Foundations of Robotics

Lab 1: Run in Gazebo

Overview

In this lab section, we will set up our development environment in Gazebo, bring up a Turtlebot2 robot and a ReactorX150 robot arm, and play with them. Make sure you can get them run at the end of the lab!

To better accommodate beginners, we provide step-by-step instructions in lab sections. If you are an advanced developer, please feel free to skip any part you feel confident about.

Submission

- Files to submit:
  - A few screenshots to show that you have successfully installed the required software and launched the turtlebot mobile robot and the robot arm.
- Grading rubric:
  - You are good to go as long as the screenshots can demonstrate that you have successfully installed the software and launched the robots.

General Information

Here are some general ideas that can help you get prepared for this lab section.

When working with robots, we need skills in Linux, ROS, Python/C++, Git/GitHub, and Virtual Machine (VM).

1. The best way to learn **Linux** is to spend time playing with it. It's just like the first time you had your Windows/Mac computer. Additionally, you may follow some tutorials online and try those commonly used commands in terminals. For example, it is recommended to go over chapter 1-3 of this tutorial.
2. Robot Operating System (ROS) is a Linux software library specifically designed for programming on robots. It has some features of OS, but is not actually an independent OS. With ROS, people do not need to worry about low-level communications and keep reinventing the wheel. On top of ROS, developers all over the world can work on their own software packages, and contribute to ROS

community. These packages are similar to those libraries that we “import” in Python.

3. A good way to learn **ROS** is to learn from ROS wiki, which provides official tutorials. In addition, there is a reference book *A Gentle Introduction to ROS* by Jason M. O’Kane (free online), which introduces useful design ideas behind ROS.
4. For **Python**, basically you need to have a rough idea about data structures, operators, flow control, etc. There are also many good tutorials online. For example, the tutorial on W3Schools. Going through the first 20 sections (until Python Functions) would be sufficient for this class. For **C++**, we recommend this option only if you are an advanced developer and are confident doing so.
5. **Git** is a version control tool and GitHub is a website (or company) that offers Git-based version control service. It’s good to learn Git in the sense that you can better manage your code. With Git, you can see all your change history, and have backups of each version. Many ROS packages that we are going to use in this course are hosted on GitHub. However, it’s not strictly required in this class. Going through chapter 1-5 of this tutorial might be helpful.
6. For **Virtual Machine**, there are mainly two kinds of software available online. One is VMware and the other is VirtualBox. The former has better utilization of GPU and hence supports better graphics, but it is not free of charge. The good news is that in recent years VMware has released a free “Player” version for individual users, which we will discuss later. On the other hand, VirtualBox is totally open source (free) for all platforms (Windows, Mac, Linux). However, it does not perform well in heavy simulation tasks in Gazebo. (Gazebo is a simulator that we are going to use throughout the course, together with ROS.)

Please familiarize yourself with the above concepts/tools, if they are new to you.

In the following, we will go through some basic steps to get our development environment ready.

Install Virtual Machine

- If you have a Linux laptop, or you can dual boot with Linux operating system, that’s great! This is the best way to work on robots. (Please be careful about dual boot, since you have to take potential risks. We recommend using VMware instead.)

- If you have a Windows laptop, please go for VMware Workstation Player. The free version is available for non-commercial, personal use.
- If you have a Mac laptop, please go to VMware Fusion webpage, register under "Get a Free Personal Use License" tab and download **VMware Fusion Player** using a free personal license.

Install Linux

Once you have your VMware installed, let's create a new VM and install Ubuntu 18.04.

- Download Ubuntu 18.04 disc image from official website (64-bit PC Desktop).
- In VMware, create a new VM.
    - Typical configuration
    - Choose the disc image you download
    - Enter some information about this VM
    - Again, enter name
    - Please allocate at least 30GB (preferred 50GB or more)
    - Store virtual disk as a single file
    - Customize Hardware: Please allocate more memory and CPU processors for better performance
    - Finish
- Great. Now you have a (virtual) Linux computer. Take your time and play with it!

Note that the disk size 30GB/50GB will not be allocated instantly, but will grow gradually as you add more stuff.

Install ROS

Please install ROS following the official installation tutorial.

Set up ROS Workspace

From now on, we assume that you have already installed Ubuntu 18.04 and ROS Melodic.

- Please open a new terminal and create a new ROS workspace by the following commands.

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin_make
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/catkin_ws/devel/setup.bash
```

- Take a look at `catkin_ws` directory and see what happens. You can use `ls` command to see the files in this directory, or use `ls -a` to see all files including hidden files. Alternatively, you can open File Explorer and navigate to this folder, and use `Ctrl + H` to see hidden files. The file `.catkin_workspace` was created to tell the system that this current directory is a ROS workspace.
- Next let's create a new ROS package.

```
cd ~/catkin_ws/src
catkin_create_pkg fall2022 std_msgs rospy roscpp
```

- Take a look at your new package `fall2022` and see what happens. You should be able to see a `package.xml` file and a `CMakeLists.txt` file. Open them and take a quick look. You may use Google to help you build up a high-level understanding.
- After creating a new package, we can go back to our workspace and build this package. This is to tell ROS that "Hey, we have a new package here. Please register it into the system."

```
cd ~/catkin_ws
catkin_make
```

- Now the system knows this ROS package, so that you can have access to it anywhere. Try navigating to different directories first, and then go back to this ROS package by `roscd` command. See what happens when running the following commands.

```
cd
roscd fall2022

cd ~/catkin_ws
roscd fall2022

cd ~/Documents
roscd fall2022
```

- Congratulations. You have completed the basic ROS tutorials.

Set up Turtlebot in Gazebo

- Please install turtlebot2 packages using the script available at https://github.com/UCR-Robotics/Turtlebot2-On-Melodic
- Navigate to your `fall2022` package and create a new folder and a new launch file.

```
roscd fall2022
mkdir launch
cd launch
touch gazebo.launch
gedit gazebo.launch
```

- Please copy and paste the following script, then save it. (This script is also available in our provided zip folder.)

```
<launch>
  <arg name="world_file" default="worlds/empty.world"/>

  <arg name="urdf" default="$(find
turtlebot_description)/robots/kobuki_hexagons_astra.urdf.xacro" />
  <param name="robot_description" command="$(find xacro)/xacro --inorder $(arg urdf)" />

  <!-- include two nodes gazebo (server) and gazebo_gui (client) -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(arg world_file)"/>
  </include>

  <!-- Gazebo model spawner -->
  <node name="spawn_turtlebot_model" pkg="gazebo_ros" type="spawn_model"
        args="$(optenv ROBOT_INITIAL_POSE) -unpause -urdf -param robot_description
-model mobile_base"/>

  <!-- Velocity muxer -->
  <node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
  <node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
        args="load yocs_cmd_vel_mux/CmdVelMuxNodelet mobile_base_nodelet_manager">
    <param name="yaml_cfg_file" value="$(find turtlebot_bringup)/param/mux.yaml"/>
    <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
  </node>

  <!-- Publish robot state -->
  <node pkg="robot_state_publisher" type="robot_state_publisher"
name="robot_state_publisher">
    <param name="publish_frequency" type="double" value="30.0" />
  </node>
</launch>
```

Play with Turtlebot in Gazebo

- Launch Gazebo simulator and spawn a new robot by the following command. It may take a while at the first time you open Gazebo, since it will need to download some models and world environments.

```
roslaunch fall2022 gazebo.launch
```

- Note: If you experienced graphic issues in Gazebo, please run the following command for once. Then close all terminals and try again.

```
echo "export SVGA_VGPU10=0" >> ~/.bashrc
```

If the issue persists, please shutdown your VM, go to VM settings and allocate more resources (Processor Cores, Memory, Graphics Memory). If the issue still persists, please disable “3D Acceleration” in Display settings.

- Once the robot is successfully spawned in Gazebo, we can open a new terminal and launch the teleop node.

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

- Keep the teleop terminal open (selected) and you should be able to control the robot using keyboard now. The teleop program in this terminal takes in whatever keys you entered and converts them into velocity commands to send to the robot. Now spend some time playing with it! (Don’t send the keyboard teleop commands to the Gazebo window, it won’t work; send commands to the terminal)
- You can also put some obstacles (objects) in Gazebo simulation environment, and drive the robot to collide with obstacles. See what happens :)
- Note: To terminate the programs running in the terminal, please use `Ctrl + C` and wait for a moment (it can take up to 10s for Gazebo to shut down). If the terminal is closed without terminating the programs properly (meaning that the programs are still running in the backend), you will see a Gazebo crash error in the next run.

Set up robot arm in Gazebo

- First let’s download the ROS packages for the robot arm.

```
cd ~/catkin_ws/src
git clone https://github.com/UCR-Robotics/interbotix_ros_arms.git
```

- We can install the dependencies by the following commands.

```
cd ~/catkin_ws
rosdep update
rosdep install --from-paths src --ignore-src -r -y
```

- We need to add one more package that is not currently supported by `rosdep` install. (BTW, this *modern_robotics* library is developed by the authors of our textbook *Modern Robotics*. It contains the Python implementation of some common operations. We will learn them in lectures as well.)

```
sudo apt install python-pip
sudo pip install modern_robotics
```

- Lastly, with all dependencies ready, we can build the ROS package by the following commands.

```
cd ~/catkin_ws
catkin_make
```

Play with robot arm in Gazebo

- Launch the ReactorX 150 robot arm in Gazebo by the following command.

```
roslaunch interbotix_moveit interbotix_moveit.launch robot_name:=rx150 use_gazebo:=true
```

- You will see the robot arm is ready in Gazebo but the RViz (the visualization software used in ROS) is still pending. This is because it is still waiting for Gazebo to start simulation. In the bottom left of Gazebo window, you will see a small **Play ▶ button**. Click it to let it run!
- Once Gazebo starts simulation, the RViz will prompt you two panels on the left and a visualization of the robot arm on the right. On the top left panel, go to "MotionPlanning" -> "Planning Request" -> "Query Goal State" and check this box. Then you can drag the "ball" on the tip of the robot arm to wherever you want it to go.
- Once a goal pose is set, in the bottom left panel, go to "Planning" tab and try buttons "Plan", "Execute", or "Plan and Execute". Cool! The software can figure out a path for the arm to follow and reach the exact goal pose you just set. Spend some time playing with it!
- You can also take a look at Gazebo to see the current status of the robot arm. RViz provides a tool for better interaction, but only Gazebo shows the real physical status.
- Have fun!!