Appendix B

Maintenance Manual

This manual explains the technical details of the software package. The tasks performed by the automatic installer are described so that they can be performed by hand, and information about how to extend the software package is also given.

B.1 Technical Overview

The software package provided as part of this project consists of a *Catkin workspace* called *curiosity_mars_rover_ws*. Catkin workspaces are used for building and installing ROS packages using the build tool CMake. They typically consist of three main folders:

- *build/* - where CMake runs so it can build ROS packages
- *devel/* - where built ROS packages are automatically placed for testing
- *src/* - where source code for ROS packages is located

The *curiosity_mars_rover_ws* software package only comes with a *src/* folder, since all of the other files are automatically generated by building the contents of *src/*. Within this folder, there are five ROS packages for the Curiosity rover simulation. A list of the files in these packages is provided in Appendix D. An additional three packages are included in the *dependencies* folder. The three dependencies included are:

- *ackermann_drive_controller* - for providing rover wheel control mechanisms
- *hugin_panorama* - for automatically stitching panoramas
- *tf2_web_republisher* - to transmit robot joint positions over a web server

These dependencies are included in the software package since they cannot be installed in the same fashion as most ROS packages (via the APT package management tool). All of these packages are open-source, and software licences for each have been retained in their file structure.

B.2 Detailed Requirements

The software package was built with the ROS version Noetic. ROS Noetic mainly targets Ubuntu 20.04, so this version of Ubuntu was used throughout development. There is no guarantee that the simulation will function on other versions of Ubuntu or distributions of Linux.

Version 11.10.2 of Gazebo was used during development. Gazebo does not have specific system requirements, however a dedicated graphics processor is highly recommended on their website for suitable simulation speeds.

At least 4.8 GB of free disk space is required to install and run the simulation. This quantity was determined by comparing the disk usage on a fresh Ubuntu 20.04 installation before and after installing the Curiosity simulation.

ROS Noetic also depends on Python 3, and therefore it is required for running the simulation. Ubuntu 20.04 comes preloaded with Python 3. A full list of the software that is necessary to install before running the simulation is included in Section B.3. Since the software must be downloaded from the Internet, a stable internet connection is required.

B.3 Detailed Installation

The installation script provided is the easiest way to install all of the dependencies and automatically build the Curiosity simulation software. By default, it builds the contents of a Catkin workspace located at the */home/username/curiosity_mars_rover_ws* folder (where *username* is your Linux username). However, this can be overridden. If you have an existing Catkin workspace with a different name, you can copy the ROS packages to your workspace and run the same installation script with an additional parameter to specify what folder to use. For example, if your Catkin workspace is called *catkin_ws*, you can run the script like so:

```
bash install_everything.sh catkin_ws
```

This will install all of the dependencies of the Curiosity ROS simulation and set up your system for use with the MarsViz visualisation tool, with *catkin_ws* as the workspace directory for ROS instead of *curiosity_mars_rover_ws*.

In some cases, it may be preferable to perform the installation steps manually. The remainder of this section will walk through the process of installing everything in a similar fashion to the *install_everything.sh* script. The steps are as follows:

1. Begin by adding the ROS package repository to your APT sources.

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

2. You will need a public key to use the ROS packages. It can be downloaded with *curl*.

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

3. Update your package index.

   ```
   sudo apt update
   ```

4. Install the full ROS Noetic package (roughly 3.7 GB).

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

5. Install the Curiosity software package dependencies (roughly 400 MB).

   ```
   sudo apt install ros-noetic-ros-control ros-noetic-ros-controllers ros-noetic-amcl
       ros-noetic-gmapping ros-noetic-map-server ros-noetic-move-base ros-noetic-
       rtabmap ros-noetic-rtabmap-ros ros-noetic-ira-laser-tools ros-noetic-pointcloud-
       to-laserscan ros-noetic-rosbridge-server ros-noetic-teleop-twist-keyboard hugin-
       tools imagemagick-6.q16 enblend
   ```

6. Modify the permissions of the scripts in the software package so that they are executable. (You can replace *curiosity_mars_rover_ws* with your Catkin workspace folder).

   ```
   chmod +x ~/curiosity_mars_rover_ws/src/curiosity_mars_rover_control/scripts/*
   chmod +x ~/curiosity_mars_rover_ws/src/curiosity_mars_rover_navigation/scripts/*
   chmod +x ~/curiosity_mars_rover_ws/src/curiosity_mars_rover_viz/scripts/*
   ```

7. Before you can build the Curiosity packages and use ROS launch commands, you need to *source* the ROS installation.

   ```
   source /opt/ros/noetic/setup.bash
   ```

   Instead of typing this into every terminal window, it's much more convenient to automatically run the command every time a new terminal is opened. You can add the *source* command to your *.bashrc* file to achieve this.

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

8. With the ROS installation sourced, you will now be able to build the Curiosity packages.

   ```
   cd ~/curiosity_mars_rover_ws/
   catkin_make
   ```

9. Once it's finished, you also have to source the Catkin workspace's built files.

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

   You can add another entry to *.bashrc* for automating this as well.

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

10. You can now test whether the simulation has installed correctly.

    ```
    roslaunch curiosity_mars_rover_gazebo main_mars_terrain.launch
    ```

Some additional setup is also performed by the installation script so that the MarsViz web application can work properly. This is due to the fact that it requires a security certificate in order to run. Example certificates are provided in the visualisation package, but they will eventually expire, so you may want to create your own certificate. The steps for setting up a security certificate for MarsViz are as follows:

1. Generate a new security certificate. This will produce two files.

   ```
   openssl req -x509 -newkey rsa:4096 -keyout server1.example.com.key -out server1.
       example.com.pem -days 365 -nodes
   ```

2. Create the system directory for storing certificates.

   ```
   sudo mkdir /etc/ssl/certs/localcerts/ -p
   ```

3. Copy both files to this new directory.

   ```
   sudo mv server1.example.com.key /etc/ssl/certs/localcerts/server1.example.com.key
   sudo mv server1.example.com.pem /etc/ssl/certs/localcerts/server1.example.com.pem
   ```

4. Make sure to update the permissions of these files.

```
sudo chmod 777 /etc/ssl/certs/localcerts/
sudo chmod 777 /etc/ssl/certs/localcerts/server1.example.com.key
sudo chmod 777 /etc/ssl/certs/localcerts/server1.example.com.pem
```

5. Finally, open up the local ports 8080 and 9090 so that ROS and the web application can communicate.

```
sudo ufw allow 9090
sudo ufw allow 8080
```

After performing these steps and starting MarsViz using one of the ROS launch files, you will be able to visit the web application in your browser. If your certificate files have different names to the ones in the instructions, you will need to make a small change to the source code. Navigate to the *scripts/run_server.py* file in the visualisation package (*curiosity_mars_rover_viz*) and edit the *ssl_key_file* and *ssl_certificate_file* variables to match your filenames.

B.4 Running Tests

The unit tests included in the navigation package use the Python library *unittest*, and can therefore be accessed from a number of command-line tools or IDEs. In this project, the Visual Studio Code editor was used. The following steps can be taken to replicate the testing process:

1. In a terminal, start a simulation, for example:

```
roslaunch curiosity_mars_rover_gazebo main_simple.launch
```

2. In a second terminal, start the navigation system:

```
roslaunch curiosity_mars_rover_navigation navigation.launch
```

3. Open the navigation package source code in the Visual Studio Code editor:

```
code ~/curiosity_mars_rover_ws/src/curiosity_mars_rover_navigation/
```

4. Make sure to install two extensions, ‘Python’ and ‘Test Explorer UI’.

5. In the ‘Testing’ tab, click ‘Configure Python Tests’.

6. Click ‘unittest’, choose the ‘test’ folder and click ‘*_test.py’.

7. Click the ‘Run Tests’ icon in the ‘Testing’ tab. The rover should begin to move, and the tests should pass once the rover reaches its goals.

B.5 Building and Making Changes

At any point, you can rebuild the Curiosity simulation by navigating to the Catkin workspace and running the *catkin_make* command:

```
cd ~/curiosity_mars_rover_ws/
catkin_make
```

To get started with implementing more Gazebo worlds, you can duplicate any of the launch files in the Gazebo package. Look for the line that has the *world* parameter:

```
<arg name="world" value="$(find curiosity_mars_rover_gazebo)/worlds/mars_terrain.world"/>
```

Here the ROS launch file is looking for a package, then finding a Gazebo world file in that package. You can make your own world, save it to the Gazebo *worlds* directory and have it launch by changing the *world* parameter:

```
<arg name="world" value="$(find curiosity_mars_rover_gazebo)/worlds/gallifrey.world"/>
```

Information about creating your own Gazebo worlds is available at https://classic.gazebosim.org/tutorials?tut=build_world.

The rover control launch file can also be extended to launch any additional Python scripts, using the *node* XML definition. For example, the arm/mast node is launched with the following code:

```
<node name="curiosity_rover_arm_and_mast_node" pkg="curiosity_mars_rover_control" type="
    arm_and_mast.py" respawn="false" output="screen" args=""/>
```

You can copy and modify this code to work with your own Python files. Replace *curiosity_rover_arm_and_mast_node* with the name you would like your ROS node to have, set the *pkg* to the ROS package your script is in, and then change the *type* of the node to be the Python filename. It will then launch automatically when new simulations are started.

The software package contain many opportunities for future development. More simulation environments could be added, the Python services for arm and mast control can be enhanced, more sophisticated autonomous navigation code could be implemented and even more features could be added to MarsViz. If you would like to learn more about the software package's structure before making changes, an explanation of each file's purpose is provided in Appendix D.