COMSPS726: Robotics and Intelligent Systems

Assignment: Storage Warehouse: (Weighting 15%)

Robots are cyber-physical devices that interact with the physical world using sensor and actors. They are designed to solve specific problems or tasks. This assignment requires you, the designer to solve the problems of a storage warehouse. The owner wishes to utilise a robot (Pioneer 3 DX) to find and identify all containers and barrels of various sizes within his storage warehouse. The key requirements are as follows:

- The robot is to avoid and not hit any obstacles within the warehouse including the walls.
- The robot is to create a map of the warehouse using SLAM.
- The warehouse has wall 10m by 6m (see Figure 1).
- There are barrels (round object) and containers (rectangle object) scattered around the warehouse with random sizes and orientations.
- The robot is to identify all of the following:
    - The total number of objects.
    - The type (barrel or container), size (radius or width and length) and location (x-y coordinates) of each objects.
    - Assume that all objects are tall enough to be seen by the sensors.
    - Barrels have radiuses from 10cm to 50cm.
    - The containers have lengths and widths from 10cm to 100cm.
    - The sizes of objects are in 2cm increments (i.e., 10cm, 12cm …).
- The robot should be able to return to any object it has found.
- You will using the laser range finder, the odometer, the GMapping ROS library.

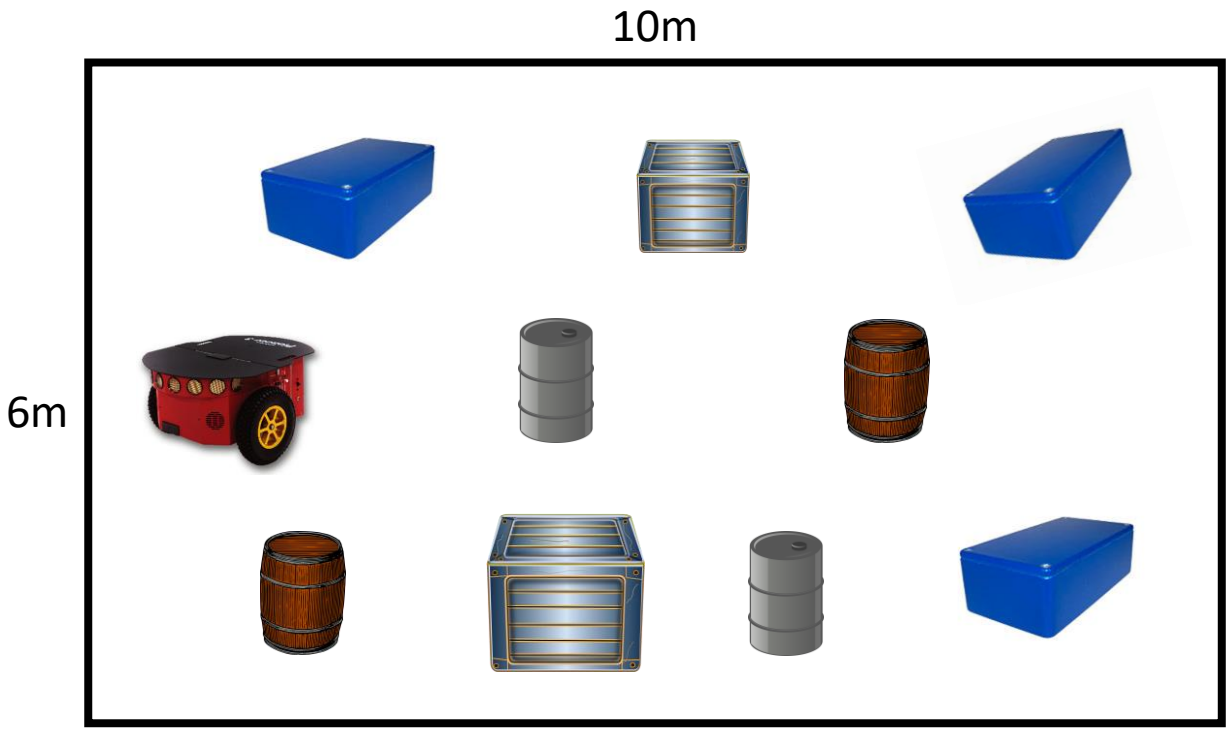


Figure 1 A depiction of the environment

Object identification from here onwards will refer to the determining the type, dimension and x-y location of the object. Once an item is identified it should be printed to the console and stay on the console for a reasonable amount of time (i.e., 5 seconds). The robot is to provide an integer ID for each object it detects.

For example:

```
>Object Found!! – ID: 7
>Type: Barrel, Radius: 20cm, X: 3, Y: 5.23
```

Assignment Components

Part A – Basic simulation – Individual - 5%
Part B – Advanced simulation – Individual - 5%
Part C – Real robot implementation – Group - 3%
Code submission – Individual - Code documentation, functionality, commenting etc. - 2%
-The code should be well documented and be executable and can compile.

Assignment Part A – Basic Simulation (weighting 5% - individually assessed):

Tasks:

1. **Basic object identification:** you will be required to be able to identify a single random object while the robot is stationary.
   1.1. The object will be placed in front of the robot such that all sides can be seen (i.e. 45 degrees rotation) see Figure 2. The object identification has to work when there are walls as well as when there are no wall.

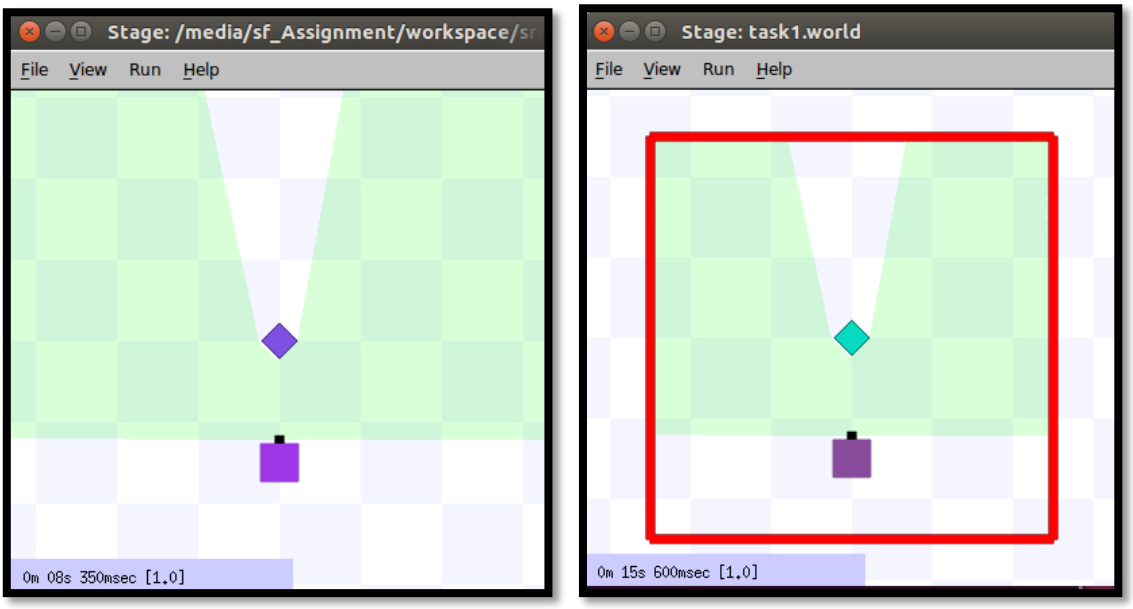


Figure 2: Task 1 simulation example

- In the assignment resources folder, you have been provided a "task1.world" file for you to use. In this file there are circle and rectangle models representing barrels and containers respectively, make sure you open the file and change the parameters so that you test all possible configurations. Note that # means comments for world files.

2. **Mapping and localisation (GMapping):** you will be required to traverse and map all objects of the simulated storage warehouse while avoiding any obstacles. You will need to design an algorithm so that the entire warehouse is examined.

3. **Mobile object identification:** At the same time as (2) (i.e., while the robot is moving), you are required to identify a single object in the map.

   - In the assignment resources folder, you have been provided a world generator executable (GenerateWorld.out) and its source code (GenerateWorld.cpp). This is a basic random object generator for you to test your code. Executing it will generate a random world file (generated.world) for you to use. Due to its basic nature it may take a long time if you are generating many objects, if it becomes stuck while generating simply press "CTRL+C" and retry.

   >./GenerateWorld.out --help

   Usage:

-noBorder	Generate physical border	Default: has border
-noBarrels	Generate barrel	Default: can generate barrels
-noContainers	Generate containers	Default: can generate container
-items n	Generate n items	Default: 1

   Example usage:

   >./GenerateWorld.out -noBarrels -items 10

   This command will place 10 containers in the generated.world file.

Note: to know the dimension of the objects in the map, you can mouse over it to see its coordinates. Then match those coordinates with terminal output of the "GenerateWorld.out" executable or the "generate.world" file. If you get a **permission denied** type "`chmod 777 GenerateWorld.out`" then try again. To rebuild the file type "`g++ GenerateWorld.cpp -o GenerateWorld.out`"

Assignment Part B – Advanced Simulation (weighting 5% - individually assessed):

Tasks:

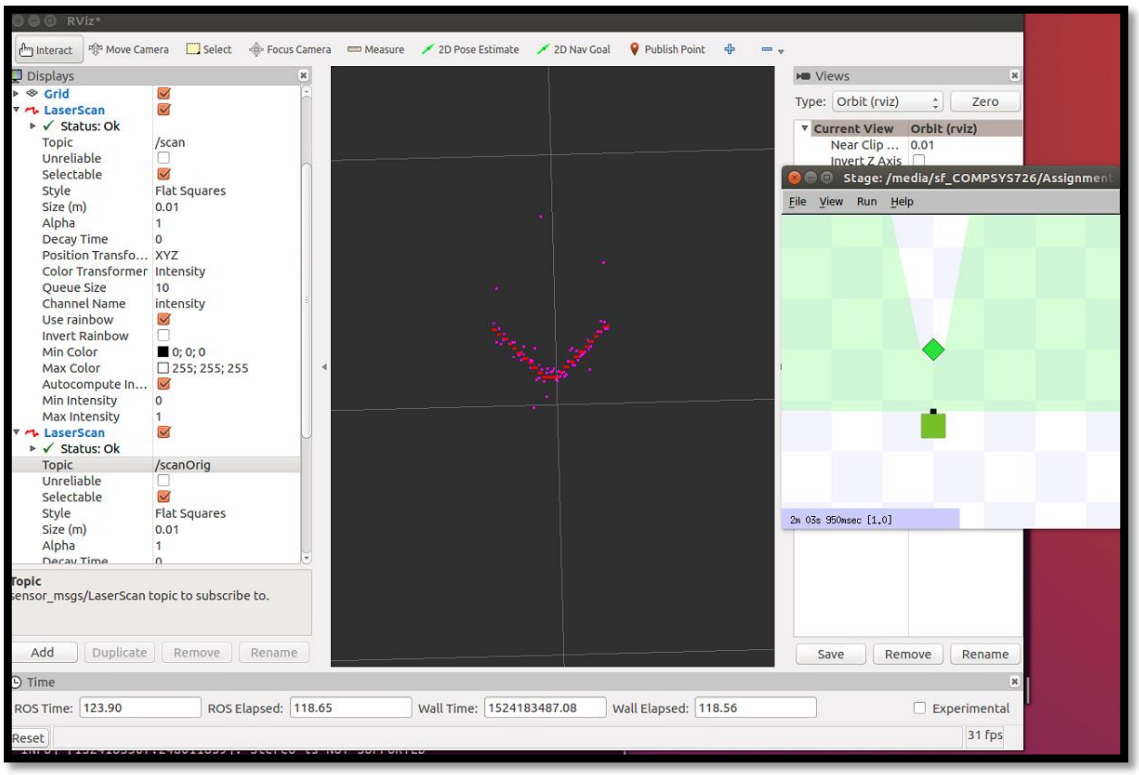
1. **Advanced object identification:** you will be required to be able to identify and count all the objects within the warehouse (from 1 to 10).
2. **Object retrieval:** You will need to create a separate "User Input" ROS node. This node should be executed in a **separate terminal**. The user input node will ask the user continuously the following questions:
   - Type 'r' to enter object "retrieval mode".
   - Enter the object ID:

Once the user has entered "retrieval mode" and the object ID has been set, the "User Input" node should send a message (i.e., string or other type) to your robot controller node. The robot should then stop what it is doing and go straight to the object. It is to stop in front of it for at least 5 seconds, print to the console that it has found the object and spin 360 degrees. The robot should then resume with continue with Task 1 until it has again enter retrieval mode.

3. **Noisy sensors:** you will be required to use the inputs from a noisy sensor node instead of the simulated sensor data directly to execute Task 1 and Task 2. The noise node will generate a random noise on top with the simulated sensor inputs. The noise node introduces a 3cm normally distributed noise on the sensor values and a 10% chance that a normally distributed noise 30cm noise will appear.

   The noise node is provide to you in the resources directory. To use the noise node you will need to do the following:

   - Copy the "custom_noise" and "random_numbers" folder to your source directory.
   - Map the "base_scan" topic from stage to "scanOrig" topic and map the "scan_noise" topic from nose_node to the "scan" topic. An example of how to do this is provided in the "noiseLaunch.launch" file in the resources folder.
   - When view the data using RVIV you can see the "scanOrig" has ideal data point and "scan_noise" will has random noise overlaid.



<u>Assignment Part C – Robot Implementation (weighting 3% - assessed as a group):</u>

In real-life both the laser and the odometer data is noisy. The odometer data can drift due to wheel slip and other factors so you will need to take those into account. Additionally it is important to note that the simulated laser range finder, odometer and motors may not be the same as the actual robot. Therefore, it is always a good idea to use a variable rather than a fixed number. For example, the "angle_max", "angle_min", and "angle_increment" of the laser range finder may be different on the actual robot compared to the simulation.

1. Implement Part B – Task 1 onto the actual robot in RCL.
2. Implement Part B – Task 2 onto the actual robot in RCL.

Assignment Key Dates:

11th May 2018 – Assignment Part A due:

- Demo Assignment Part A

25th May 2018 - Assignment Part B and C due:

- Demo assignment Part B and C

25th May 2018 - Online code (individual) submission of both Part A, B and C

- Include your entire workspace directory for the simulation and the robot.
- Include a "readme.txt" file.
  - A user whom is experienced with ROS should be able to read you readme file and understand how your code functions and builds.
- Marks for any part will only be awarded if there the code for the part is submitted.

Resources:

1) Code resources
   a. Your lab code
   b. Lab resources: "odomExample.cpp" and "slamExample.cpp"
   c. Assignment resources:
      i. custom_noise node
         - Dependencies: random_numbers package
      ii. noiseLaunch.launch
         - Dependencies: task1.world
      iii. task1.world
         - Dependencies: circle.png
      iv. GenerateWorld.out
         - Dependencies: circle.png and template.world
2) Physical objects
   a. Border boards

b. Boxes as containers and buckets as barrels.