Smart Floor-Cleaning Robot with Robotic Arm

Anirudh Gudi
Jerin Peter
Department of Electrical and Computer Engineering
University of California, Riverside
Email: {agudi018, jpete085}@ucr.edu
Github: FCleanBOT

***Abstract*—This project presents a smart floor-cleaning robot that integrates autonomous navigation and robotic arm manipulation to clean floors while detecting and relocating objects. The system is implemented in the Robot Operating System (ROS) and simulated in Gazebo, using a TurtleBot3 Waffle Pi equipped with an OpenManipulator-X arm. Key features include LiDAR-based SLAM, a boustrophedon coverage algorithm for path planning, and inverse kinematics (IK) for object handling. Experiments in a simulated home environment demonstrate efficient cleaning coverage and successful object manipulation.**

***Index Terms*—Autonomous Robots, Robotic Manipulation, ROS, SLAM, Inverse Kinematics, Smart Cleaning.**

I. Introduction

Floor cleaning and object tidying are common household tasks, often requiring significant time and effort. Although several commercial robotic vacuum cleaners handle basic floor cleaning, they typically cannot manipulate objects obstructing their cleaning path. This project addresses that limitation by integrating a robotic arm onto a mobile base, enabling both cleaning coverage and object handling. Our system builds upon the TurtleBot3 Waffle Pi, adding an OpenManipulator-X arm, with simulation in Gazebo and control via ROS.

A. Motivation

The primary motivation is to design a comprehensive household robot that can autonomously cover a floor area for cleaning while also interacting with objects it encounters. By leveraging LiDAR for SLAM and depth sensing for object recognition, the robot can detect an obstacle, stop, and pick it up, thus avoiding partial cleaning.

II. System Architecture

A. Hardware Overview

- **TurtleBot3 Waffle Pi**: Provides a differential-drive base, LiDAR, onboard IMU, and wheel encoders for odometry.
- **OpenManipulator-X Arm**: A 4-DoF manipulator that can pick, place, or reposition small objects.
- **Sensors**: 2D 360° LiDAR, depth camera, IMU, and wheel encoders fused for accurate localization and obstacle detection.

This project was completed as part of EE283A: Foundations of Robotics.

B. Software Stack

- **ROS GMapping**: SLAM algorithm used to build a 2D occupancy grid map.
- **Boustrophedon Coverage**: Algorithm to generate systematic cleaning paths.
- **Gazebo**: Simulation environment for testing navigation, mapping, and arm manipulation.
- **Inverse Kinematics (IK)**: Calculates the appropriate joint angles for the OpenManipulator-X based on target position.

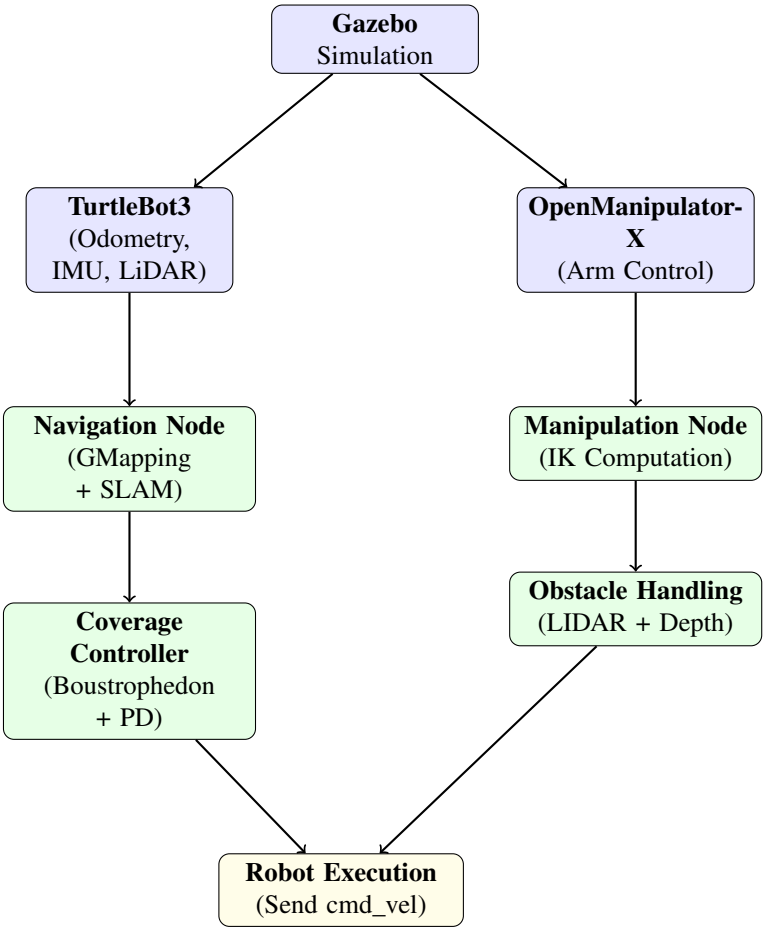


Fig. 1: ROS Node-Level Architecture. Nodes handle SLAM, coverage planning, obstacle detection, and manipulation.

III. Methodology

A. Mapping and Localization

GMapping creates a 2D occupancy grid by fusing LiDAR scans with wheel encoder and IMU data. This occupancy grid is saved and later used for coverage planning.

B. Coverage Path Planning

To ensure full floor coverage, we implement a **boustrophedon** algorithm that partitions the mapped environment into systematic rows. This approach is especially useful for planar coverage tasks.

Algorithm 1 Boustrophedon Path Generation

1: **Input**: Occupancy grid map M
2: **Output**: Set of waypoints W
3: Partition M into free-space regions
4: **for** each region **do**
5: Identify leftmost starting point (x_0, y_0)
6: Generate alternating horizontal paths
7: Append waypoints to W
8: **end for**
9: **return** W

C. Obstacle Detection and Manipulation

LaserScan data is continuously monitored to detect obstacles within a certain distance threshold in front of the robot. If an object is detected while the robot is stationary, the manipulation routine is invoked:

1) Compute IK for the manipulator based on the object's relative position.
2) Execute pick operation (e.g., close gripper).
3) Move to a place position, or simply hold the object out of the cleaning path.
4) Resume coverage path.

D. Custom Object Model

Initially, the manipulator encountered difficulties when attempting to grasp objects within the Gazebo simulation environment. To address this issue, we designed a customized object to facilitate reliable pick-and-place operations. The 3D object was modeled using SolidWorks and subsequently converted into the URDF format utilizing the `solidworks2urdf` tool, along with its associated STL file for visualization in Gazebo. During initial simulations, several stability issues were observed, prompting iterative adjustments to the object's mass and inertia parameters, as summarized in Table I, to achieve stable interaction within the Gazebo environment.

IV. Implementation Details

A. Coverage Controller (`coverage.py`)

The coverage node loads or receives the occupancy grid from GMapping and plans a zigzag pattern (boustrophedon). A

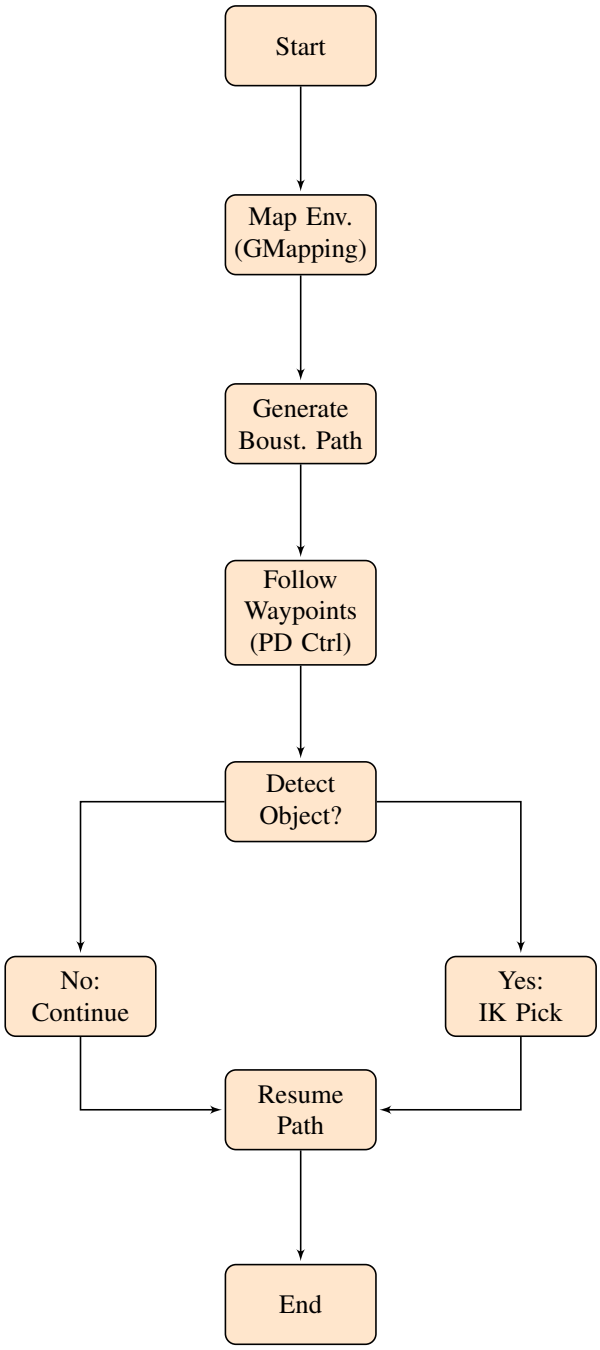


Fig. 2: High-Level Flowchart. The robot maps the environment, generates a boustrophedon path, and either continues cleaning or picks the obstacle with the robotic arm.

TABLE I: Mass and Inertial Properties of the Custom Object

Parameter	Value
Mass (m)	0.05 kg
Inertia I_{xx}	0.001 kg·m^2
Inertia I_{yy}	0.001 kg·m^2
Inertia I_{zz}	0.001 kg·m^2
Inertia I_{xy}, I_{xz}, I_{yz}	0.0 kg·m^2

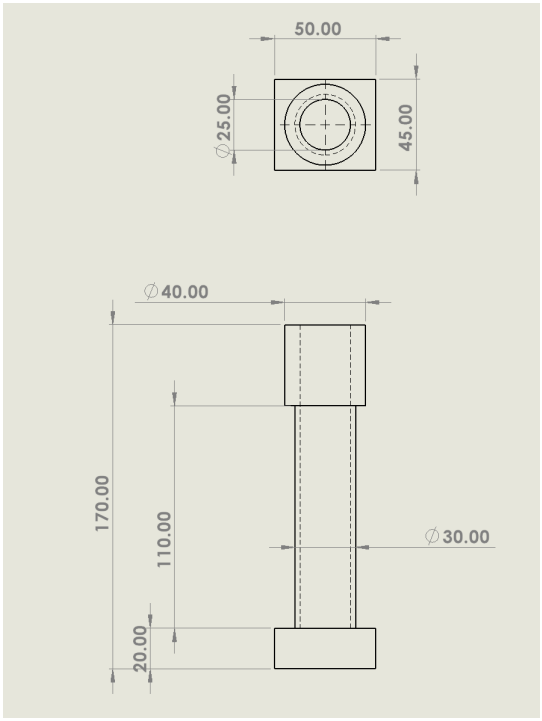


Fig. 3: Custom Object with dimensions(Image_1)

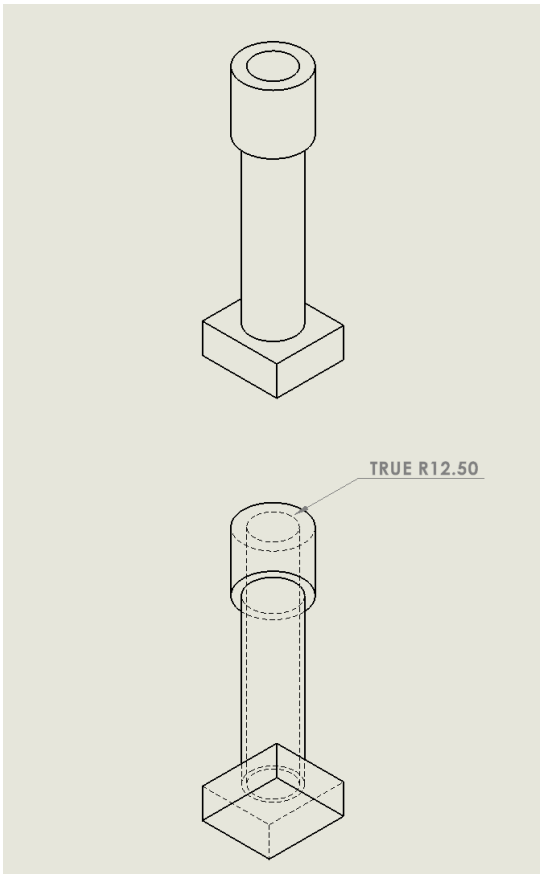


Fig. 4: Custom Object with dimensions(Image_2)

PD controller is employed for reaching each waypoint. During motion:

$$v = K_p e + K_d \frac{de}{dt} \tag{1}$$

where: - v is the control velocity, - K_p and K_d are the proportional and derivative gains, - e is the tracking error.

This controller ensures smooth **path following** by adjusting velocity dynamically.

- If no obstacle is detected, the robot continues forward.
- If an obstacle is detected within 0.3 m, it halts and waits for a signal from the obstacle-handling node.

B. Obstacle Handling and Manipulation (`obs_manipu.py` / `manipu.py`)

- Checks robot velocity and LiDAR data to confirm an obstacle is within picking range.
- If conditions are met, triggers the manipulation routine using inverse kinematics.

- **IK Computation**: A geometric approach to compute joint angles for the OpenManipulator-X, ensuring the end-effector can reach the object's position.

C. Integration with Gazebo

We spawn the TurtleBot3 Waffle Pi and OpenManipulator-X in a custom home environment. The simulation allows testing of:

- Robot odometry and mapping.
- Coverage path planning correctness.
- Obstacle detection and manipulator actions.

V. Experimental Results

A. Map Generation

A 4x4 meter simulated home space was created. GMapping successfully produced a 2D occupancy grid. The boustrophedon path covered the entire 4x4 region with a specified step size (e.g., 0.5 m), ensuring complete coverage. Figure 7 presents the 2D occupancy grid map created via GMapping. This map serves as the foundation for the subsequent boustrophedon path planning.

B. Trajectory Visualization

To illustrate the coverage algorithm in action, Figure 5 displays the boustrophedon path traced by the robot during cleaning:

C. Object Handling Demonstration

During the object manipulation process, the arm transitions through three key poses:

A small box was placed in the robot's path. Upon detection, the system halted, executed IK to pick the box (transitioning from home to intermediate to a grasped position, as shown in Fig. 6), and then resumed the cleaning path. This confirmed correct integration between obstacle handling and coverage.

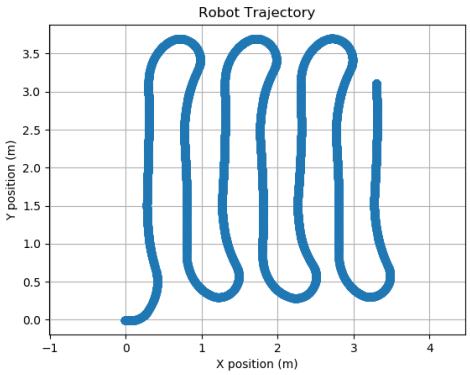
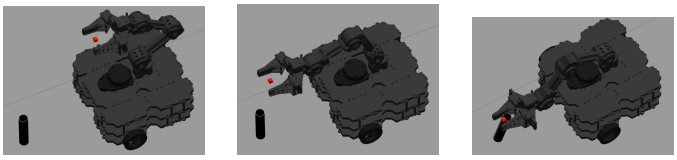


Fig. 5: Robot trajectory in a 4 m × 4 m area. The boustrophedon coverage algorithm systematically covers the region, as shown by the path trace.



(a) Home Position (b) Intermediate (c) Grasped

Fig. 6: Sequence of arm actions upon obstacle detection: a home position, b moving to object, and c grasped position.

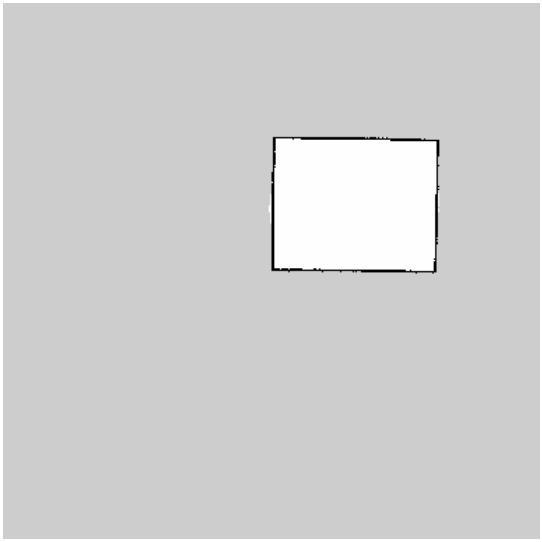


Fig. 7: Occupancy grid map generated by the GMapping algorithm, representing the simulated 4 m × 4 m environment.

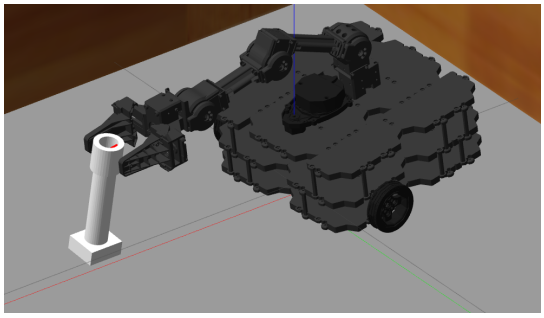


Fig. 8: Robot picking up the custom object.

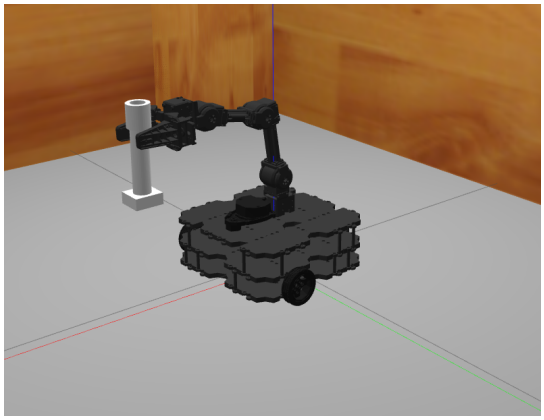


Fig. 9: Robot grasped position with the custom object.

VI. Conclusion and Future Work

We presented a comprehensive floor-cleaning robot that integrates an arm for obstacle (object) handling. The system was validated in Gazebo, demonstrating successful coverage and manipulation. Future enhancements include:

- Real-time object recognition for better pick-and-place operations.
- Improved collision avoidance in cluttered environments.
- Physical deployment to test hardware integration and real-world performance.

The complete implementation is open-source and can be accessed at: https://github.com/anirudhgudi/FCleanBOT

Acknowledgment

We would like to thank the EE283A course instructors for their guidance.

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.
[3] G. H. Nilsson, "A mobile exploration robot for small environments," *Robotics and Autonomous Systems*, vol. 62, no. 5, pp. 753–758, 2014.
[4] R. Siegwart et al., "Introduction to Autonomous Mobile Robots," 2nd ed. MIT Press, 2011.