MARYLAND APPLIED
GRADUATE ENGINEERING

Z. Kootbally
Fall 2023
UMD
College Park, MD

Final (v1.0)

ENPM809Y: Introductory Robot Programming

Due date: **Tuesday, December 19, 2023, 11:59 pm**

Contents

1 Changelog

- **v1.0** (12/14/2023): Original release of the document.

2 Disclaimer

Certain commercial products or company names are identified in this document to describe examples of robots, batteries, and sensors. Such identification is not intended to imply recommendation or endorsement by the University of Maryland, nor is it intended to imply that the products or names identified are necessarily the best available for the purpose.

3 Conventions

In this document, you will find a number of text styles that distinguish between different kinds of information.

- link *folder* *file*
- Important note.
- List of tasks to do.
- T topic N node M message F frame A action P parameter

4 Prerequisites

- Part of the code from RWA3.
- Knowledge on mapping and navigation.

5 Objectives

- This is a group assignment.
- This assignment consists of the following steps:
  - Create a map of the environment (already provided).
  - Load the generated map (already performed).
  - Read one ArUCo marker and retrieve the appropriate parameters.
  - Detect parts using logical cameras.
  - Write an action client to go through each part using the order described by the parameters.

- Learning outcomes and skills to be developed:
  - Creating a ROS package.
  - Writing ROS node(s), publishers, and subscribers.
  - Broadcasting and listening of frames.
  - Using parameters.
  - Writing an action client to go through multiple poses.

6 Setup and Installation

- ArUco marker detection requires the latest version of OpenCV.
  - `pip3 uninstall opencv-python`
  - `pip3 uninstall opencv-contrib-python`
  - `pip3 install opencv-contrib-python`
- To install ROS packages for this assignment, it is recommended to create a new workspace.
  - `cd`
  - `mkdir -p ~/final_ws/src`
  - `cd final_ws`
  - `git clone https://github.com/zeidk/enpm809Y_fall2023.git -b final src`
  - `rosdep install --from-paths src -y --ignore-src`
  - `colcon build`
- Create your package for this assignment using your group name as the package name, e.g., `ros2 pkg create group1_final ...`
  - This package should have at least the following dependencies (add more later if needed): rclcpp, mage_msgs, geometry_msgs, std_msgs, tf2_ros, tf2, and tf2_geometry_msgs
  - Create a *config* folder in your package and place *waypoint_params.yaml* in this folder. This file should be loaded with your node.
    - Move (cut and paste) *waypoint_params.yaml* from *final_project/config* to the *config* folder of your package.
- Since we are now using a new workspace, make adjustments in your *.bashrc|.zshrc* file.

7 Task Description

This section describes all the tasks you need to complete.

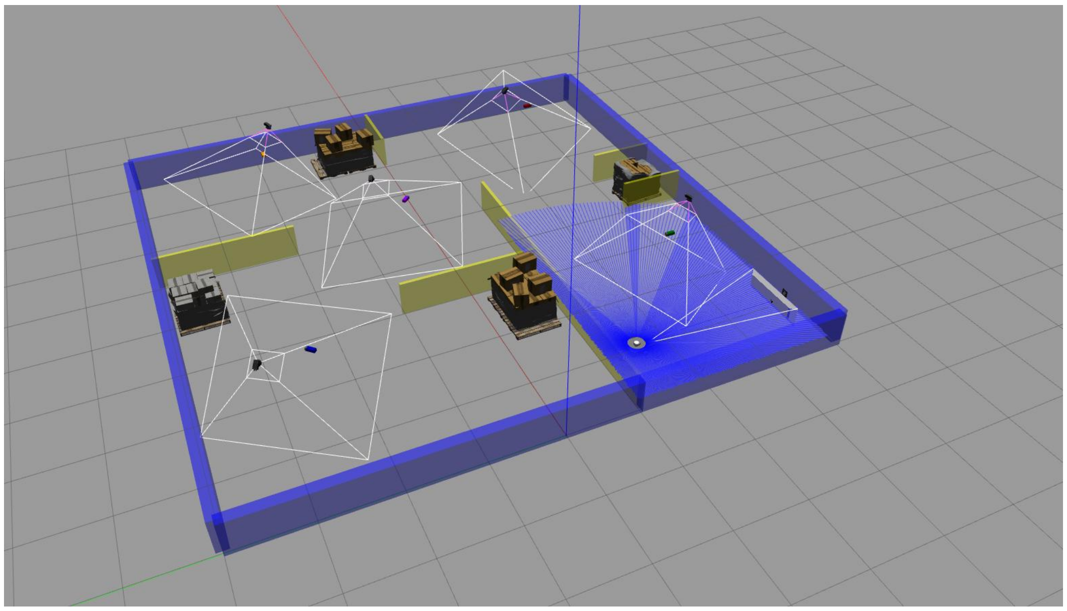

Figure 1: Environment for the final project.

7.1 Start the Simulation

First, start the simulation with `ros2 launch final_project final_project.launch.py`

The Gazebo environment seen in Fig. 1 should start. This environment consists of:

- 1 turtlebot.
- 5 static logical cameras.
- 5 parts: 1 x blue battery, 1 x red battery, 1 x green battery, 1 x orange battery, and 1 x purple battery.

An RViz window should also start with a map of the environment (see Fig. 2).

7.1.1 Turtlebot

For the final project, we are using the original waffle turtlebot. This means that the robot does not have any logical camera attached to it. If you see a logical camera on the turtlebot then it means you are still using packages from RWA3 and you should get this fixed.

As seen in RWA3, the RGB camera can be used to to detect ArUco markers. When an ArUco marker is detected by the camera, its information is published on the topic T /aruco_markers. ArUco markers are reported in the RGB camera frame F camera_rgb_optical_frame

For this final project, only one ArUco marker is used and the robot's camera points to it when you start the simulation.

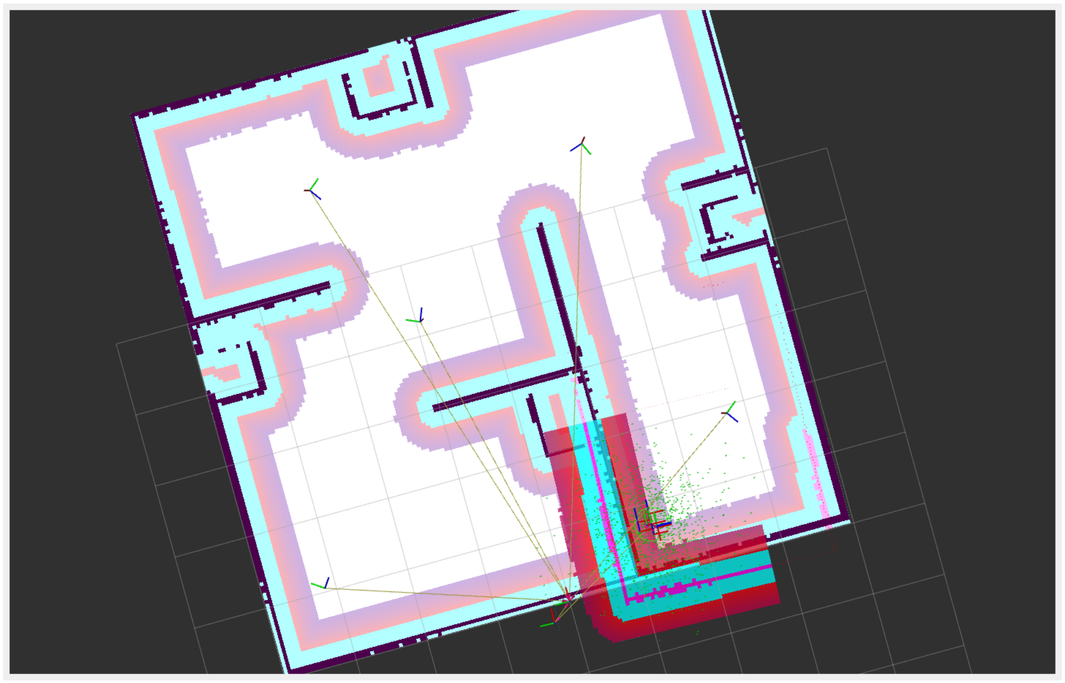

Figure 2: RViz after initializing the robot's pose in the map.

7.1.2 Logical Cameras

The logical cameras are static as they do not move during the simulation. Each camera publishes to its own topic and each camera has a frame.

- ***Camera 1***: T /mage/camera1/image and F camera1_frame
- ***Camera 2***: T /mage/camera2/image and F camera2_frame
- ***Camera 3***: T /mage/camera3/image and F camera3_frame
- ***Camera 4***: T /mage/camera4/image and F camera4_frame
- ***Camera 5***: T /mage/camera5/image and F camera5_frame

7.1.3 Frames

It is highly recommended you visualize, at least once, the frames generated for this final project.

7.2 Read the ArUco Marker

Reuse the code from RWA3 to parse messages published to T /aruco_markers. For this final project, we only need the id of the marker. Once you receive the id of the marker stop your subscription to the topic T /aruco_markers as we don't need this topic anymore and to also decrease bandwidth usage. Do your research on how to stop subscribing to a topic (it involves resetting the subscriber object).

7.3 Retrieve Parameters

After starting the simulation, your next step is to retrieve parameters associated to the ArUco marker id. These parameters are located in *waypoint_params.yaml* and should be loaded with your node (don't forget to edit this file to use your node's name). Do your own research to figure out how to retrieve nested parameters.

You need to map the parameters to the id of the detected ArUco marker. For instance, if the id is 0, then you need to retrieve the nested parameters for the parameter P /aruco_0, if the id is 1 then you need to use P /aruco_1, and so on.

Below is the content of *waypoint_params.yaml*. These parameters will be used to navigate your robot through different poses in a specific order. For instance, if the detected ArUco marker has the id 0 then you will need your robot to navigate through different (x,y) position of each part in the following order: 1) green battery, 2) red battery, 3) orange battery, 4) purple battery, and 5) blue battery.

```yaml
your_node_name: # edit this line
  ros__parameters:
    aruco_0:
      wp1:
        type: 'battery'
        color: 'green'
      wp2:
        type: 'battery'
        color: 'red'
      wp3:
        type: 'battery'
        color: 'orange'
      wp4:
        type: 'battery'
        color: 'purple'
      wp5:
        type: 'battery'
        color: 'blue'
    aruco_1:
      wp1:
        type: 'battery'
        color: 'blue'
      wp2:
        type: 'battery'
        color: 'green'
      wp3:
        type: 'battery'
        color: 'orange'
      wp4:
        type: 'battery'
        color: 'red'
      wp5:
        type: 'battery'
        color: 'purple'
```

7.4 Find Parts

Once you have retrieved the parts information from the parameters, you need to compute the part poses in the F map frame. You need a subscriber for each camera topic. After receiving part information from cameras and after transforming these poses in the F map frame, stop subscribing to these topics as we don't need them anymore.

7.5 Navigate Through Poses

The last step of this assignment is to write an action client to make the robot go through each pose in the correct order. In the current version of this final project, we should see the robot placing itself below the green, red, orange, purple, and blue batteries, respectively. In class we used the action server A /navigate_to_pose to move the robot to a single pose.

For this project you will use the action server A /navigate_through_poses

You need to initialize your robot using programming, as seen in class. Also, when doing navigation, the topic to use to retrieve the pose of the robot is T /amcl_pose and not T /odom

8 Documentation

All classes, methods, and attributes must be documented using Doxygen. For this assignment, there is no need to generate the HTML documentation. You only need to document your code.

9 Submission

- Place *Readme.txt* in the root folder of your package. In this file, include instructions on how to run your code. You will lose 5% of your grade if the instructions are not correct. With the large number of groups, our TA and grader cannot chase students who provided wrong instructions.
- Zip your package only and submit it. You will lose 5% of your grade if you submit anything else than your package. If you include the packages I have provided in your submission, `colcon build` will not work because of duplicate packages.
- Make sure you submit the correct package as we will not accept anything else passed the deadline.
- Any submission passed the deadline will incur a penalty, even if it is 1 minute after the deadline. Rules which pertain to penalty can be found in the syllabus. We were lenient in the past about late submissions but as we are reaching the end of the semester you are required to submit the assignment by the due date.
- There is no extension for this assignment unless you provide documented justifications. If you have other submissions due around the same time, you are responsible for organizing your work to submit the assignment on time.
- A grade of 0 (zero) will be assigned for any plagiarized submission.

- A grade of 0 (zero) will be assigned for hardcoded information. For example, parameters should be retrieved using ROS C++ methods and not hardcoded. If we change the value for some parameters, your code will not work if you hardcode parameter information. In the same way, the color and the type of detected parts have to be retrieved from M mage_msgs/msg/Part (we saw something similar in class with constants in msg files). If you are not sure about what is allowed/not allowed to hardcode, ask us.

10 Grading Rubric

The grading of this assignment consists of 4 sections

- 80% – Code completion, which includes everything described from Sections 7.2 to 7.5.
- 10% – Your code is properly documented AND commented. Make sure you comment long methods.
- 10% – C++ and ROS best practices are followed. Include guards, naming conventions, etc. During office hours I saw many of you are not following naming conventions.

11 Note

- We received to emails with conflicting dates on the grade submission deadline. One says 12/20 as the hard deadline and the other says 12/22 as the deadline. While we are trying to figure this out, we will use 12/19 as the current submission deadline.
- I originally planned to include an extra activity to this project to count as bonus points. If you are interested, the activity is as follows:
  - Complete the final project without using a stored map. This requires performing actual SLAM and is harder than using a generated map.