Zeid Kootbally & Craig Schlenoff
University of Maryland
College Park, MD

RWA-2: Coordinate Frames

ENPM809B : Spring 2020
Due **Wednesday, February 26, 2020**

Contents

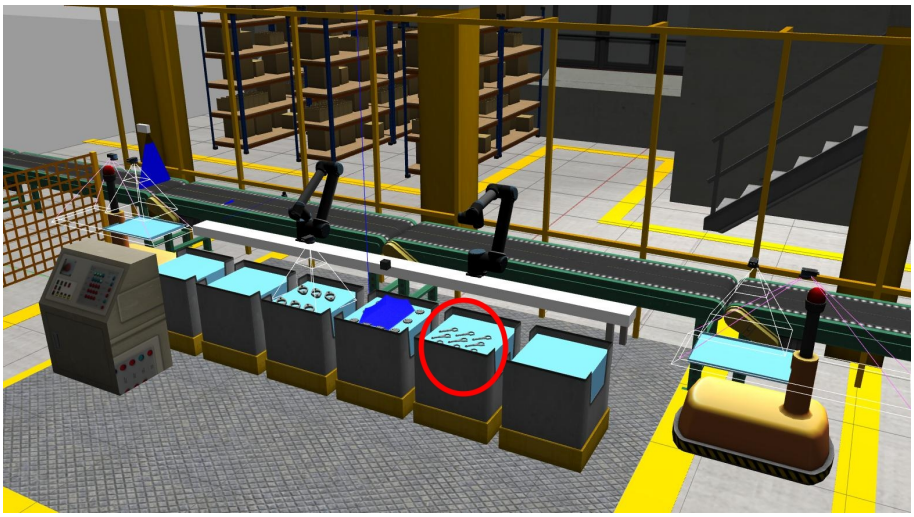

Figure 1: Gazebo environment.

Assignment

The goal of this assignment consists of adding a camera/sensor in the environment and then create a ROS program that outputs sensor/camera information on the screen

Instructions

Create a Competition Package

1. Create a competition package by following these instructions.
   - In the example provided, the name of the package is helloworld_ws, however, you should name your package using the assignment number and your group number, e.g., group1_rwa1.
2. Run Gazebo and the GEAR interface with:
   - `roslaunch osrf_gear sample_environment.launch`
   - sample_environment.launch is located at /opt/ros/melodic/share/osrf_gear/launch/sample_environment.launch

Add Sensor/Camera in Config File

- When the environment starts, you will see a bin of parts which does not have a camera above it (see figure 1).
- Your task is to add a logical camera above this bin. The correct approach to add sensors and cameras in ARIAC is to do it through config files (YAML files).
- If you look at sample_environment.launch, you will see the following section:

```
--visualize-sensor-views
-f $(find osrf_gear)/config/sample.yaml
$(find osrf_gear)/config/sample_user_config.yaml
" required="true" output="screen" />
```

  - `$(find osrf_gear)/config/sample_user_config.yaml` tells ROS to use the sensor configuration file sample_user_config.yaml which is located in the config directory inside the osrf_gear package.

- You need to modify this config file to add the missing logical camera. **However**, you **must not** modify anything installed in the ROS repository (/opt/ros/melodic), instead, you will need to do all the work in your own workspace.

  1. Inside your competition package, create a launch directory.
  2. Inside the launch directory, create a launch file (e.g., group1-rwa1.launch).
  3. Copy and paste the content of sample_environment.launch inside group1-rwa1.launch
  4. Modify `$(find osrf_gear)/config/sample_user_config.yaml` to be `$(find group1_rwa1)/config/sample_user_config.yaml` (**note**: `group1_rwa1` is used as an example).
  5. Inside your competition package, create a config directory.
  6. Inside the config directory, copy and paste sample_user_config.yaml.
  7. Edit sample_user_config.yaml, located in your package to add the logical camera.
     - To get the pose of the camera, you can drag an existing logical camera in Gazebo and place it over the bin, get the pose of the camera in the left pane, and reuse those numbers in your config file.

Read Sensor/Camera Data

- The main function of ariac_example_node.cpp (in the helloworld_ws package) will start the competition, will zero-out the arms, and start callback functions for sensor data.
- When the competition is started, you will see parts moving on the conveyor belt.
- You can also check the state of the competition by querying the topic /ariac/competition_state (see competition interface).
- ariac_example_node.cpp contains the stub of your callback functions but no sensor data is reported.
  - In this file, you need to edit the function `void logical_camera_callback` to output poses of all the parts reported by logical cameras, **in the world frame** (very important).
- Make sure the other callback functions (e.g., `proximity_sensor_callback`, `laser_profiler_callback`, etc) work. For instance, when a part crosses the break beam sensor, I should see the message "Break beam triggered." in the console.

Grading Rubric (15 pts)

3 pts– Create your own competition package:

  - 1.5 pts Did not follow instructions on how to create the ROS package (e.g., missing files, wrongly named package, etc).
  - 3 pts: Followed instructions.

3 pts– Package launch file and config file.

  - 0 pt: No launch file or config file in your package.

  - 3 pts: Followed the guidelines.

8 pts– Reporting sensors/cameras data:

  - 0 pt: You fail at reporting any sensor data.
  - 4 pts: Some sensor/camera data reported but not all of them OR logical camera data reported, but not in the world frame.
  - 8 pts: All sensor/camera data reported and in the world frame.

1 pt– Provide instructions on how to run your program through a Readme.txt, located inside your package.

  - 0 pt: No Readme.txt.
  - 0.5 pts: Readme.txt provided but instructions do not work.
  - 1 pt: Readme.txt provided and program runs.