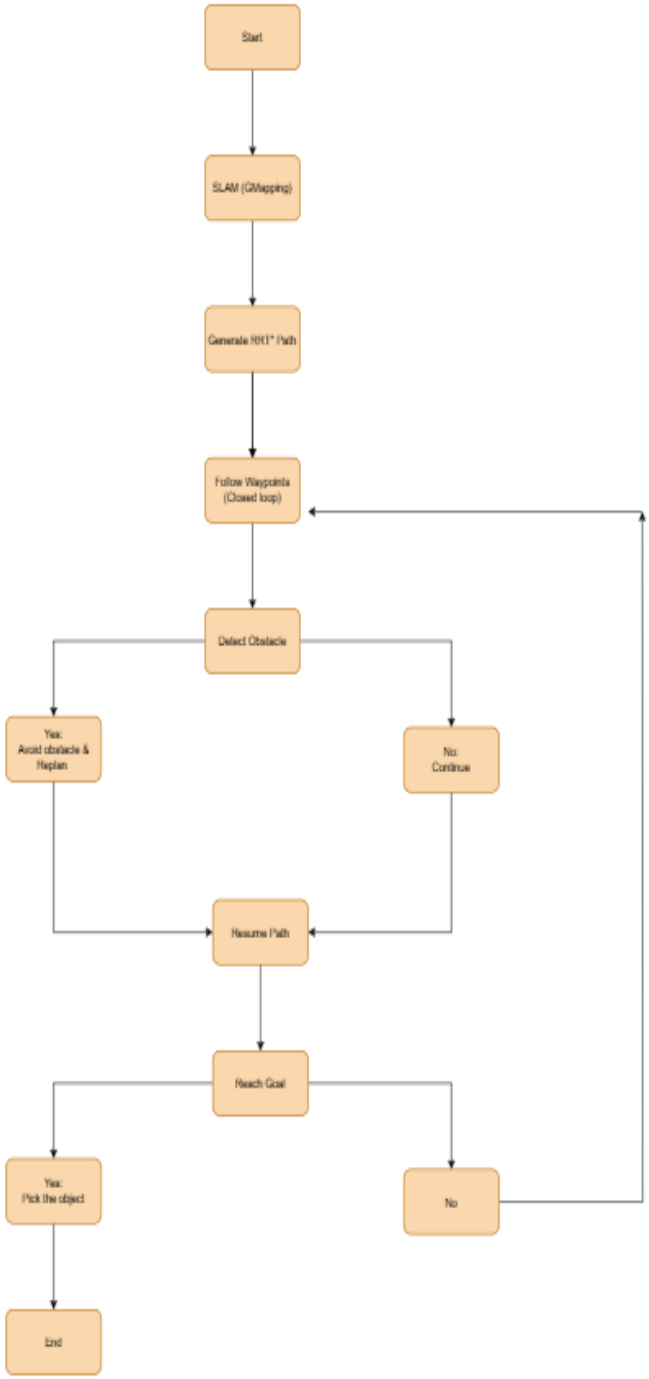

Fig. 2. High Level Flowchart

IV. Code Status and Resource Used

A. Navigation Controller

The navigation node loads the occupancy grid from GMapping and uses RRT* to generate waypoints. The PID controller ensures precise traversal:

- If no obstacles are detected, the robot proceeds.
- If an obstacle is within range, it pauses and triggers the collision avoidance routine

B. Manipulation Module

The manipulation routine employs IK to position the OpenManipulator arm. A geometric approach calculates joint angles based on the object's relative position.

C. Integration with Gazebo

The TurtleBot 3 and OpenManipulator arm are spawned in a custom Gazebo warehouse environment, enabling testing of mapping, navigation, and preliminary manipulation.

V. Results

A. Map Generation

A simulated warehouse environment was mapped using GMapping, producing an accurate 2D occupancy grid. This map served as the basis for RRT* path planning.

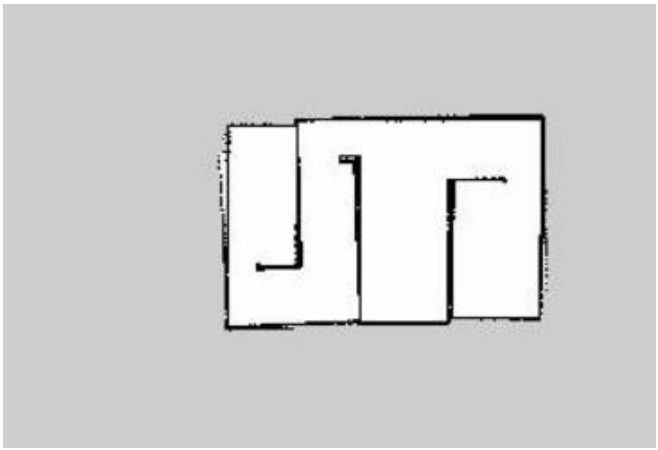

Fig. 3. Grid Map generated using GMapping

B. Trajectory Visualization

The RRT* algorithm and PID controller enabled the robot to follow waypoints effectively, covering the warehouse area systematically.

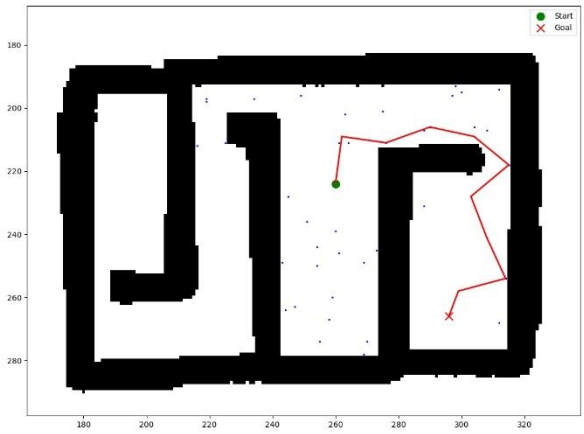

Fig. 4. Path generated by RRT* Algorithm

C. Collision Avoidance

The warehouse environment simulates human workers as cylindrical models moving along designated passages. The autonomous robot proactively detects and avoids collisions with these moving cylinders, demonstrating effective obstacle avoidance capabilities during navigation.

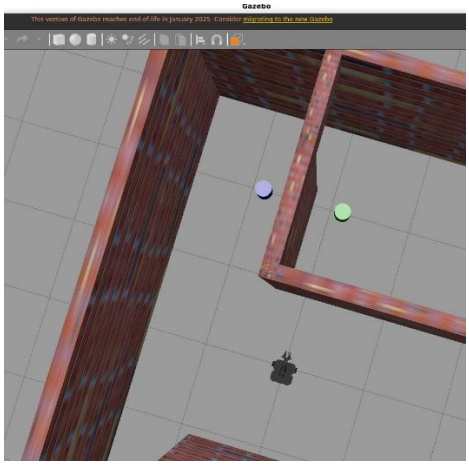

Fig. 5. Cylinders vizualized along the path

VI. Conclusion and Future Work

This project demonstrates a promising autonomous warehouse pick-and-deliver robot, validated in Gazebo with successful mapping, path planning, and navigation. The integration of the OpenManipulator arm enhances its potential for object handling, distinguishing it from traditional warehouse robots.

Future work includes:

- Adding camera-based dynamic object detection and real-time SLAM for adaptability.
- Deploying the system on physical hardware to validate real-world performance.

ACKNOWLEDGMENT

The authors thank the EE283A course instructor for their guidance. We also acknowledge the open-source ROS community for packages like GMapping, and TurtleBot3.

REFERENCES

[1] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open-source software, 2009.

[2] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. MIT Press, 2005.