RoboServe: An Autonomous Food-Serving Robot

Group 10

Sree Vidya Alivelu, Surya Teja Sri Jonnada, Likhitha Emmadi

Overview

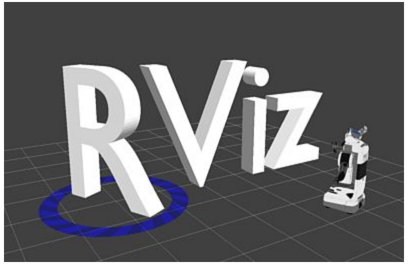
- Introduction
- Simulated Environment
- Method
- Experiments Done
- Evaluation Metrics
- Demo
- Conclusion
- Future Work
- References

Introduction

- The food service sector is always changing, so finding new and creative ways to improve customer service and simplify operations have become critical.
- RoboServe is a cutting-edge method that has the potential to transform the way food is served in restaurants. It offers accuracy and efficiency while also pushing the limits of human-robot interaction.

Simulated Environment

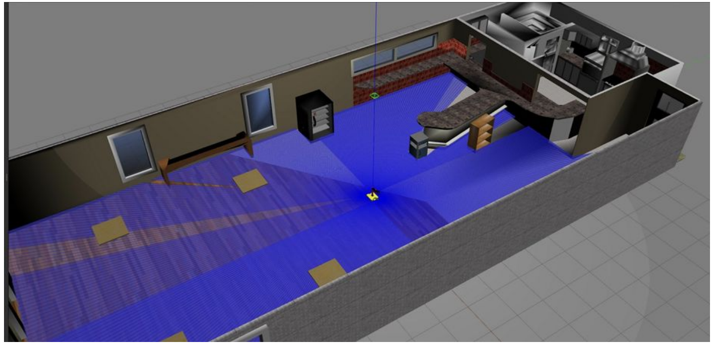
- The simulation environments used for the project are ROS, Rviz, and Gazebo.
- Gazebo and ROS offer a simulated testing environment that is cost-effective, realistic, and scalable.

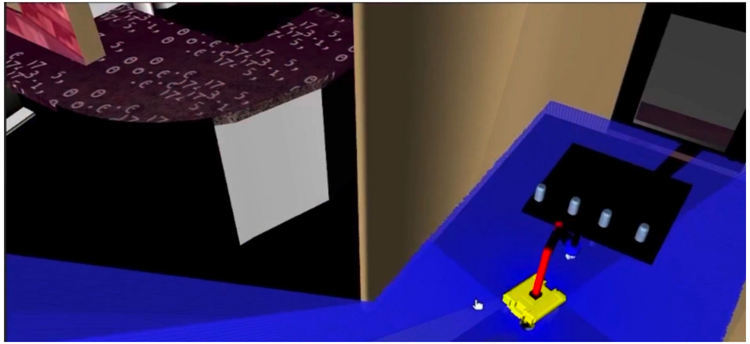


Method

Environment

- The Gazebo simulation environment is designed to replicate a restaurant setting.
- The world consists of a building with a kitchen and a dining area.
- We have a pick up table with cans in the kitchen for the robot to pick up and 4 tables in the dining area.
- Static obstacles have been added to the world.

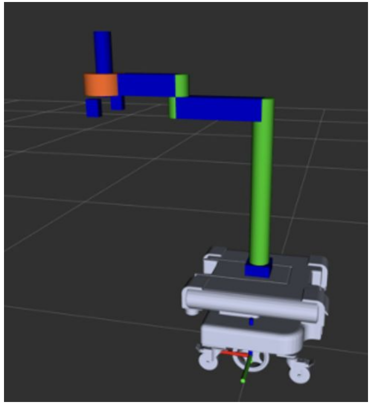




Method

Robot

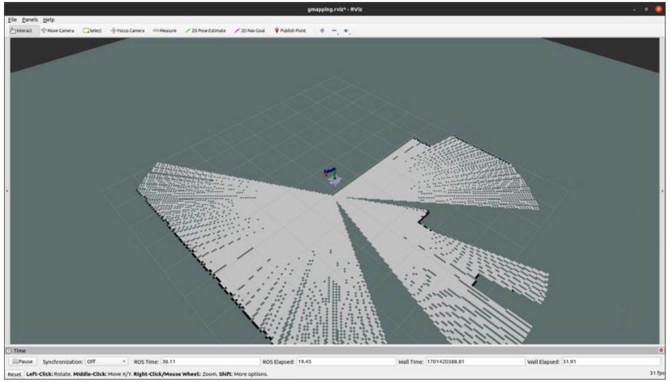
- The Differential drive robot base has two lidar sensors attached to the front and back of the base for complete environment visibility.
- Attached a SCARA arm to the robot base and adjusted the position of the arm.
- Integrated a two finger gripper to the arm to help with the pick and place functionality.

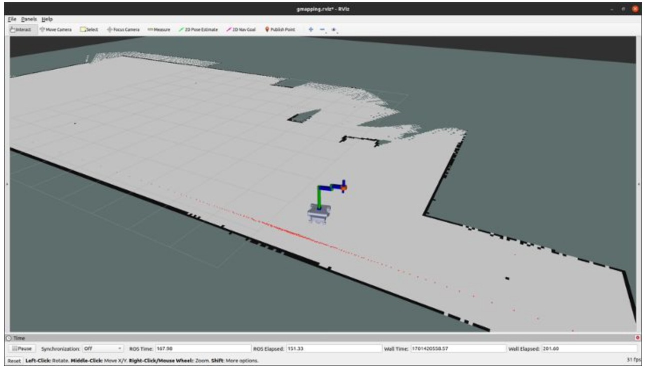


Method (cntd)

Mapping

- We used SLAM (Simultaneous Localization and Mapping) with GMapping to create and retrieve map information.
- Gmapping is a package that provides laser-based SLAM as a ROS node.
- Used teleoperation commands to navigate the robot and created a map of the environment.

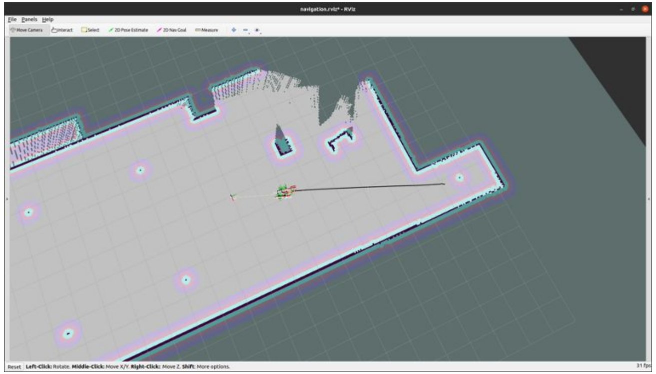




Method (cntd)

Navigation

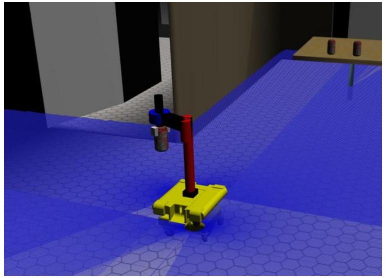
- Used ROS navigation stack
- AMCL and Costmap2d are utilized for localization, and path planning.
- Fine-tuned various parameters, including the robot's footprint, inflation radius, and map files to optimize the performance.
- Integrated mapping sensor information in costmap_common_params yaml file.

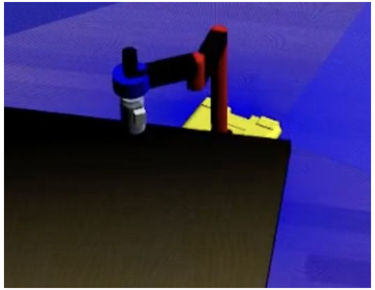


Method (contd)

Manipulation

- The location of the target object and the drop-off tables are known.
- Fine tuned the pid values for the robot joints for a stable and accurate performance.
- We use inverse kinematics to calculate the joint angles given the desired position.
- Implemented tasks to move to the pre-grasp position, open the gripper, pick up the object, close the gripper.





Experiments Done

- The user inputs the desired table number (1 to 4).
- The robot executes a sequence wherein it navigates to the designated pickup table in the kitchen, employs the robotic arm to grasp a can of coke, proceeds to the specified input table (e.g., Table 1), and deposits the can.
- Ran this whole simulation 5 times by selecting different table numbers to check the accuracy of the robot.

Evaluation metrics

- Number of successful object grasps - 4/10
- Effectiveness in placing objects - 6/10
- Collision Incidents during navigation - 2/10
- Correct table arrival rate - 10/10

Demo

Demo video

Conclusion

- In conclusion, we integrated pick-and-place functionality into a mobile robotic system.
- The script, designed to interact with user input for table selection, orchestrates a seamless sequence of actions.
- The robot navigates to the designated pickup table, utilizes the articulated arm to secure a can of coke, and then navigates to the specified input table for deposition.

Future Work

- Add more dynamic obstacles to the robot path.
- Grasping objects of different shapes and sizes based on object segmentation.
- Placing the objects smoothly on the table.

References

1. https://github.com/ICube-Robotics/scara_tutorial_ros2
2. https://automaticaddison.com/how-to-set-up-the-ros-navigation-stack-on-a-robot/
3. https://github.com/aniketmpatil/scara_robot

Thank You !