WM-3 - Intelligent Robot Arm

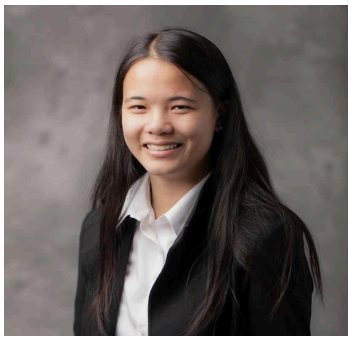
Development Document

CS 4850 – Section 03

Fall 2024

Professor Perry

November 05, 2024

 Zhiwen Zheng Team Leader	 Ellie Ireland Developer

Team Members:

Name	Role	Cell Phone / Alt Email
Zhiwen Zheng (Team Lead)	Documentation and Test	678-818-5275 zhiwen20010111@gmail.com
Ellie Ireland	Developer	863-221-3999 elirel973@outlook.com
Waqas Majeed	Project Owner and Advisor	470-578-6005 wmajeed@kennesaw.edu

1.0 Individual Concepts

Our selection of robotic devices allows us to use ROS 2 for development, providing us with an open source Linux-based platform equipped with libraries and resources. This is what we are using to develop controllers for each joint and subsystem of the system. We perform all of our tasks in simulation, due to lack of hardware resources; ROS 2 also provides us with virtual environments, which we are testing in. Additionally, we are working with NVIDIA's Isaac Sim.

1.1 Manipulator

We are using the OpenManipulatorX as the robotic arm for the mobile manipulator. This unit is responsible for pose estimation, based on camera data parsing and output, and optimal grasp selection, which is the execution of that pose estimation.

1.2 Mobile Robot

For the mobile robot base, we are using the Turtlebot3 Waffle Pi. This subsystem performs the tasks of autonomous navigation and object recognition. Being open source, the robot is modifiable with different pieces of hardware and peripherals. However, we are interfacing with the default parameters which it comes with, including a Raspberry Pi microcomputer, a LIDAR sensor, and a camera module. These components allow us to perform the machine learning-based tasks by utilizing machine vision from its provided sensors and GPIO devices.

2.0 Outline and Implementation Details

As we are using two different simulation platforms, we have the division of work split between them. Autonomous navigation development will take place in both Gazebo and the Isaac Sim. However, the main tasks of pose estimation and optimal grasp selection will be delegated to Gazebo development, while object recognition will be performed with the Isaac Sim.

2.1 ROS2 and Gazebo

2.11 Autonomous Navigation

In ROS2, the autonomous navigation is handled with the navigation library, Nav2. This allows the robot to travel from a start point to a designated end point using behavior trees, which are customized based on the obstacles in the map the robot resides in using independent modular servers computing each path and control system. The Turtlebot3 Waffle Pi, being a differential driven mobile robot, is then able to handle the commands given by the navigation stack to avoid collisions with different objects.

2.12 Pose Estimation and Optimal Grasp Selection

Both pose estimation and optimal grasp selection is controlled with the ROS2 grasp library. This utilizes the OpenVINO toolkit, which, once installed, will indicate grasp approach directions, angles, and grasp boundaries the robot may process. This is done through an RGBD camera after

hand-eye calibration is performed so that the manipulator knows the transfer function of its location in reference to the camera input.

2.2 NVIDIA Isaac Sim

2.21 Autonomous Navigation

In the simulation environment of Isaac Sim, the autonomous navigation will be based on the navigation package and the ROS 2 bridge in Isaac Sim. Because the models are ROS-based, the ROS 2 bridge can let Isaac Sim to get access to the control of the model and use the ROS command and packages to support the simulation, like RVIZ. When setting up the navigation, the map of the environment, model definition, navigation parameters and some other files should be defined before starting the navigation.

2.22 Object Recognition

The object detection in Isaac Sim is related to the built-in object detection module, but before doing this, the models should have sensors and they have already been set up. And then, because the models need ROS 2 to help control, the camera or sensor image should be able to transfer to the ROS 2 topic. When the ROS 2 subscribes to the Isaac camera image topic, the images will be pre-processed to resize or normalize.

3.0 Project Setup

3.1 Setup Tutorial

1. Install ROS2. If using a Windows machine, use Windows subsystem for Linux.
2. Clone the Turtlebot3 Waffle Pi and OpenManipulatorX model from GitHub into the ROS2 workspace and build the package.
    a. $ git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3_manipulation.git
    b. colcon build
3. Fix the MoveIt config by defining the planning groups to allow the base as well as the joints and end effector to be controlled.
    a. ros2 launch moveit_setup_assistant setup_assistant.launch.py
4. Ensure the ROS2 nav2 packages are installed and integrate that into your package launch.
    a. sudo apt install ros-humble-navigation2
    b. sudo apt install ros-humble-nav2-bringup
    c. sudo apt install ros-humble-turtlebot3-gazebo
5. Map the area manually with teleop (using keyboard to control robot) and save the map. We will import this as the known map when we launch autonomous navigation.
    a. ros2 launch turtlebot3_cartographer cartographer.launch.py
    b. ros2 run turtlebot3_teleop teleop_keyboard
    c. ros2 run nav2_map_server map_saver_cli -f ~/map
6. Import the ROS2 grasp libraries and OpenVINO toolkit to the package

    a. sudo apt-get install libpcl-dev libeigen3-dev
    b. git clone https://github.com/intel/ros2_grasp_library.git
    c. git clone https://github.com/openvinotoolkit/openvino.git
7. Implement grasp planning into the mobile manipulator package.
8. Run simulation, troubleshoot, and test