SWROB2
Simulation guide

Submitted by: Nicolai, Johannes, Anders, Daniel

Lecture: Mirgita Frasheri,
Andryi Sarabakha,
& Søren M. Dath

Århus Universitet

March 20, 2024

Contents

1 Purpose

This document serves as a comprehensive guide aimed at facilitating the setup of a simulation environment for the TurtleBot 3. It is intended for students, researchers, and robotics enthusiasts who wish to explore the capabilities of TurtleBot 3 within a simulated environment before implementing solutions in real-world scenarios. By following the steps outlined in this guide, readers will be able to install and configure the necessary software components, including ROS Noetic, Gazebo, and the TurtleBot 3 packages, on a Ubuntu 20.04 LTS system.

The simulation environment provides a safe and cost-effective platform for testing algorithms, developing robotics applications, and conducting experiments without the need for physical hardware. It offers an accessible entry point into the world of robotics, emphasizing hands-on learning and experimentation. Through this guide, we aim to empower users with the knowledge to successfully establish a robust simulation setup, enabling them to simulate a variety of tasks and scenarios with the TurtleBot 3.

Additionally, this guide underscores the importance of understanding the underlying principles of robot operation, ROS architecture, and simulation dynamics. By the end of this setup, users will be well-equipped to navigate the ROS ecosystem, leverage the TurtleBot 3's features, and extend their exploration to more advanced robotics concepts and applications.

2 System Requirements and Compatibility

The choice of operating system is crucial for a seamless installation and operation of ROS (Robot Operating System). ROS Noetic Ninjemys, being the thirteenth release of ROS, officially supports Ubuntu 20.04 LTS (Focal Fossa). This compatibility is designed to leverage the long-term support and stability offered by Ubuntu 20.04 LTS, ensuring that developers have access to consistent updates and security patches.

It is highly recommended to use Ubuntu 20.04 (Focal Fossa) LTS as the primary operating system for installing ROS Noetic to avoid potential compatibility issues that may arise with other versions of Ubuntu or different Linux distributions. Ubuntu 20.04 LTS provides an optimal foundation for running ROS Noetic, thanks to its updated libraries, improved security features, and enhanced performance.

While ROS Noetic might be installed on newer versions of Ubuntu or through the use of Docker containers on other operating systems, such setups may require additional configuration steps and might not provide the same level of stability and support as the recommended Ubuntu 20.04 LTS platform. Users opting for alternative methods should be prepared for a more complex installation process and potential troubleshooting.

Ensuring the alignment between the operating system and ROS version is essential for developers and researchers aiming to utilize the full spectrum of ROS features and capabilities without encountering unnecessary obstacles. By adhering to the recommended system requirements, users can expect a smoother installation experience and more reliable system performance during their robotics development and simulation endeavors.

3 VMware Installation

1. Visit the official VMware website and download either VMware Workstation Pro or VMware Player.
2. Follow the on-screen instructions to install VMware on your host machine.

4 Ubuntu 20.04 LTS Installation in VMware

1. Launch VMware and select "Create a New Virtual Machine".
2. Choose "Installer disc image file (iso)" and browse to select your downloaded Ubuntu 20.04 LTS ISO file.
3. Complete the guided setup, allocating desired resources (disk space, memory, CPU).
4. Proceed with the Ubuntu installation process within the VM, creating a user account when prompted.

5 ROS Noetic Installation

Follow this tutorial or visit This Link.
Open a terminal in Ubuntu (no specific directory required).

1. Add the ROS Noetic repository:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Install curl and add the ROS key:

```
sudo apt install curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc |
sudo apt-key add -
```

3. Update apt sources and install ROS Noetic Full Desktop:

```
sudo apt update
sudo apt install ros-noetic-desktop-full
```

4. Add ROS environment setup to your bash session:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

5.1 Environment Setup

For optimal use of ROS, it's crucial to source the ROS environment setup script in every bash terminal session used for ROS development. This ensures that your terminal is aware of ROS and its configurations. To automatically source the setup script for ROS Noetic in all new shell sessions, add the following to your `.bashrc` or `.zshrc` file, depending on your shell:

For Bash:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

For Zsh:

```
echo "source /opt/ros/noetic/setup.zsh" >> ~/.zshrc
source ~/.zshrc
```

This step is particularly important if you have more than one ROS distribution installed, as it ensures that the correct version's environment is sourced.

5.2 Dependencies for Building Packages

To manage your own ROS workspaces and compile ROS packages, additional tools and dependencies are required. These tools facilitate the creation, management, and building of ROS packages and workspaces. To install these dependencies, execute the following command in your terminal:

```
sudo apt install python3-rosdep python3-rosinstall
python3-rosinstall-generator python3-wstool build-essential
```

5.3 Initialize rosdep

The `rosdep` tool helps in installing system dependencies for the software that you want to compile. It is also required to run some core components in ROS. If `rosdep` has not been installed yet, it can be installed and initialized as follows:

```
sudo apt install python3-rosdep
sudo rosdep init
rosdep update
```

Initializing `rosdep` is a critical step for ensuring that your ROS packages can resolve their system dependencies efficiently.

6 Installing ROS Dependencies

1. Initialize rosdep:

   ```
   sudo rosdep init
   rosdep update
   ```

7 TurtleBot3 and Gazebo Installation

To utilize TurtleBot3 robots within the Gazebo simulation environment, it's necessary to install the TurtleBot3 and Gazebo packages first.

1. Install TurtleBot3 and Gazebo packages by executing:

```
sudo apt install ros-noetic-turtlebot3 ros-noetic-turtlebot3-simulations
```

Additionally, for the latest features or custom modifications, TurtleBot3 and related simulation packages can be installed from source:

2. Navigate to the `src` directory of your Catkin workspace:

```
cd ~/catkin_ws/src
```

3. Clone the TurtleBot3 message, TurtleBot3, and TurtleBot3 simulations repositories:

```
git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
```

4. Return to your Catkin workspace root and compile the workspace:

```
cd ~/catkin_ws
catkin_make
```

5. Source the workspace to make the ROS packages available in your environment:

```
source ~/catkin_ws/devel/setup.bash
```

7.1 Configuring the TurtleBot3 Model Environment Variable Permanently

Setting the TurtleBot3 model environment variable permanently streamlines the process of working with TurtleBot3 simulations, ensuring the specified model is automatically recognized in new terminal sessions.

1. Open the terminal and edit the `.bashrc` file using nano:

```
nano ~/.bashrc
```

2. Scroll to the bottom and add the line to set the TurtleBot3 model environment variable. For example, to select the `burger` model:

```
export TURTLEBOT3_MODEL=burger
```

Replace `burger` with `waffle` or `waffle_pi` as needed.

3. Save the changes (`Ctrl+O`, `Enter`) and exit nano (`Ctrl+X`). Source the `.bashrc` file to apply the changes:

```
source ~/.bashrc
```

7.2 Testing the Installation with TurtleBot3 in Gazebo

Verify the installation and configuration by launching a TurtleBot3 simulation in Gazebo:

1. Execute:

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

8 Oneline install of ROS Neotic

For the oneline install of ROS Neotic then insert one by one in ther terminal:

```
sudo apt update

sudo apt upgrade

wget  https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/
install_ros_noetic.sh

chmod 755 ./install_ros_noetic.sh

bash ./install_ros_noetic.sh
```

9 Install Dependent ROS Packages

Important Dependent ROS Packages. Notice it's one long string that needs to be copied into terminal:

```
sudo apt-get install ros-noetic-joy ros-noetic-teleop-twist-joy \
ros-noetic-teleop-twist-keyboard ros-noetic-laser-proc \
ros-noetic-rgbd-launch ros-noetic-rosserial-arduino \
ros-noetic-rosserial-python ros-noetic-rosserial-client \
ros-noetic-rosserial-msgs ros-noetic-amcl ros-noetic-map-server \
ros-noetic-move-base ros-noetic-urdf ros-noetic-xacro \
ros-noetic-compressed-image-transport ros-noetic-rqt* ros-noetic-rviz \
ros-noetic-gmapping ros-noetic-navigation ros-noetic-interactive-markers
```

10 Managing, Creating, and Launching a New ROS Project

Efficiently managing multiple projects, creating new ones, and launching them are crucial skills for any roboticist working with ROS (Robot Operating System). This section provides a concise guide on these tasks within a Catkin workspace.

10.1 Managing Multiple Projects

A Catkin workspace offers a versatile environment for simultaneously developing multiple ROS projects. Each project, structured as a package, resides in the `src` directory of the workspace.

Best Practices:

- Organize related projects into separate packages within the `src` directory.
- Utilize version control systems like Git within each project package to track changes and collaborate.

10.2 Creating a New ROS Project

Creating a new project involves generating a new ROS package within an existing Catkin workspace, containing all necessary files for development.

Steps to Create a New Package:

1. Navigate to the `src` directory of your Catkin workspace:

   ```
   cd ~/catkin_ws/src
   ```

2. Use `catkin_create_pkg` to create a new package with necessary dependencies:

   ```
   catkin_create_pkg my_ros_project rospy std_msgs
   ```

3. Develop your ROS nodes or scripts within the newly created package directory.

10.3 Building Your ROS Project

Before launching your project, it's crucial to build your Catkin workspace to compile the new package and any changes made.

Steps to Build Your Workspace:

1. Navigate to the root of your Catkin workspace:

   ```
   cd ~/catkin_ws
   ```

2. Build the workspace using `catkin_make`:

```
catkin_make
```

3. Source the workspace setup file to update your environment:

```
source devel/setup.bash
```

10.4 Launching a Newly Created Project

Testing your project's functionality involves running ROS nodes or launch files you've developed.

Launching a ROS Node:

1. Open a new terminal and source your Catkin workspace:

```
source ~/catkin_ws/devel/setup.bash
```

2. Launch a node from your package:

```
rosrun my_ros_project my_node
```

Using a ROS Launch File:

1. Create a launch file within the `launch` directory of your package if not already present.
2. Launch the project using `roslaunch`:

```
roslaunch my_ros_project launch_file.launch
```

Note: Always ensure the ROS master is running (using `roscore`) before launching any ROS nodes or applications.

By following these guidelines, developers can effectively manage and scale their ROS projects within a single Catkin workspace, streamlining the development and testing of complex robotic systems.

11 ROS Network Configuration Guide

11.1 Introduction

Before diving into multi-machine communication with ROS, it's crucial to set up your network configurations correctly. This section guides you through configuring the `ROS_MASTER_URI` and `ROS_IP` environment variables, ensuring your ROS nodes can find and communicate with each other across a network.

11.2 Prerequisites

- Ensure ROS Noetic is installed on all machines involved.
- Determine the IP addresses of the machines. You can use the `hostname -I` or `ifconfig` command on Linux to find out your IP address.

11.3 Configuration Steps

11.3.1 Identify the ROS Master Machine

Decide which machine will run the ROS Master. This is typically the machine that initiates the ROS network.

11.3.2 Configure the `ROS_MASTER_URI`

On **every** machine that will participate in the ROS network, open the `~/.bashrc` file in a text editor. For example, you can use `nano ~/.bashrc`.

1. Add the following line at the end of the file, replacing `YOUR_MASTER_IP` with the IP address of the machine running the ROS Master:

   ```
   export ROS_MASTER_URI=http://YOUR_MASTER_IP:11311
   ```

2. Save and close the file.

11.3.3 Set the `ROS_IP`

Still in the `~/.bashrc` file of **each** machine, add another line to specify the `ROS_IP`, replacing `YOUR_MACHINE_IP` with the machine's IP address:

1. ```
   export ROS_IP=YOUR_MACHINE_IP
   ```

2. Save and close the file.

If you are having trouble to connect even though you are using the right IP then you can try `0.0.0.0`.

11.3.4 Apply the Configuration

To apply the changes, source the `~/.bashrc` file by running:

```
source ~/.bashrc
```

Do this on each machine.
```

11.3.5 Verify the Configuration

Verify that the environment variables are set correctly by running:

```
echo $ROS_MASTER_URI
echo $ROS_IP
```

Ensure that the output matches the IP addresses you configured.

11.4 Troubleshooting Tips

- If ROS nodes cannot communicate, double-check the IP addresses and ensure there are no typos.
- Ensure there's no firewall blocking the communication on port `11311`.
- Use the `ping` command to check network connectivity between machines.

11.5 Conclusion

By following these steps, you've configured your machines for ROS networking, allowing ROS nodes on different machines to communicate seamlessly. This setup is essential for distributed ROS applications and multi-robot systems.

12 Installation and Configuration of roscpp with ROS Noetic, Gazebo, and TurtleBot Simulation

12.1 Introduction

This section outlines the steps to set up and configure ‘roscpp‘ with ROS Noetic, Gazebo, and the TurtleBot simulation. By integrating ‘roscpp‘ with these tools, developers can create a dynamic environment for robot programming and simulation, leveraging the capabilities of C++ with ROS.

12.2 Prerequisites

- Installation of ROS Noetic on your system.
- Basic knowledge of ROS packages and the catkin build system.
- Gazebo and TurtleBot simulation packages compatibility with ROS Noetic.

12.3 Configuration Steps

12.3.1 Install roscpp

Ensure ‘roscpp‘ is installed by executing:

```
sudo apt-get install ros-noetic-roscpp
```

12.3.2 Install Gazebo for ROS Noetic

To integrate Gazebo with ROS Noetic, run:

```
sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-ros-control
```

12.3.3 Install TurtleBot3 Simulation Packages

Install TurtleBot3 and its simulation packages with:

```
sudo apt-get install ros-noetic-turtlebot3 ros-noetic-turtlebot3-simulations
```

12.3.4 Running a TurtleBot3 Simulation

Test the installation by launching a TurtleBot3 simulation in Gazebo:

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

12.4 Troubleshooting Tips

- Ensure all installations are completed without errors.
- Verify the ROS environment is correctly sourced before running the simulation.
- Check the compatibility of the TurtleBot3 model with the Gazebo version.

12.5 Conclusion

Following these steps, you will have successfully configured ‘roscpp‘ with ROS Noetic, Gazebo, and TurtleBot simulation. This setup offers a comprehensive platform for developing and testing robot applications, providing invaluable insights into robotics programming and simulation.

13 Setting Up ROS Workspace

First, we need to create a new ROS workspace, which is a directory where your ROS projects will reside.

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
```

Initialize the workspace with `catkin_make`:

```
catkin_make
```

Source your new workspace so ROS can find the packages you develop:

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

14 Create a ROS Package

Now that your workspace is set up, let's create a new ROS package using roscpp.

Navigate to the `src` directory in your workspace:

```
cd ~/catkin_ws/src
```

Create a new package. We'll name it `my_roscpp_package` and include dependencies for roscpp and std_msgs:

```
catkin_create_pkg my_roscpp_package roscpp std_msgs
```

Build your workspace to ensure your new package is set up correctly:

```
cd ~/catkin_ws
catkin_make
```

15 Installing Gazebo and Rviz Plugin

15.1 Installing the Camera Plugin

After creating backups of the necessary files, proceed with the installation of the camera plugin by modifying the TurtleBot3's URDF/Xacro files to include the camera specifications and the associated Gazebo plugin.

15.1.1 Add the Camera Model to the URDF/Xacro File

To integrate the camera into your TurtleBot3 model, you need to edit the `turtlebot3_[model].urdf.xacro` file, which describes your robot's physical structure. Here, you will reference the camera's Xacro file that you have created or will create.

First, if not already created, make a new Xacro file named `turtlebot3_camera.xacro` within the `turtlebot3_description/urdf` directory with the camera specifications. This file should define the camera as a separate component of your robot, specifying its size, position, and orientation relative to the robot.

turtlebot3_camera.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
<xacro:macro name="turtlebot3_camera" params="parent_link">
<!-- Definer kameraets link -->
<link name="camera_link">
<inertial>
<mass value="0.001"/>
<origin xyz="0 0 0" rpy="0 0 0"/>
<inertia ixx="1e-6" iyy="1e-6" izz="1e-6" ixy="0" ixz="0" iyz="0"/>
</inertial>
<visual>
<origin xyz="0 0 0" rpy="0 0 0"/>
<geometry>
<box size="0.05 0.05 0.05"/>
</geometry>
<material name="black"/>
</visual>
</link>

<!-- Fastgør kameraets link til robotten med et fast joint -->
<joint name="camera_joint" type="fixed">
<origin xyz="0.1 0 0.2" rpy="0 0 0"/>
<parent link="${parent_link}"/>
<child link="camera_link"/>
</joint>

<!-- Gazebo plugin for at simulere kameraet og publicere billeder til ROS -->
<gazebo reference="camera_link">
```

```
<sensor type="camera" name="turtlebot3_camera_sensor">
<update_rate>30.0</update_rate>
<camera>
<horizontal_fov>1.3962634</horizontal_fov>
<image>
<width>640</width>
<height>480</height>
<format>R8G8B8</format>
</image>
<clip>
<near>0.02</near>
<far>300</far>
</clip>
</camera>
<plugin name="camera_controller" filename="libgazebo_ros_camera.so">
<cameraName>camera</cameraName>
<imageTopicName>image_raw</imageTopicName>
<cameraInfoTopicName>camera_info</cameraInfoTopicName>
<frameName>camera_link</frameName>
<hackBaseline>0.07</hackBaseline>
</plugin>
</sensor>
</gazebo>
</xacro:macro>
</robot>
```

Then, in your robot model's URDF/Xacro file (e.g., `turtlebot3_burger.urdf.xacro`), include this camera Xacro file at the top with the following line:

```
<xacro:include filename="$(find turtlebot3_description)/urdf/turtlebot3_camera.xacro
```

And instantiate the camera component within the robot model where appropriate, typically near the end of the file before the closing `</robot>` tag, like so:

```
<xacro:turtlebot3_camera parent_link="base_link" />
```

This tells the URDF processor to include your camera component as part of the TurtleBot3 model, attaching it to the 'base_link' or another specified parent link.

15.1.2 Integrate the Camera Plugin in Gazebo Configuration

Next, you'll modify the `turtlebot3_[model].gazebo.xacro` file to include the Gazebo plugin for the camera, enabling simulation and ROS interaction. Add the detailed Gazebo plugin configuration as shown previously. This code snippet specifies how the camera is simulated within Gazebo, including its field of view, image resolution, and the topics over which image data is published.

Remember to replace the placeholder '[model]' with your specific TurtleBot3 model, such as 'burger' or 'waffle'.

15.2 Testing the Camera Integration

After integrating the camera into your TurtleBot3 model and its Gazebo simulation environment, it's crucial to test and ensure everything is working as expected. Follow these steps to compile your ROS workspace, launch the Gazebo simulation, and visualize the camera feed:

```
cd ~/catkin_ws
catkin_make
source devel/setup.bash
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Then, use ROS tools such as `rqt_image_view` to visualize the camera feed and confirm the camera has been successfully integrated:

```
rosrun rqt_image_view rqt_image_view
```

Select the appropriate topic (e.g., '/camera/image_raw') to view the live feed from the TurtleBot3 camera.

This section provides a comprehensive guide to integrating a camera into the TurtleBot3 robot for use within the Gazebo simulation environment, covering file modification, plugin integration, and testing procedures. It's designed to be accessible even to those new to working with ROS and Gazebo.

16 Write Your First ROS Node in C++

Let's create a simple "Hello World" node.

Create a cpp file in your package's `src` directory. Name it `hello_world.cpp`:

```
cd ~/catkin_ws/src/my_roscpp_package/src
touch hello_world.cpp
```

Edit `hello_world.cpp`. Use a text editor *(nano)* to write the following code:

```
#include <ros/ros.h>

int main(int argc, char **argv) {
 ros::init(argc, argv, "hello_world_node");
 ros::NodeHandle nh;

 ROS_INFO("Hello, World from ROS!");

 ros::spin();
 return 0;
}
```

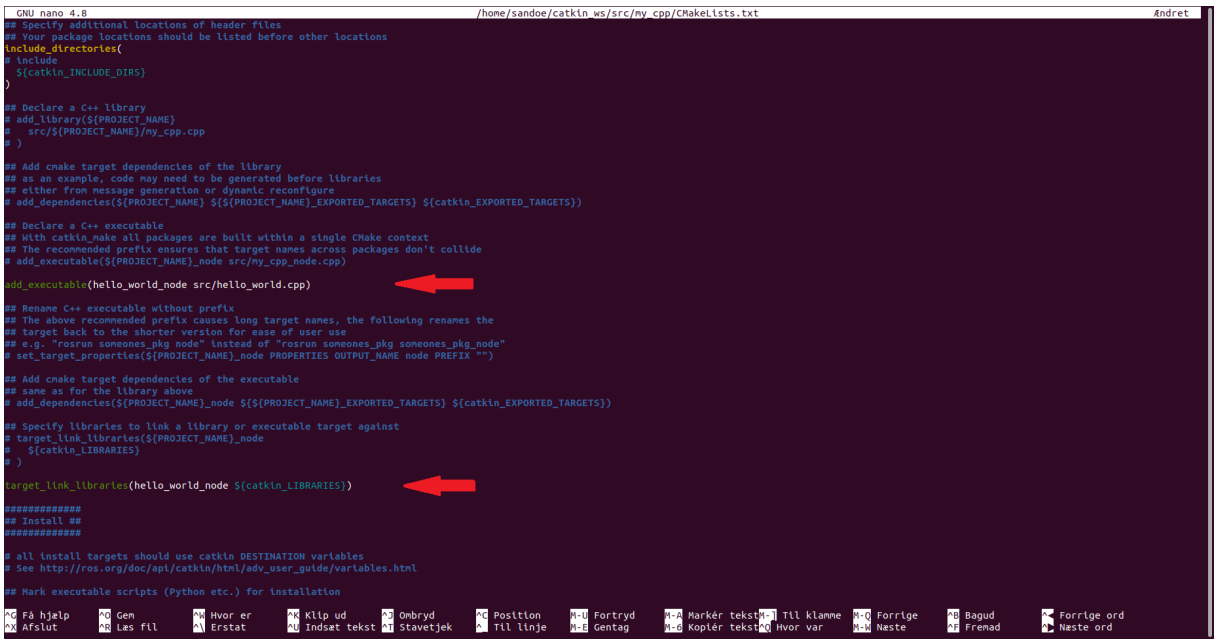
Go to the folder by typing

```
cd ~/catkin_ws/src/my_roscpp_package/
```

Update `CMakeLists.txt` to include your node. Open `CMakeLists.txt` in your package's root directory, and add the following:

```
add_executable(hello_world_node src/hello_world.cpp)
target_link_libraries(hello_world_node ${catkin_LIBRARIES})
```

This tells CMake that you want to build `hello_world.cpp` as an executable file (`hello_world_node`). It needs to be looking like this inside `CMakeLists.txt`:



Build your package again from the workspace root:

```
cd ~/catkin_ws
catkin_make
```

17 Run Your ROS Node

Make sure ROS master is running by opening a new terminal and typing:

```
roscore
```

In another terminal, run your node:

```
rosrun my_roscpp_package hello_world_node
```

You should now see a "Hello, World from ROS!" message in the terminal.

18 Source Your Environment (if necessary)

If you open a new terminal and ROS cannot find your package or node, you may need to source your workspace's setup script again:

```
source ~/catkin_ws/devel/setup.bash
```

This is the basic process for creating and running a simple ROS node in C++ using roscpp. From here, you can start expanding your node with more functionality, subscribing to topics, publishing messages, and much more.

19 Installing MATLAB on Ubuntu with Robotics Toolbox

19.1 Introduction

This guide will walk you through the process of installing MATLAB on Ubuntu and ensuring that the Robotics Toolbox is included as part of your installation.

19.2 Step 1: Download MATLAB

First, you need to download MATLAB from the official MathWorks website.

1. Visit `https://www.mathworks.com/products/matlab.html` and log in with your MathWorks account. If you do not already have an account, you will need to create one.
2. Once logged in, navigate to the downloads section and select the Linux platform for download.
3. Download the installer file to your Ubuntu machine.

19.3 Step 2: Prepare the Installation

After downloading, make the installer executable and run it.

```
$ cd ~/Downloads
$ chmod +x matlab_R20XXx_glnxa64.sh
$ sudo ./matlab_R20XXx_glnxa64.sh
```

Replace `R20XXx` with the actual version of MATLAB you have downloaded.

19.4 Step 3: Complete the Installation

Follow the on-screen installation guide.

1. When prompted, log in with your MathWorks account.
2. Accept the license agreement.
3. Select the installation folder (default is usually fine).
4. When you reach the step to choose the products you want to install, make sure to **check Robotics Toolbox**.
5. Continue through the installation wizard and finish the installation.

19.5 Step 4: Activate MATLAB

After installation, you need to activate MATLAB with your license. This is usually done at the first startup of MATLAB.

19.6 Conclusion

After these steps, MATLAB should be installed on your Ubuntu machine with the Robotics Toolbox ready for use. For further support, visit the official MathWorks support page.