ENPM702: Introductory Robot Programming
Take-Home Final Project

version 1.1

Lecturer: Z. Kootbally

School: University of Maryland

Semester/Year: Fall/2024

2024/12/02

Table of Contents

Changelog

Changelog

- ☐ v1.1: Added context on `rclcpp::SensorDataQoS()` in slide 12.
- ☐ v1.0: Original version.

Convention

General

- ☐ 🗎 ***file***
- ☐ ***folder***
- ☐ link
- ☐ `command`
- ☐ note
- ☐ warning

ROS

- ☐ n **node**
- ☐ t **topic**
- ☐ m **message**
- ☐ f **frame**

Prerequisites

Retrieve Packages

☐ It is recommended you create a new workspace for this project, for example:

```
mkdir -p ~/final_enpm702_ws/src
```

☐ Use the following instructions to retrieve only the folder ***final_home*** from https://github.com/zeidk/enpm702_fall2024_ros

- ☐ `cd ~/final_enpm702_ws/src`
- ☐ `git clone --no-checkout https://github.com/zeidk/enpm702_fall2024_ros.git`
- ☐ `cd enpm702_fall2024_ros`
- ☐ `git sparse-checkout init`
- ☐ `git sparse-checkout set final_home`
- ☐ `git checkout`
  - ☐ Ensure you now have the structure below:

```
enpm702_fall2024_ros
└── final_home
    ├── final_project
    ├── mage_msgs
    └── part_publisher
```

Figure: Provided packages for the final project.

Prerequisites

Start Environment

- ☐ Build your new workspace:
  - ☐ `cd ~/final_enpm702_ws`
  - ☐ `rosdep install --from-paths ./src --ignore-packages-from-source -y`
  - ☐ `colcon build`
  - ☐ Source the workspace.
- ☐ Start the environment:
  - ☐ `ros2 launch final_project final_project.launch.py`
  - ☐ A Gazebo environment will initialize with a TurtleBot, cameras, and parts floating in midair. See Figure in the next slide.

Prerequisites

Gazebo Environment

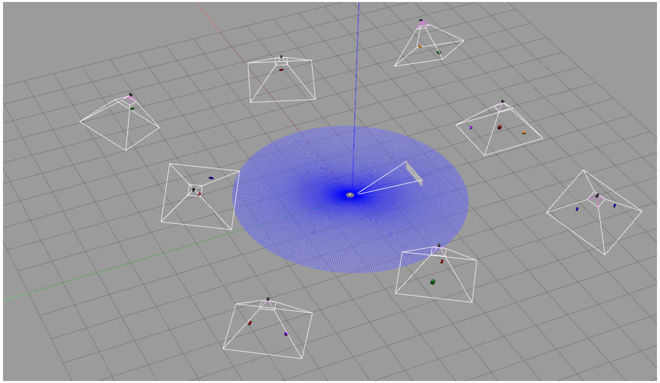
Figure: Gazebo environment.

The environment consists of:

- ☐ 1 Turtlebot.
- ☐ 8 cameras.
- ☐ 15 parts:
  - ☐ Batteries:
    - ☐ 3 x red battery
    - ☐ 3 x blue battery
    - ☐ 1 x green battery
    - ☐ 1 x orange battery
    - ☐ 1 x purple battery
  - ☐ Pumps:
    - ☐ 2 x green pump
    - ☐ 1 x red pump
  - ☐ Regulators:
    - ☐ 1 x purple regulator
    - ☐ 1 x orange regulator
    - ☐ 1 x red regulator

Guidelines

This assignment must be completed ***as a group***. Ensure compliance with all specified guidelines, as failure to follow any part of these guidelines will lead to a grade of ***zero*** for the assignment ***for the whole group***.

- ☐ Do not reuse a package obtained from another group.
- ☐ Keep your work confidential and refrain from sharing it with peers.
  - If you use github, you have to make your repository private.
- ☐ While discussing assignment challenges is encouraged, refrain from exchanging code with other groups.

Objectives

Objectives

The goal of this project is to develop a ROS 2 node that autonomously navigates a TurtleBot in a Gazebo environment. The robot will locate specific parts using camera data, transform their positions from the camera frame to the world frame, and position itself under the parts using a proportional controller. After completing all tasks, the robot will return to its starting position.

Skills Practiced

Skills Practiced

- ☐ ROS 2 Programming
- ☐ Processing data from ROS topics.
- ☐ Converting part poses from the camera frame to the world frame.
- ☐ Implementing a proportional controller to guide the robot to precise positions.
- ☐ Combining sensor data acquisition, transformations, and navigation into a cohesive solution.

Package

- ☐ Create the package ***group<number>_final***.
    - ☐ Within this package, you are free to create as many nodes and launch files as necessary.
    - ☐ In this package, create a folder named ***doc*** and include the file ***readme.txt***, which provides instructions on how to run your code.
- ☐ If additional packages are required for this project, organize them under a different structure. Place all your packages in the folder ***group<number>_final_meta***, as illustrated below.

```
group<number>_final_meta
├─ group<number>_final
│  ├─ doc
│  │  └─ readme.txt
│  └─ ...
├─ <other package>
├─ <other package>
└─ ...
```

Figure: Directory structure for multiple packages.

Requirements

Retrieve Parts List

- ☐ Subscribe to the t **/parts** topic to retrieve a list of parts, including the color and type of each part your robot needs to visit. The message type published to this topic is m **mage_msgs/msg/Parts**, which contains an array of m **mage_msgs/msg/Part**.
  - ☐ Each published message consists of the same 4 parts in a loop. You only need to process one message and ignore the remaining ones. There are two ways to achieve this:
    - ☐ Use a flag in the subscriber callback to ignore the remaining messages.
    - ☐ Call the `reset()` method on the subscriber object to release the subscription.
      - ☐ This approach is a bit more advanced and improper use may crash your node.

Requirements

Process Camera Data

- ☐ Subscribe to topics from fixed cameras in the Gazebo environment to locate parts. There are 8 camera topics, formatted as t **/mage/camera<n>/image**.
  - ☐ Since the parts are static, once you have retrieved the parts from each camera, you can ignore the remaining messages published on these topics. Refer to the previous slide for suggested approaches.
  - ☐ You have to use `rclcpp::SensorDataQoS()` with your camera subscribers. For instance:

```
camera1_sub_ = this->create_subscription<mage_msgs::msg::AdvancedLogicalCameraImage>(
    "/mage/camera1/image", rclcpp::SensorDataQoS(),
    std::bind(&RobotTargetClient::camera1_callback, this, std::placeholders::_1));
```

- ☐ Transform part positions from the camera frame to the world frame using the appropriate transformation logic.
  - ☐ Use one of the approaches covered in class to visualize available frames.
  - ☐ The pose of each camera in the f **world** frame is provided by the field `sensor_pose` from t **/mage/camera<n>/image**.
- ☐ The topic t **/parts** only provides a list of parts. For example, a **red battery** is part of that list. You must locate all red batteries in the environment, as your robot will need to visit each **red battery** only once.
  - ☐ The robot must reach each part only once.

Requirements

Navigate to Parts

- ☐ Use the proportional controller from your previous assignment to guide the TurtleBot to position itself directly beneath each part, use $\theta = 0$.
  - ☐ The robot should sequentially move to each part.
- ☐ Once the robot has reached each part, make it return to $(0, 0)$ and shut down your node.

Evaluation

Evaluation

- ☐ Your code will be evaluated by publishing a different message to t **/parts**, ensuring that your code is flexible and does not rely on hardcoded values that should be retrieved dynamically. The evaluation will include:
  - ☐ A different number of parts. Do not assume that 4 parts will always be published to t **/parts**.
  - ☐ Different part types and colors. Parts that are not available in the environment will be used during testing. Your code should be robust enough to handle this scenario. In such cases, you can simply ignore these parts and print a message for the user. Example:

    ```
    Purple pumps are not available in the environment; they will be skipped.
    ```
- ☐ Test with various part numbers, types, and colors by modifying ***part_publisher_node.cpp***, located in the package ***part_publisher***.

Deliverables

Deliverables

- ☐ Submit one zipped folder containing your package(s).
  - ☐ All classes, methods, and attributes must be documented using Doxygen. Ensure that each ***.hpp*** file includes a Doxygen header.
    - ☐ Do not include Doxygen-generated HTML files in your submission.
  - ☐ Use comments appropriately throughout your code.
  - ☐ Ensure you are not repeating the same mistakes that were identified in your previous assignments. Review everything carefully.
  - ☐ Edit ***package.xml*** to include a project description and maintainer tags.
  - ☐ Include a ***readme.txt*** file with minimal instructions on how to run your code.
  - ☐ Delete the ***log***, ***build***, and ***install*** directories, then re-run your code to ensure functionality. A significant penalty will be applied if your package does not compile or run successfully on my system.

Plagiarism

Plagiarism, including the inappropriate use of AI tools, is strictly prohibited and will result in serious consequences. All work submitted must be original, properly attributed, and in full compliance with academic integrity guidelines.

While AI tools can assist in learning and brainstorming, using them to generate code, text, or solutions without proper acknowledgment or as a replacement for your own effort constitutes a violation.

It is important to note that previous assignments have already been flagged for potential misuse of AI tools and are currently under investigation.

For group projects, accountability is shared among all members; if one student fails to follow these rules, the entire group will receive a zero. Maintain open communication within your team and uphold ethical standards to protect everyone's academic standing.

Grading Rubric (120 pts)

Retrieve Parts List: 20 pts

- 10 pts Subscription to t **/parts**.
  - **10 pts** Successfully subscribes to t **/parts** and retrieves the list of parts, including color and type. Processes only one message while ignoring duplicates (using a flag or `reset()` method).
  - **5 pts** Partially subscribes or retrieves data with errors (e.g., duplicates are not ignored, or some part data is missing).
  - **0 pt** No subscription or retrieval of parts from the t **/parts**.
- 10 pts Handling Variations in Part Data
  - **10 pts** Handles varying numbers of parts, colors, and types and skips unavailable parts with an appropriate user message.
  - **5 pts** Handles some variations but skips parts improperly or provides incomplete user message.
  - **0 pt** Cannot handle variations or does not output a message for skipped parts.

Grading Rubric (120 pts)

Process Camera Data: 25 pts

- 10 pts Camera Subscription and Data Retrieval.
  - **10 pts** Subscribes to all 8 camera topics, retrieves data correctly, and processes it efficiently, ignoring unnecessary messages after parts are located.
  - **5 pts** Subscribes to cameras but retrieves or processes data with errors (e.g., processes unnecessary messages or misses part data).
  - **0 pt** Does not subscribe to camera topics or retrieve relevant data.
- 15 pts Transformation to f world Frame
  - **15 pts** Correctly transforms part poses from camera frames to the f world frame using the provided transformation logic (broadcaster and listener).
  - **8 pts** Partially correct transformations, with some errors or inaccuracies in f world frame coordinates.
  - **0 pt** No transformations performed or incorrect results for all transformations.

Grading Rubric (120 pts)

Navigate to Parts: 40 pts

- 15 pts Integration of Proportional Controller
  - **15 pts** Correctly uses the proportional controller to guide the robot to each part.
  - **8 pts** Uses the controller but with issues in alignment or navigation precision.
  - **0 pt** Does not use the proportional controller.
- 15 pts Sequential Navigation to All Parts
  - **15 pts** Robot visits all parts sequentially without skipping or revisiting any part.
  - **8 pts** Robot visits most parts but skips or revisits some parts.
  - **0 pt** Robot does not navigate to parts or navigates incorrectly.
- 10 pts Return to Origin and Node Shutdown
  - **10 pts** Robot navigates back to (0, 0) after visiting all parts and node is shut down.
  - **5 pts** Robot attempts to return to origin but does so inaccurately or incompletely.
  - **0 pt** Robot does not return to origin.

Grading Rubric (120 pts)

Code Quality and Documentation: 20 pts

- 10 pts Code Organization and Modularity
  - **10 pts** Code is well-organized, modular, and adheres to ROS 2 and C++ best practices. No repeated mistakes from previous assignments.
  - **5 pts** Code has moderate organization issues or repeats minor mistakes.
  - **0 pt** Code is poorly organized or repeats major mistakes.
- 10 pts Documentation
  - **10 pts** Includes Doxygen headers for all classes, methods, and attributes, along with meaningful comments throughout the code. ***package.xml*** contains project description and maintainer tags.
  - **5 pts** Partially documented with some Doxygen headers or comments missing. ***package.xml*** has incomplete information.
  - **0 pt** Documentation is absent or does not follow required standards.

Grading Rubric (120 pts)

Deliverables and Compilation: 15 pts

- 5 pts Submission Completeness
  - **5 pts** Submission includes all required files and excludes unnecessary directories and files.
  - **3 pts** Submission includes most required files but contains unnecessary directories or files.
  - **0 pt** Submission is incomplete or missing critical files.
- 10 pts Code Compilation and Execution
  - **10 pts** Code compiles and runs successfully without errors.
  - **5 pts** Code compiles with warnings or minor errors that do not significantly impact functionality.
  - **0 pt** Code does not compile or run.