THE UNIVERSITY OF HONG KONG

Department of Data and Systems Engineering

DASE7505 Intelligent Unmanned Systems

Lab: Setup of ROS and Sensor Data Processing for Mobile Robots

Objectives:

- By following the instructions in this lab sheet, you should have properly installed ROS packages on your Ubuntu installed in the last tutorial.
- Understand the pipeline of ROS or any middleware.
- Read and Process robot's data; sensors and actuators.

Equipment and Files:

- A PC with the Ubuntu based on VMware Workstation that installed beforehand.
- Downloaded code zip file with the demonstration video from HKU Moodle.

Brief Introduction to ROS

Watch the following video and understand how ROS works, and tools we are going to use often (i.e., Gazebo and Rviz, this mainly include the fundamental knowledge, you may skip this part if you have already learned ROS before or would like to directly try the commands in ROS environment)

YouTube link: https://youtu.be/mjrxf8EFSb8?si=ABkLSVqEVD1ZZ5QM

Step-by-Step Instructions:

I. ROS Setup

1. Open your installed Ubuntu from the VMware Workstation and open the terminal. (It would be a good habit to check the internet connection by using the methods we introduced in the last tutorial).

2. The relevant guidelines are accessible in the GitHub repository: https://github.com/PeterWANGHK/DASE7505_student.git

3. Type the following commands in terminal step-by-step to enable git and clone the repository to the Ubuntu home:
   ```
   sudo apt install git
   git clone https://github.com/PeterWANGHK/DASE7505_student.git
   ```
   ```
   peterwang@peterwang-virtual-machine:~/DASE7505_student$ git clone https://github.com/PeterWANGHK/DASE7505_student.git
   Cloning into 'DASE7505_student'...
   remote: Enumerating objects: 39, done.
   remote: Counting objects: 100% (39/39), done.
   remote: Compressing objects: 100% (36/36), done.
   remote: Total 39 (delta 9), reused 0 (delta 0), pack-reused 0 (from 0)
   Receiving objects: 100% (39/39), 113.45 KiB | 1001.00 KiB/s, done.
   Resolving deltas: 100% (9/9), done.
   ```

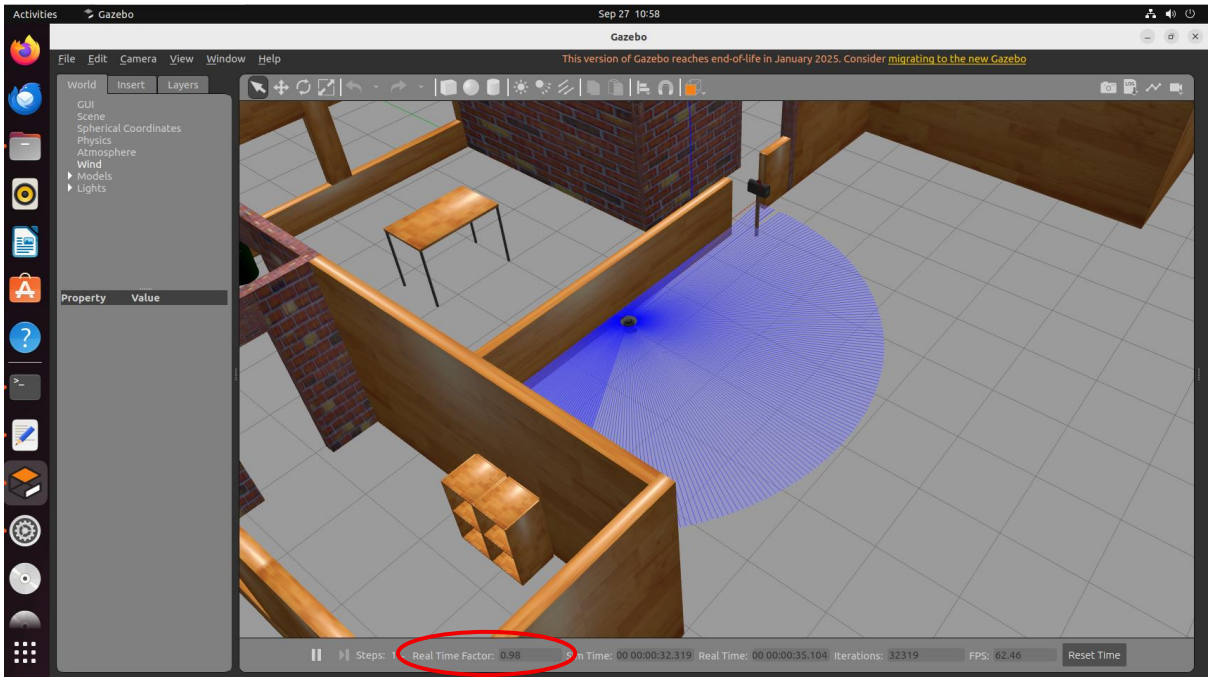
4. In this repository, we have different branches after the main branch. Since we are about to perform ROS setup, we need to change the branch to "setup" and then install relevant file by the following commands one by one: (change to the right directory first)
   ```
   cd DASE7505_student
   git checkout setup
   ```

5. Run the script to install the relevant packages for ROS development:
   ```
   sh setup_dase7505.sh
   ```
   It will automatically start the collection of all required packages for ROS2 and you would see this result if everything went smoothly:
   ```
   peterwang@peterwang-virtual-machine: ~/DASE7505_student
   Setting up libgazebo-dev (11.10.2+dfsg-1) ...
   Setting up ros-humble-gazebo-dev (3.9.0-1jammy.20250701.022615) ...
   Setting up ros-humble-gazebo-ros (3.9.0-1jammy.20250915.221523) ...
   Setting up ros-humble-gazebo-plugins (3.9.0-1jammy.20250915.230654) ...
   Setting up ros-humble-turtlebot3-manipulation-gazebo (2.3.8-1jammy.20250915.2311
   11) ...
   Setting up ros-humble-gazebo-ros-pkgs (3.9.0-1jammy.20250915.235938) ...
   Setting up ros-humble-turtlebot3-gazebo (2.3.8-1jammy.20250916.000008) ...
   Setting up ros-humble-turtlebot3-simulations (2.3.8-1jammy.20250916.000709) ...
   Setting up ros-humble-turtlebot3-gazebo-dbgsym (2.3.8-1jammy.20250916.000008) ..
   .
   Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.8+dfsg-1ubuntu0.3) ...
   Processing triggers for install-info (6.8-4build1) ...
   Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
   Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...
   Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
   Processing triggers for hicolor-icon-theme (0.17-2) ...
   Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
   Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
   Processing triggers for man-db (2.10.2-1) ...
   ------------------------------
   ....turtlebot enviornment is correctly set!....
   ------------------------------
   ```
   A quick check of the ROS version could be performed by typing the following command:
   ```
   ls /opt/ros/
   ```
   In our case, it should output "humble"

6. Check the installation by running a demonstration program by the following commands step-by-step (remark: it really differs from one to another for the PC capability, while some will wait a long time and others may have a faster opening, so please be patient)
   ```
   . /usr/share/gazebo/setup.sh
   export TURTLEBOT3_MODEL=burger
   ros2 launch turtlebot3_gazebo turtlebot3_house.launch.py
   ```

   Here are some tips for visualizing in Gazebo:

- Left-click: select entity.
- Right-click: opens menu with options:
- Left-click and drag: pan around the scene.
- Right-click and drag: zoom in and out.
- Scroll wheel forward/backward: zoom in and out.
- Scroll wheel click and drag: rotate the scene.



At the bottom of the Gazebo window, you will see the Real Time Factor, if this number is consistently below 0.5 (assuming you are not running anything else heavy on your VM or on your computer), this means that your simulation will likely be quite slow, and you may want to consider utilizing one of the alternative systems proposed (you may still use your system for code development and connecting with the real robot, just the testing in simulation might be slow).

7. (Optional) To check the latency of the topics in TurtleBot (needed when you will be using the physical robot to check the latency in the communications)

```
./latency_check.py topic msgType
# for example for scan topic
./latency_check.py /scan LaserScan
```

II. Sensor Data Processing for Mobile Robots

1. Open the Gazebo visualization by the command in the last section:

```
. /usr/share/gazebo/setup.sh
export TURTLEBOT3_MODEL=burger
ros2 launch turtlebot3_gazebo turtlebot3_house.launch.py
```

2. Open a new terminal and initialize the RViz by the following command:
   `rviz2`

3. Open a new terminal and type the following command:
   `ros2 launch slam_toolbox online_sync_launch.py`
   Then we need to import the map as well as the robotic model.
   - Click "Add" in the lower left corner and then browse "Map" under the "Topic" section, then click "ok"
   - Browse "RobotModel" under the "display type" and then click "ok". Select the urdf model of the robot under the directory "opt/ros/humble/share/turtlebot3_description/urdf/turtlebot3_burger.urdf"

4. Open a new terminal and type the following command and then could control the turtlebot through keyboard:
   `ros2 run teleop_twist_keyboard teleop_twist_keyboard`
   The use of keyboards will be displayed automatically after typing the above command and show as:

```
See the on-screen instructions:
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
anything else : stop

CTRL-C to quit
```

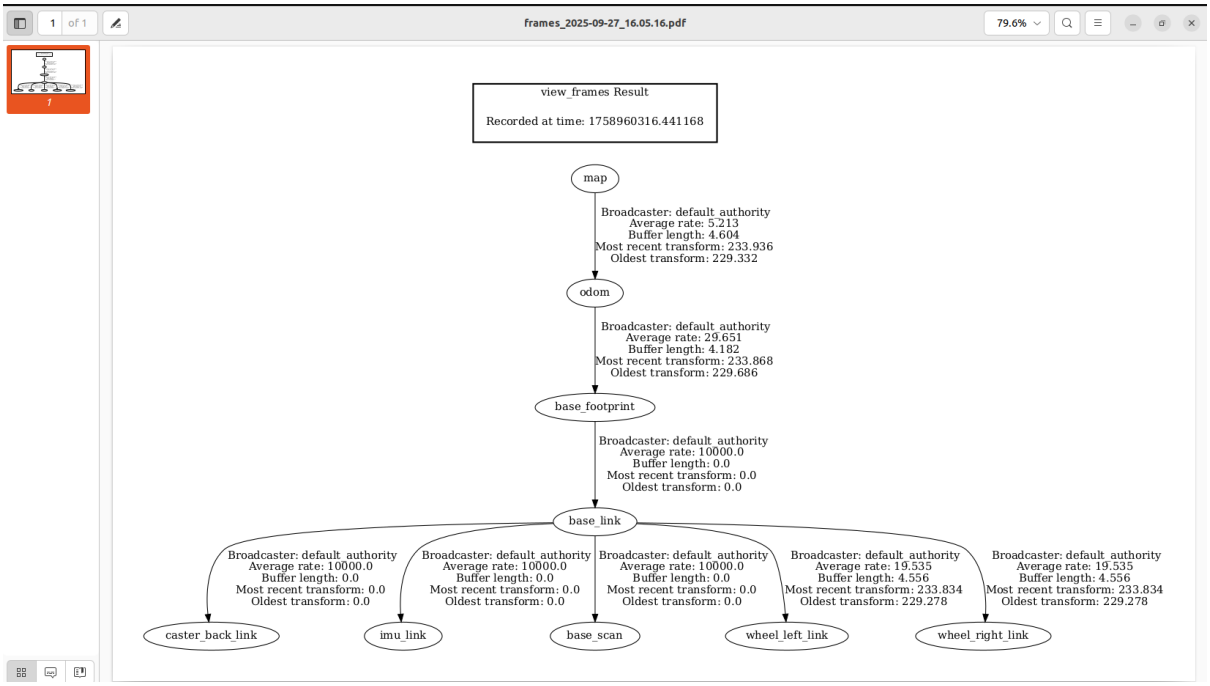
   Then you can use the keyboards to control the turtlebot and visualize the movement in both Gazebo and RViz, while you should keep the teleop terminal page not minimized, otherwise the robot will not listen from the keyboard.

5. Now we try to save the map obtained from this SLAM by the following commands:
   `ros2 run nav2_map_server map_saver_cli -f map`

6. Then we proceed to the tf information retrieval.
   - The "tf tree," short for "transform tree," is a fundamental concept in robotics and 3D spatial reasoning
   - It represents a hierarchical structure of coordinate frame transformations that describe the spatial relationships between various objects or sensors in a robotic system

- The "tf tree" helps maintain a consistent reference frame for different parts of a robot or its environment, enabling accurate data fusion and coordination
- It's crucial for tasks like localization, mapping, and navigation, ensuring that data from different sensors can be properly aligned and interpreted

The "tf tree" can be viewed for your current scene by the following command:
`ros2 run tf2_tools view_frames`
and then you are expected to have this frame image stored:



7. Now we try to edit some python programs for the defined motion planning

you will firstly complete the provided code motions.py to move the robot and collect data. For robot movement, you will be sending motion commands as velocities (twists). This means you will need to publish velocities over the /cmd_vel topic. For data collection from the IMU, the Lidar (laser scan), and the wheel encoders (odometry), you will be subscribing to /imu, /scan, and /odom, respectively.

- Find utilities.py and motions.py in the current branch (labOne) and download them.
- Open each script and follow the TODO comments to implement the requirements. You should replace each "..." in the script before you attempt running it.

To setup your code:

- Import the right types of messages needed (see motions.py); to do so, you will need to check for message type and message components given the topic names and using
  `ros2 commands`

`ros2 topic info /topic_name`
`ros2 interface show message_type`, respectively, as covered in the tutorials. For online documentation of the messages (you need to select the ROS2 distro you are using)
https://index.ros.org/p/geometry_msgs/
https://index.ros.org/p/sensor_msgs/
https://index.ros.org/p/nav_msgs

- Set up the publisher for the robot's motions;
- Create the QoS profile.

you need to implement three different motions for the robot:

- Circle;
- Spiral;
- Straight line.

Follow the comments in motions.py to implement these motions for the robot, there is one function for each of these motions.

- Fill in the imports
- Create your velocity publisher
- Create the QoS profile for your subscribers
- Create your IMU, Encoder (Odom), and Lidar (Scan) subscribers
- Create your subscriber callbacks
  - All we want to do in this lab is log the values.
  - Use the provided loggers (look inside the utilities.py to see how to use the "log_values" function). These loggers create csv files.
- Complete the motion functions

Remember that these are velocity commands "…" so how would you control the velocity of the TurtleBots to achieve those motions.
Now we move to the utilities.py:

- Complete the "log_values" function
  - Use the header writer in the init function as a hint on how you should do this
- Complete the Quaternion to Euler function
  - The Odometry messages gives you angles in quaternions, so you need to convert them to Euler angles for something easily interpretable

➔ To show the effects of motion.py, try the following commands with three cases:

```
# For circular motion
python3 motions.py --motion circle

# For spiral motion
python3 motions.py --motion spiral
```

```
# For line motion
python3 motions.py --motion line
```

Submission and Assessment:

Please prepare a written report in pdf format containing in the front page:

- Names (Family Name, First Name)
- UID

report the following:

- Screenshots of your ROS setup, program testing in Gazebo and Rviz, etc (follow the steps in this lab sheet).
- Screenshots of running the TurtleBot in house environment and the map you obtained (freely run your robot in the house)
- Your filled-in python codes (motion.py and utilities.py) with explanations.
- A brief discussion to show your understanding of the sensor information. Hint: you may leverage on the course material and the online documentation of the messages to better interpret your data.

There are no minimum page limit or format requirements, but a more professional report could contribute to a higher mark for this lab exercise.

Deadline: check the announcements in HKU Moodle.

Questions:

Any technical issues may be contacted via emails to our TA team (email addresses enclosed in lecture note).

Prepared by Peter WANG
Teaching Assistant for DASE7505
September 2025