Cooking System: Grasp Object - Final Report

Jiwei Wang, Jielin Zhu, Yidong Li

January 17, 2025

Abstract

This report presents the final outcomes of the "Cooking System: Grasp Object" project, which aimed to automate kitchen tasks using the Fetch robot. The system successfully integrated perception, motion planning, and manipulation modules to enable the robot to autonomously grasp and manipulate various kitchen objects. Key achievements include high accuracy in object recognition, successful grasp execution, and efficient task completion. The project demonstrates the potential of robotics in reducing human effort and improving efficiency in both home and professional kitchens.

1 Introduction

1.1 Project Motivation

The motivation behind this project is to enhance kitchen automation by utilizing the Fetch Robotics arm to perform object grasping tasks using traditional robotic methods. The primary goal is to develop a robotic system capable of autonomously and efficiently grasping a wide variety of kitchen objects, such as utensils, ingredients, and cookware.

1.2 System Overview

This project leverages the Fetch Robotics arm integrated with the Robot Operating System (ROS). Specific ROS packages, such as `fetch_ros` and `moveit`, are used to enable precise control and motion planning for the arm. By focusing on traditional grasping methods, the system ensures reliable and repeatable motions for securely handling kitchen objects of various sizes, shapes, and materials.

1.3 Technical Approach

The Fetch Robotics arm is equipped with advanced sensors that enable it to recognize object features, such as shape, orientation, and size. Based on these features, the system adapts its grasping strategy using a combination of motion planning, image recognition, and force feedback. This allows the robot to determine the optimal approach for securely grasping each object. For example, the system can retrieve a spoon from a drawer, place an ingredient in a pot, or handle fragile objects like glasses with precision.

1.4 Objectives and Contributions

The primary objective of this project is to automate mundane kitchen tasks, such as retrieving and manipulating objects, while ensuring high levels of accuracy and safety. By utilizing the ROS framework and packages like `fetch_ros` and `moveit`, the system provides the necessary tools for motion control, object recognition, and real-time feedback. This enables the Fetch arm to operate effectively in typical kitchen environments, paving the way for more intelligent kitchen automation.

2 State of the Art

2.1 Literature Review

The project builds on several state-of-the-art technologies and research papers, including:

- **CuRobo** [1]: A parallelized collision-free motion generation framework.
- **ASGrasp** [2]: A method for transparent object grasping using RGB-D cameras.
- **RGBD-Grasp** [3]: A 7-DoF grasp pose detection pipeline.
- **MoveIt! Task Constructor** [4]: A framework for task-level motion planning.

2.2 Technological Advancements

These technologies influenced the design and implementation of our system, particularly in the areas of motion planning, object recognition, and grasp synthesis.

3 System Description

3.1 Overview

The system consists of hardware and software components that work together to enable the Fetch robot to perform kitchen tasks autonomously.

3.2 Hardware Components

The Fetch robot is equipped with a 7-DoF arm, a gripper, and onboard sensors such as a 3D camera and LiDAR. These components enable the robot to perceive its environment and manipulate objects.

3.3 Software Components

- **ROS**: The backbone of the system, providing communication between modules.
- **fetch_ros**: A package for controlling the Fetch robot's hardware.
- **fetch_gazebo**: A simulation environment for testing and development.
- **moveit_grasps**: A package for generating grasp poses.

3.4 Key Functional Modules

- **Perception Module**: Handles object recognition and localization.
- **Motion Planning Module**: Generates collision-free trajectories.
- **Manipulation Module**: Controls the gripper for object handling.
- **Control Module**: Manages task sequencing and error recovery.

4 Implementation

4.1 Navigation

The navigation system is a critical component of our project, enabling the Fetch robot to autonomously move from a starting position to the kitchen counter. This capability is essential for the robot to perform subsequent tasks such as grasping and placing objects, as well as operating the water tap. The navigation process can be divided into two main phases: map building and code execution. Below, we describe each phase in detail.

4.1.1 Map building

Before the robot can navigate autonomously, it requires a map of the environment. This map serves as a reference for localization and path planning. Using fetch_navigation package, we control the robot by remote control to move through the experimental environment and build out a map. Then we use the map saver to create two files in the specified map_directory. The first is the map in a .pgm image format (Fig. 1), and the second is a YAML file that specifies metadata for the image. The generated map is validated by visualizing it in a tool like RViz (a ROS visualization tool).

4.1.2 Navigation

Once the map is created, the robot can use it to navigate autonomously. The navigation process is implemented using the fetch_navigation package and involves the following steps:

- Initialization: The navigation system is launched with a pre-generated map of the environment, which provides the robot with a reference of the environment. Note that the navigation system is launched using the fetch_navigation package. This package initializes the necessary components for navigation, including localization, path planning, and obstacle avoidance.
- Initial Pose Setting: Before the robot can navigate, its initial pose (position and orientation) must be set within the map frame. This is done by publishing the robot's initial coordinates (x, y) and orientation (theta) to the /initialpose topic. The robot uses this information to localize itself within the map.
- Goal Setting: The target position in front of the kitchen counter is defined as the navigation goal. This goal includes the desired coordinates (x, y) and orientation (theta) relative to the map. The goal is sent to the move_base action server, which is responsible for planning and executing the path to the goal.

Figure 1: Fetch navigation map

- Path Planning and Execution: Once the goal is received, the fetch_navigation package computes a collision-free path to the target position. The robot follows this path while continuously updating its position and avoiding obstacles. During navigation, the robot uses feedback from its sensors to adjust its trajectory in real time.

4.2 grasping and placing

This section aims to enable the fetch robot to grasp an object, and place toe object into the sink. The system can be divided into 2 parts:

- Recognizing the table plane and the object item.
- Motion Planning through picking the item and delivering it to the object place.

The first part involves point cloud generation, plane and object recognition. The second part mainly covers motion planning.

4.2.1 Visual recognizing

The module first acquires the point cloud data of the scene from the depth camera. In order to reduce the amount of computation and remove unnecessary background points, the point cloud is straight-through filtered to retain points closer to the camera.

Next, major planes are extracted from the filtered point cloud. The plane segmentation uses the RANSAC algorithm in conjunction with the camera head tilt angle to determine the direction of the normal vector of the plane.

Next, the point cloud of objects located above the plane is segmented from the point cloud of the plane. By calculating the convex envelope boundary of the plane, combined with the height range, the point cloud of the grasping object is extracted. Then, the

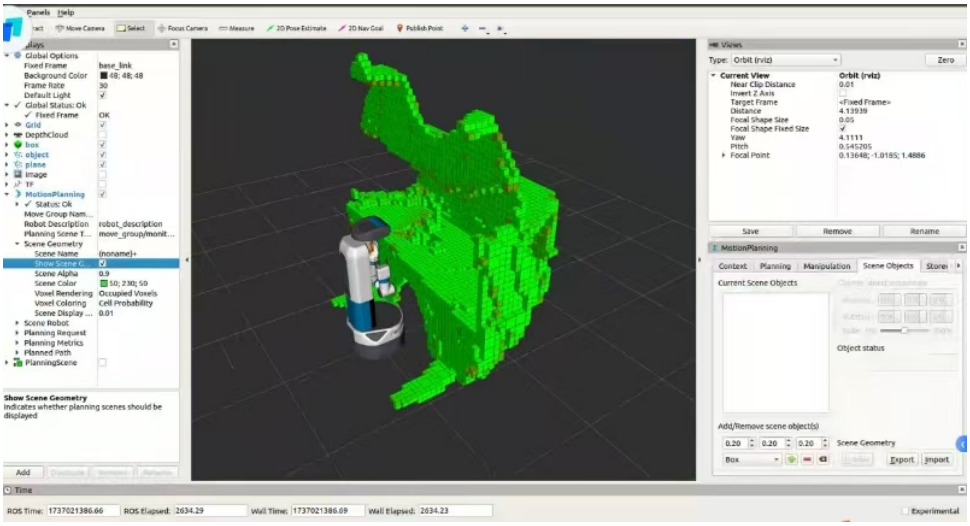

Figure 2: Octomap and plane recognition

extracted object point cloud is analyzed by clustering. After identifying the target object, the Bounding Box of the object is calculated by the bounding box extraction algorithm, and its position and attitude are determined.

Finally, the results of the process are published as ROS messages for use by other modules.

4.2.2 Picking and placing

The process begins with initializing the ROS node and setting up communication with the robot's components, including the arm, gripper, torso, head, and base. The `MoveIt!` framework is used for motion planning, and the environment is prepared by adding collision objects to avoid obstacles during operation.

The object picking procedure starts with looking around and building the OctoMap, to avoid collisions. Then it moves to the object and grasp it using `Moveit!`.

Later, the robot arm moves to the given end point, which is on the top of the water sink. Then the gripper will open, placing the item into the sink.

4.3 Water Tap Operating

After grasping kitchen objects and placing them into the sink, we control the fetch robot to open the water tap for washing. There are two main challenges:

- Localising the water tap handle
- Control the end effector to open the water tap handle

4.3.1 Water tap localization

We choose to use point cloud captured from the RGBD camera to locate the water tap. We believe that this is more stable and reliable compared to image-based vision methods. However, locating the water tap handle in a point cloud is not easy, since it is not in a regular shape like a sphere, box, or cylinder. To enhance the accuracy of localization, we extract the cylinder-shaped water pipe above the platform. The detailed steps are:

- Retrieve the raw point cloud from camera
- Clip points that are too far away
- Extract the platform plane and remove corresponding points
- Apply clustering to remaining points
- The largest cluster should be the water pipe

After getting the water pipe position, we add an offset to get the water tap handle position.

4.3.2 Water tap opening

After getting the water tap position, we use *MoveIt!* to plan a path to open the water tap. When planning the path, we added the platform and the water pipe to the planning scene so that the robot would not collide with them. To better interact with the handle, we first move the end effector to a prepare location that is close to the handle, then move a little bit in to make the handle lie within the gripper, and close the gripper to move the handle, opening the water. After that, the gripper is opened and moved back to the prepare location, and finally move back to idle position. When returning, we added a virtual cylinder representing the running water into the planning scene, so that the arm will not go into water when moving back.

5 Conclusion

5.1 Summary of Achievements

The Cooking System: Grasp Object project successfully demonstrated the feasibility of using the Fetch robot for kitchen automation. Key achievements include:

- **Object Recognition**: High accuracy in recognizing and localizing kitchen objects using point cloud data and advanced algorithms like RANSAC and clustering.
- **Motion Planning**: Integration of the MoveIt! framework enabled collision-free trajectories, ensuring safe and efficient movement.
- **Grasping and Placing**: The robot reliably grasped and placed objects of various shapes and sizes, such as placing items in the sink.
- **Water Tap Operation**: The system localized and manipulated the water tap handle, allowing the robot to open and close the tap autonomously.
- **Navigation**: The Fetch robot autonomously navigated to the kitchen counter, avoiding obstacles and accurately localizing itself.

These accomplishments highlight the potential of robotics in automating repetitive kitchen tasks, reducing human effort, and improving efficiency.

5.2 Impact

The project has significant implications for both home and professional kitchens:

- **Home Kitchens**: Assists individuals with limited mobility, enhancing their independence.
- **Professional Kitchens**: Streamlines operations, reduces labor costs, and improves consistency in food preparation.
- **Future Applications**: The technologies can be extended to healthcare, manufacturing, and logistics, where precise manipulation and navigation are required.

5.3 Final Thoughts

This project marks a significant step in applying robotics to everyday environments. By integrating perception, motion planning, and manipulation, the system shows potential for robots to perform complex tasks in unstructured settings like kitchens. Future work could focus on improving adaptability, robustness, and human-robot interaction.

Overall, the project lays a strong foundation for future innovations in kitchen automation, paving the way for more intelligent and autonomous systems in daily life.

References

[1] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos *et al.*, "Curobo: Parallelized collision-free robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8112–8119.

[2] J. Shi, Y. A, Y. Jin, D. Li, H. Niu, Z. Jin, and H. Wang, "Asgrasp: Generalizable transparent object reconstruction and grasping from rgb-d active stereo camera," 2024. [Online]. Available: https://arxiv.org/abs/2405.05648

[3] M. Gou, H.-S. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu, "Rgb matters: Learning 7-dof grasp poses on monocular rgbd images," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 459–13 466.

[4] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "Moveit! task constructor for task-level motion planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 190–196.