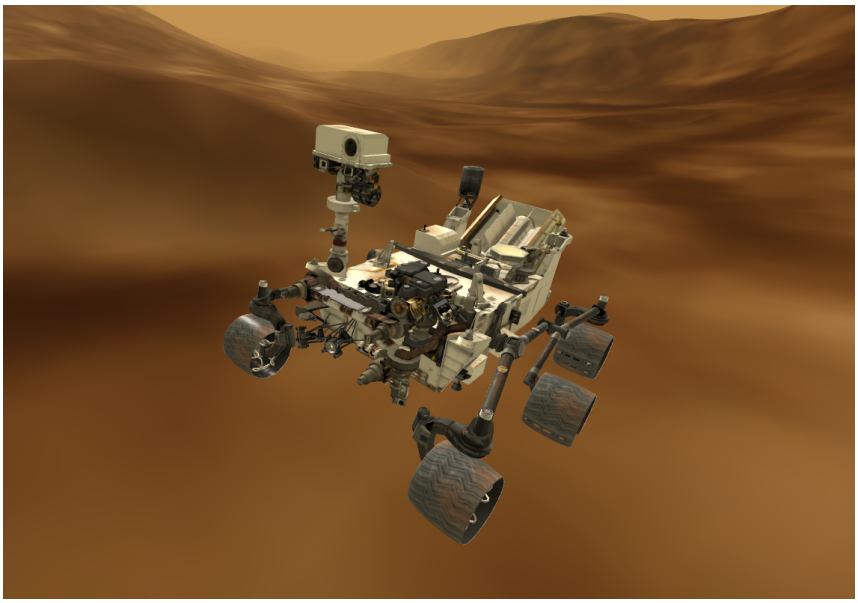
Appendix A

User Manual

This user manual explains how to get started with the Curiosity rover simulation package and make use of all the features it provides.



A.1 Requirements and Installation

In order to run the Curiosity simulation, you will need to have a Linux distribution installed. If you are using Windows 11, you can use WSL (Windows Subsystem for Linux) to access a virtualised Linux environment that works with graphical applications. The only Linux distribution that this software has been tested with is Ubuntu 20.04, so for the best results this version is highly recommended. You will also need at least 4.8 GB of free disk space. Since there are a lot of steps involved in manually installing everything, a script has been provided that will set up the simulation automatically. No extra software needs to be installed onto Ubuntu 20.04 prior to running this script.

First, make sure that the *curiosity_mars_rover_ws* folder from the software package is placed in your home folder. You can do this via a file manager or terminal. New terminal windows can easily be opened in Ubuntu with the Ctrl+Alt+T keyboard shortcut. You can use the `cd` command to navigate to the folder you downloaded the software package to. Move the *curiosity_mars_rover_ws* folder to your home folder with the following command:

```
mv curiosity_mars_rover_ws ~/
```

The folder itself should be there, not just its contents. You can check that it is using the following commands and looking for *curiosity_mars_rover_ws* in the output:

```
cd ~/
ls
```

Now all you need to do is navigate to the *curiosity_mars_rover_ws* folder and run the installation script with the following commands:

```
cd curiosity_mars_rover_ws
bash install_everything.sh
```

You will be prompted to enter your password since many of the commands in the script require elevated privileges. After entering your password, the installation will begin. It will take at least 10 minutes for everything to install, so feel free to get a cup of tea (or coffee) while you wait!

After the installation is complete, about 4.8 GB of disk space will be used by the source code for ROS, Gazebo, and the various packages used throughout the Curiosity simulation. If everything was successfully installed, your terminal should look something like this:

```
[ 19%] Built target _curiosity_mars_rover_control_generate_messages_check_deps_PanoramaActionResult
[ 19%] Built target _curiosity_mars_rover_control_generate_messages_check_deps_Arm
[ 26%] Built target tf2_web_republisher_generate_messages_cpp
[ 36%] Built target tf2_web_republisher_generate_messages_eus
[ 45%] Built target tf2_web_republisher_generate_messages_lisp
[ 48%] Built target ackermann_drive_controller
[ 48%] Built target _curiosity_mars_rover_navigation_generate_messages_check_deps_Teleop
[ 55%] Built target curiosity_mars_rover_control_generate_messages_cpp
[ 67%] Built target curiosity_mars_rover_control_generate_messages_eus
[ 80%] Built target curiosity_mars_rover_control_generate_messages_nodejs
[ 86%] Built target curiosity_mars_rover_control_generate_messages_py
[ 91%] Built target curiosity_mars_rover_control_generate_messages_lisp
[ 91%] Built target tf2_web_republisher_generate_messages
[ 93%] Built target tf2_web_republisher
[ 96%] Built target curiosity_mars_rover_navigation_generate_messages_nodejs
[ 97%] Built target curiosity_mars_rover_navigation_generate_messages_cpp
[ 97%] Built target curiosity_mars_rover_navigation_generate_messages_eus
[ 98%] Built target curiosity_mars_rover_navigation_generate_messages_lisp
[100%] Built target curiosity_mars_rover_navigation_generate_messages_py
[100%] Built target curiosity_mars_rover_control_generate_messages
[100%] Built target curiosity_mars_rover_navigation_generate_messages
---
All done! Open a new terminal to start using ROS, Gazebo and the Curiosity rover simulation.
mark@desktop-pc:~/curiosity_mars_rover_ws$
```

You will need to reload the current terminal window in order to access the simulation. Run the following command to reload the terminal:

```
exec bash
```

Finally, you are now able to run ROS launch files using the `roslaunch` command. The format of a `roslaunch` command is as follows:

```
roslaunch name_of_ros_package name_of_launch_file.launch
```

The remaining sections in this manual will explain how to make use of the various launch files provided.

A.2 Running the Simulation

The simulation can be launched with the *mars_terrain* world using the following command:

```
roslaunch curiosity_mars_rover_gazebo main_mars_terrain.launch
```

After running this command, the terminal will provide information about the simulation status. Two windows will also open: Gazebo and Rviz. The simulation environment can be seen

in Gazebo, and the rover camera feeds can be monitored from Rviz. The launch files from the rover description and control packages do not need to be launched manually as the *curiosity_mars_rover_gazebo* package does this automatically. The available launch files are as follows:

- `main_simple.launch` - opens a flat, empty world
- `main_mars_path.launch` - opens the original Mars world
- `main_mars_terrain.launch` - opens the world produced using DTM data
- `main_mars_photo.launch` - opens the photogrammetry world
- `main_shapes.launch` - opens the 'shapes' world for navigation testing

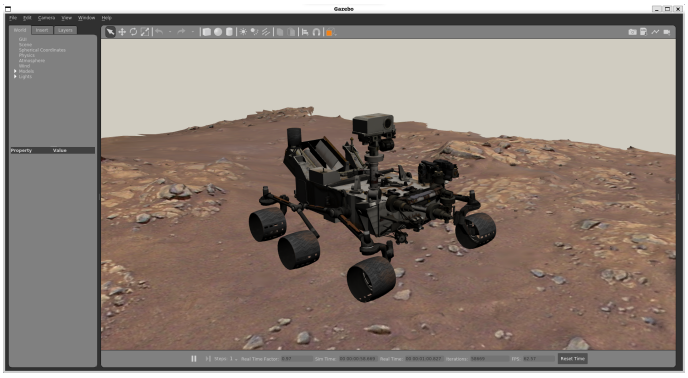
Each of these launch files can also be modified with the configuration parameters mentioned in the report. The available parameters are:

- `fake_depth` - replaces stereo cameras with depth cameras
- `fake_gps` - broadcasts rover's position from Gazebo to ROS
- `simple_camera` - disables all rover cameras except the ChemCam
- `rviz` - whether to open an Rviz window upon launch

The parameters can be invoked by adding `parameter_name:=true` after a launch file command. For instance, the following command can be used to launch the photogrammetry world with one camera:

```
roslaunch curiosity_mars_rover_gazebo main_mars_photo.launch simple_camera:=true
```

This example command will launch the least computationally demanding simulation setup available. If everything has installed correctly, after running the command the Gazebo window will look something like this:



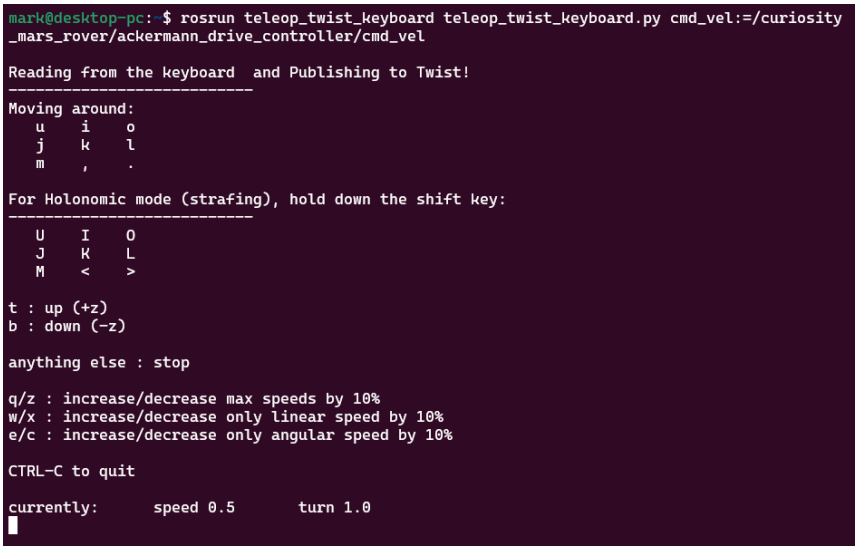
To access other functionality including rover teleoperation, autonomous navigation and visualisation, you will need to open a second terminal window for running additional ROS launch files. On Ubuntu, you can easily do this with the Ctrl+Alt+T keyboard shortcut.

A.3 Wheel Teleoperation

Teleoperation of the rover wheels can be accessed via existing ROS packages that publish to *cmd_vel* topics. The *teleop_twist_keyboard* package is included by the installation script, so it can be used to demonstrate basic teleoperation of the rover. You can start teleoperating the rover from a terminal with the following command:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/curiosity_mars_rover/
    ackermann_drive_controller/cmd_vel
```

The above command uses `rosrun` to directly launch a Python script without using a launch file. A parameter is passed to the script to tell it which ROS topic to publish *cmd_vel* messages to. After running it, you will see an interface like the following:



As indicated by the interface, keys on the right of the keyboard can be used to drive the rover around. You do not need to worry about the 'Holonomic mode' it mentions, as the Curiosity rover is a non-holonomic robot. All of the relevant *teleop_twist_keyboard* controls are provided below.

Key	**Function**
u	Drive forward & left
i	Drive forward
o	Drive forward & right
j	Turn left on the spot
l	Turn right on the spot
m	Drive backward & left
Comma key ,	Drive backward
Period key .	Drive backward & right
q	Increase driving and turning speed by 10%
z	Decrease driving and turning speed by 10%
w	Increase driving speed by 10%
x	Decrease driving speed by 10%
e	Increase turning speed by 10%
c	Decrease turning speed by 10%
Ctrl+C	Exit teleoperation interface

It is recommended that you increase the speed of the rover before driving, since by default it will drive at the real Curiosity rover's top speed (5 cm/second). Note that the terminal window needs to be in focus for keyboard input to be received, so you may want to put Gazebo and your terminal side-by-side so that you can see the rover driving. If you would like to access the teleoperation interface with obstacle avoidance feedback messages, you can launch it from the navigation package (you can exit the *teleop_twist_keyboard* interface before doing this, or simply open a new terminal window):

```
roslaunch curiosity_mars_rover_navigation teleop_keyboard_obstacles.launch
```

The terminal will display some information about subscribing to sensor data. While this interface is open, the controls are the same as *teleop_twist_keyboard*. If you attempt to drive too close to an obstacle, the terminal should display a message saying that the rover cannot move further in that direction. Note that for the best results with obstacle avoidance, it is recommended that you use the `fake_depth:=true` parameter when launching both the simulation and the `teleop_keyboard_obstacles.launch` file. This will result in much more accurate obstacle identification.

A.4 Arm and Mast Teleoperation

Using a similar approach to the wheel teleoperation in Section A.3, the rover's mast can also be teleoperated. Running the following command will open the mast teleoperation interface:

```
roslaunch curiosity_mars_rover_control teleop_keyboard_mast.launch
```

You will see a simplified version of the *teleop_twist_keyboard* interface designed for working with the rover mast. All of the available mast teleoperation controls for this interface are listed below.

Key	**Function**
u	Rotate mast upward & left
i	Rotate mast upward
o	Rotate mast upward & right
j	Rotate mast left
l	Rotate mast right
m	Rotate mast downward & left
Comma key ,	Rotate mast downward
Period key .	Rotate mast downward & right
q	Increase all rotation speeds by 10%
z	Decrease all rotation speeds by 10%
w	Increase vertical rotation speed by 10%
x	Decrease vertical rotation by 10%
e	Increase horizontal rotation by 10%
c	Decrease horizontal rotation by 10%
n	Raise or lower mast (depending on its state)
Ctrl+C	Exit teleoperation interface

As well as the mast teleoperation interface, there are ROS services for the mast and arm that provide direct control of the rover's joints. ROS services can be interacted with via a terminal using the `rosservice call` command. These commands take a full ROS message as a parameter,

which can be difficult to type out. The easiest way to make a service call is therefore to use tab completion. Start by typing into a terminal (do not press enter):

```
rosservice call /curiosity_mars_rover/mast_service
```

Now press the tab key on your keyboard, and it will add a space. Press the tab key again, and something like the following will appear:

```
mark@desktop-pc:~$ rosservice call /curiosity_mars_rover/mast_service "{mode: '',
pos_mast_p: 0.0, pos_mast_02: 0.0, pos_mast_cameras: 0.0, rot_x: 0.0, rot_y: 0.0}"
```

This is a *Mast* message. Not all of the parameters need to be set, but you do need to specify a *mode*. Use the arrow keys to scroll back to where it says `mode: ''` and specify one of the available instruction modes inside the parenthesis:

- `open` - raises the mast (ignores all other variables)
- `close` - lowers the mast (ignores all other variables)
- `toggle` - raises/lowers the mast (ignores all other variables)
- `set` - sets each mast position joint to the values specified in `pos_mast_p`, `pos_mast_02` and `pos_mast_cameras`
- `rotate` - rotates the mast horizontally and vertically by the amount specified in `rot_x` and `rot_y`

For example, to lower the mast:

```
rosservice call /curiosity_mars_rover/mast_service "{mode: 'close', pos_mast_p: 0.0,
    pos_mast_02: 0.0, pos_mast_cameras: 0.0, rot_x: 0.0, rot_y: 0.0}"
```

If the service call was successful, a message will appear providing feedback like the following:

```
mark@desktop-pc:~$ rosservice call /curiosity_mars_rover/mast_service "{mode: 'clos
e', pos_mast_p: 0.0, pos_mast_02: 0.0, pos_mast_cameras: 0.0, rot_x: 0.0, rot_y: 0.
0}"
success: True
status_message: "Done! Mast Mode: Lowered"
mark@desktop-pc:~$
```

The same principle demonstrated here can be applied to the robotic arm service, too. An example service call for the arm would be:

```
rosservice call /curiosity_mars_rover/arm_service "{mode: 'open', pos_arm_01: 0.0,
    pos_arm_02: 0.0, pos_arm_03: 0.0, pos_arm_04: 0.0, pos_arm_tools: 0.0}"
```

The robotic arm has a slightly different set of available modes and parameters to the mast. They are listed here:

- `open` - opens the arm (ignores all other variables)
- `close` - closes the arm (ignores all other variables)

- `toggle` - opens/closes the arm (ignores all other variables)
- `set` - sets each arm position joint to the values specified in `pos_arm_01`, `pos_arm_02`, `pos_arm_03`, `pos_arm_04`, and `pos_arm_tools`

The experimental mast panorama action server can also be launched via the command line. Run the following command to start the server:

```
roslaunch curiosity_mars_rover_control panorama_action_server.launch
```

You will need to publish a message to a topic in order to start a panorama action. Similar to the ROS services, full messages are included in these commands so it's much easier to use tab completion to write each command in a terminal. Open another terminal window and start by typing:

```
rostopic pub /panorama_server_node/goal curiosity_mars_rover_control/PanoramaActionGoal
```

You can then press the tab key twice. None of the parameters need to be changed in order to take a panorama, so you can simply press enter. If you return to the terminal window you opened the panorama action server in, you should see messages such as:

```
[INFO] [1653246545.767422, 5211.076000]: New goal recieved.
[INFO][139826015762176][/hugin_panorama/hugin_panorama/HuginPanorama.reset_callback:41]:
Clearing images waiting to be stitched.
[INFO] [1653246550.799505, 5216.088000]: 0.00%
[INFO] [1653246550.803952, 5216.092000]: 3.33%
[ INFO][7f74e0f6d5c0][/hugin_panorama/image_saver/saveImage:168]: Saved image /home/mark/
curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/0000.png
[INFO] [1653246551.807573, 5217.092000]: 6.67%
[ INFO][7f74e0f6d5c0][/hugin_panorama/image_saver/saveImage:168]: Saved image /home/mark/
curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/0001.png
[INFO] [1653246552.815402, 5218.092000]: 10.00%
[ INFO][7f74e0f6d5c0][/hugin_panorama/image_saver/saveImage:168]: Saved image /home/mark/
curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/0002.png
[INFO] [1653246553.824074, 5219.092000]: 13.33%
[ INFO][7f74e0f6d5c0][/hugin_panorama/image_saver/saveImage:168]: Saved image /home/mark/
curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/0003.png
[INFO] [1653246554.830040, 5220.092000]: 16.67%
[ INFO][7f74e0f6d5c0][/hugin_panorama/image_saver/saveImage:168]: Saved image /home/mark/
curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/0004.png
[INFO] [1653246555.834544, 5221.092000]: 20.00%
[ INFO][7f74e0f6d5c0][/hugin_panorama/image_saver/saveImage:168]: Saved image /home/mark/
curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/0005.png
[INFO] [1653246556.838845, 5222.092000]: 23.33%
```

The `curiosity_mars_rover_ws/src/dependencies/hugin_panorama/images/` folder will store any photos taken by the mast. Additionally, any mast service requests sent while a panorama is being taken will be rejected. However, you can prematurely cancel the panorama-taking process by sending a *cancel* message with the following command:

```
rostopic pub /panorama_server_node/cancel actionlib_msgs/GoalID "{stamp: {secs: 0, nsecs:
    0}, id: ''}"
```

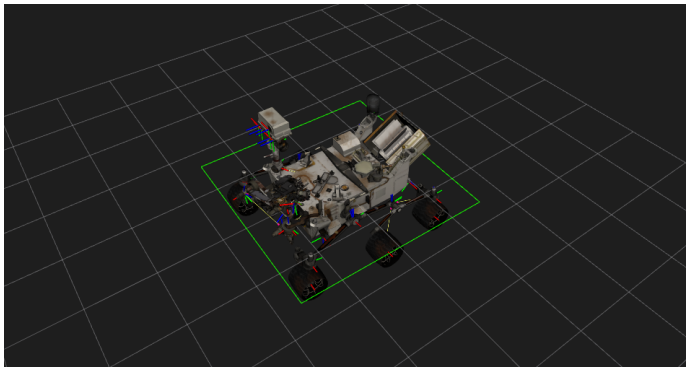
Most of the functionality described in this section can also be accessed via MarsViz for ease of use (see Section A.7).

A.5 Navigation

To start using the navigation features, make sure you already have a simulation running (see Section A.2 for the full list of simulation environment options). Open a new terminal (Ctrl+Alt+T on Ubuntu) and run the following command:

```
roslaunch curiosity_mars_rover_navigation navigation.launch
```

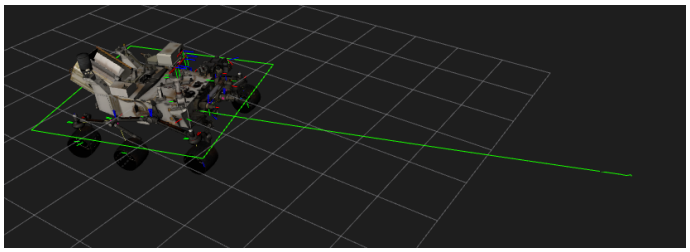
Navigation data can be monitored from Rviz. By default, Rviz will open when the simulation starts. If you do not see an Rviz window, you can refer to Section A.6 for information about opening a new one. After the navigation launch file runs, a green rectangle should appear around the rover in Rviz like so:



The green rectangle is the rover's *footprint*, the amount of space reserved on the costmaps for the rover. However, depending on the simulation configuration chosen, the costmap may initially be blank as there are no obstacles. Assuming there are none, a navigation goal can be sent from Rviz using the following steps:

1. Click '2D Nav Goal' at the top of the screen
2. Click and hold somewhere on the 2D grid around the rover. An arrow will appear on the grid.
3. Move the mouse while still clicking and rotate the arrow until it faces the desired rover orientation.
4. Stop clicking to send the navigation goal.

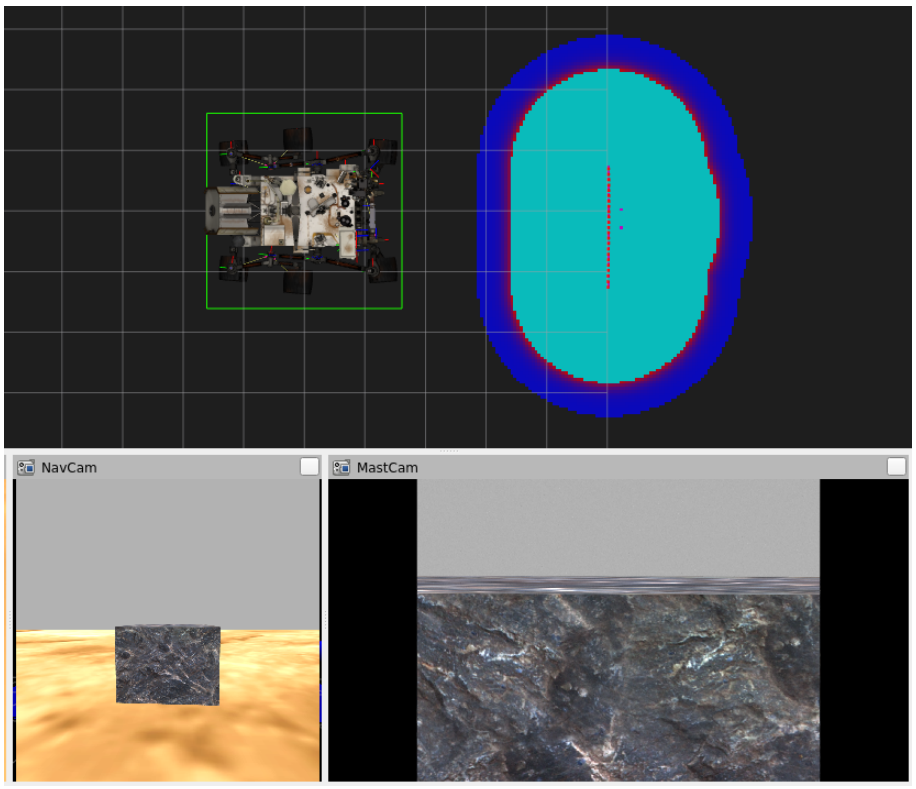
After completing this process, the rover will begin navigating to the goal. A green line in Rviz will indicate the path that is being followed.



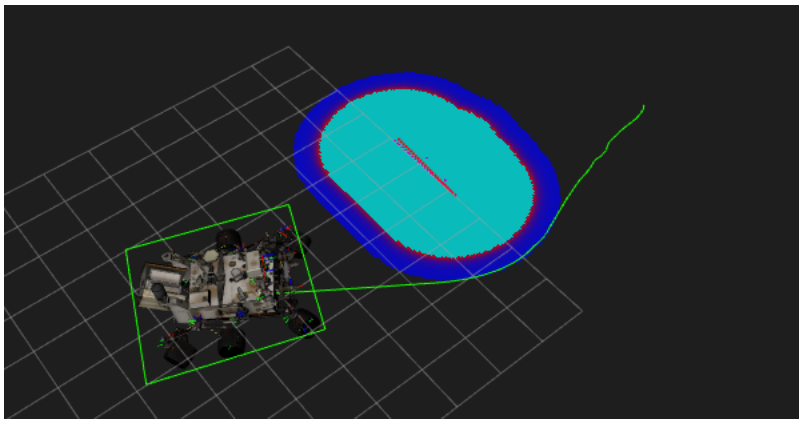
The *shapes* world can be used with the rover's depth camera configuration to demonstrate obstacle avoidance features. If you have a different simulation world running, you will need to close that one down before starting another simulation. In the terminal you used to launch the simulation, hit Ctrl+C to close everything. Afterwards, the *shapes* world and rover depth cameras can be launched with the following:

```
roslaunch curiosity_mars_rover_gazebo main_shapes.launch fake_depth:=true
```

When using simulations with the depth camera configuration, you also need to launch the navigation system with the `fake_depth:=true` parameter so that the costmaps know what sensor data to look for. The costmap in the *shapes* world should initially appear like so:



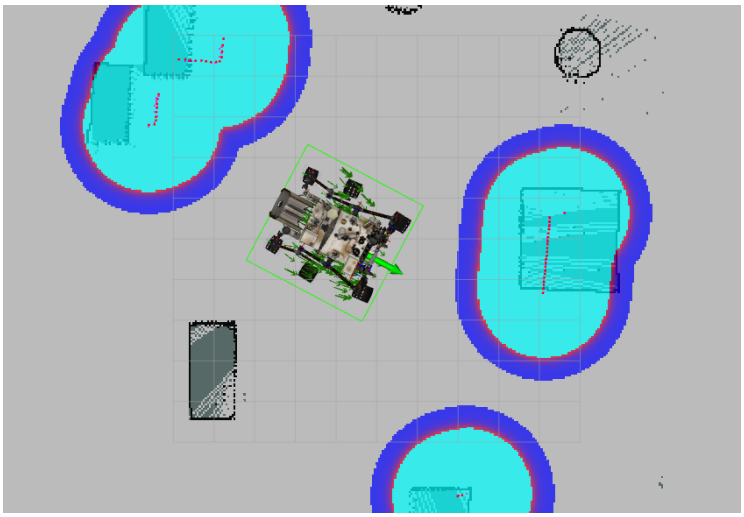
The large block visible in the NavCam and MastCam is represented in the costmap as a line of red 'obstacle' points. They are surrounded by the blue *inflation layer* that the planners will avoid so that the rover footprint stays away from obstacles. Try sending a navigation goal somewhere behind the large block, and the path generated should avoid it.



By default, the rover will rely on odometry to localise itself. However, an alternative configuration of the navigation launch file can load in a pre-defined map of the *shapes* world that can be compared with sensor data to determine the rover location. Try running the navigation system with the extra parameter `use_shapes_map:=true`:

```
roslaunch curiosity_mars_rover_navigation navigation.launch fake_depth:=true
    use_shapes_map:=true
```

After opening this launch file, a map of the *shapes* world will appear in Rviz. The map will continuously readjust its position based on current rover sensor data as it tries to localise the rover. A series of green arrows appearing around the rover will indicate potential rover locations relative to the map. One large green arrow under the rover will indicate the 'best' estimate location.



New maps can also be created using the *mapping_2d.launch* file like so:

```
roslaunch curiosity_mars_rover_navigation mapping_2d.launch fake_depth:=true
```

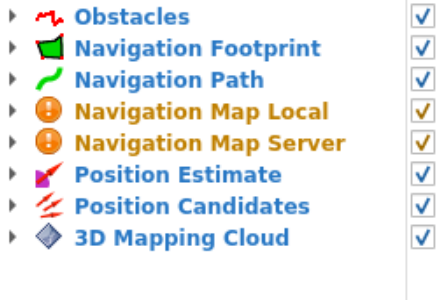
This will start recording sensor data. The map will be updated in Rviz as more obstacles are recorded onto it. To gather data from more angles and positions, you can teleoperate the rover wheels or use the autonomous navigation system. The map can then be saved to a file with the following command:

```
rosrun map_server map_saver -f 'filename'
```

Finally, the 3D mapping tool *rtabmap* can be demonstrated by the `mapping_3d.launch` file. This tool is most effective when using a high-detail simulation environment like the *mars_photo* world. It also works best with the default stereo camera configuration of the Curiosity rover simulation. To try it out, start a new simulation (run each of these commands in a separate terminal):

```
roslaunch curiosity_mars_rover_gazebo main_mars_photo.launch
roslaunch curiosity_mars_rover_navigation mapping_3d.launch
```

Now, on the left of Rviz, tick the box on the right of the text '3D Mapping Cloud'.



3D points should start to appear around the rover. Just like with the 2D mapping system, you can either teleoperate the rover or have it autonomously navigate in order to build up a more detailed map.

A.6 Using Rviz

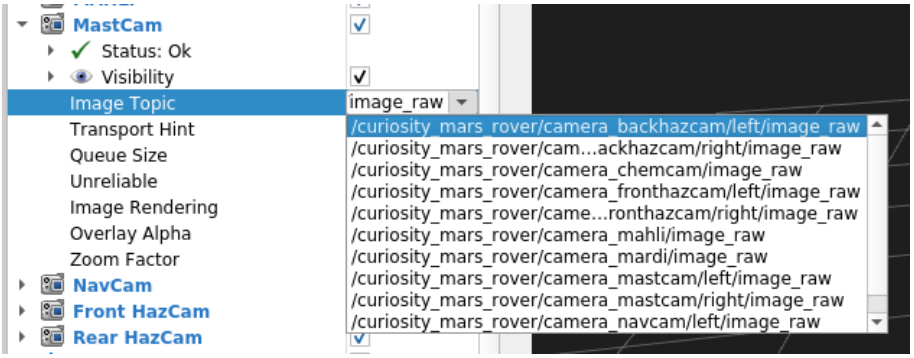
New windows of Rviz can be launched at any time during simulations using the following command:

```
roslaunch curiosity_mars_rover_viz rviz.launch
```

The Rviz launch file has two optional parameters, `fake_depth:=true` and `simple_camera:=true`. The `fake_depth:=true` parameter launches an Rviz configuration for the rover designed for depth cameras. The `simple_camera:=true` parameter launches a basic Rviz setup with only one camera and no navigation visualisations.

The Rviz configurations provided in the software package can display a variety of information about the simulated rover. The panel on the left of Rviz shows you which visualisations of information (referred to as 'displays') are enabled. By default, not all of the available displays are enabled for performance reasons. For example, the 3D depth information being produced by either the stereo cameras or depth cameras can be enabled by ticking the box next to the 'Depth' displays. For the depth cameras, this data will take a considerable amount of computing power to display.

Additionally, the camera displays included can be modified to display different camera feeds. By clicking on the arrow next to a display, properties for that display will appear in Rviz. The topic that it subscribes to can be selected like so:



Further information about using Rviz is available at `http://wiki.ros.org/rviz/UserGuide`.

A.7 Using MarsViz

To make the most out of MarsViz, simulations can be launched directly from the visualisation package. The package contains the following launch files for MarsViz:

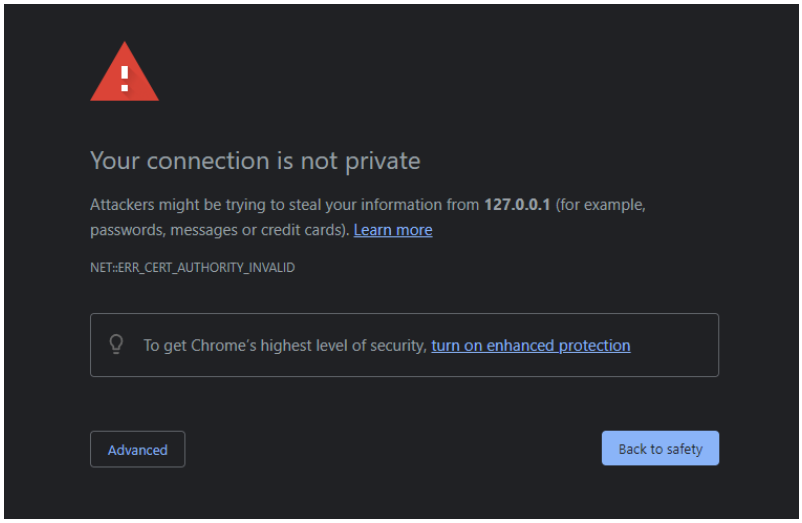
- `marsviz.launch` - launches MarsViz server assuming a simulation is already running
- `marsviz_mars_path.launch` - launches simulation with *mars_path* world and MarsViz server
- `marsviz_mars_photo.launch` - launches simulation with *mars_photo* world and MarsViz server
- `marsviz_mars_terrain.launch` - launches simulation with *mars_terrain* world and MarsViz server

The advantage of launching simulations at the same time as MarsViz is that they will already be pre-configured to use the rover's simulated GPS signal. All of the parameters accepted by the original Gazebo launch files are also accepted by these launch files.

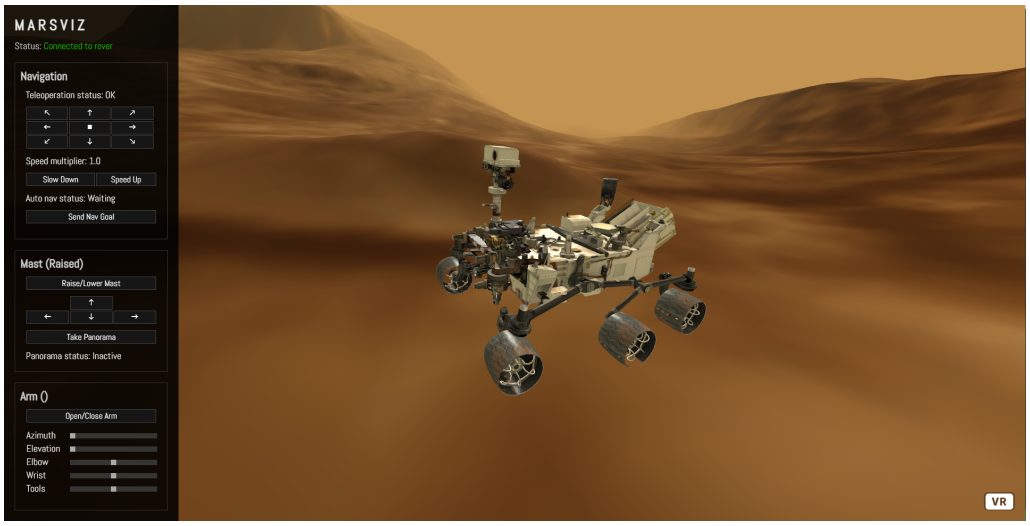
The best way to get started with MarsViz is to close any currently running simulation and start the simulation/web server like so (you can swap *mars_terrain* for your preferred simulation environment):

```
roslaunch curiosity_mars_rover_viz marsviz_mars_terrain.launch
```

After running this command, Gazebo will open but Rviz will not. You can access MarsViz by visiting `https://127.0.0.1:8080/` in a web browser. Due to modern browser security policies, the first thing you see will probably be a warning that 'Your connection is not private':



The web application is only running on your local machine, so there is not actually any security risk. The warning can be ignored by clicking 'Advanced' and then 'Proceed to 127.0.0.1 (unsafe)'. You will now be able to see the MarsViz interface!



If you cannot see the environment on your first try, you may need to refresh the page so that your browser can clear up any leftover security errors. You will now be able to access all of the controls that MarsViz makes available. The user perspective can be manipulated using a mouse

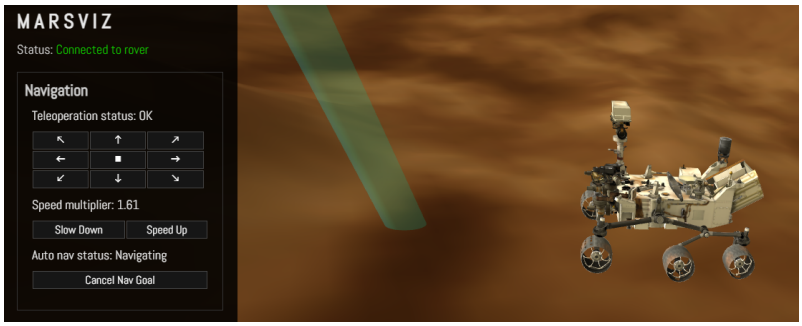
and keyboard. Drag the mouse around the view in order to look around. The W, A, S and D keys will allow you to fly through the environment. A full list of MarsViz keyboard controls is provided below.

Key	Function
w	Fly forward
s	Fly backward
a	Fly to the left
d	Fly to the right
i	Drive rover forward
k	Drive rover backward
j	Turn rover left
l	Turn rover right
q	Increase driving and turning speed by 10%
z	Decrease driving and turning speed by 10%
t	Rotate mast upward
g	Rotate mast downward
f	Rotate mast left
h	Rotate mast right
m	Raise or lower mast (depending on its state)
n	Open or close arm (depending on its state)

The control panel on the left provides access to similar functionality as the keyboard controls, with a few additions. It can be used to send autonomous navigation goals using a similar process to Rviz:

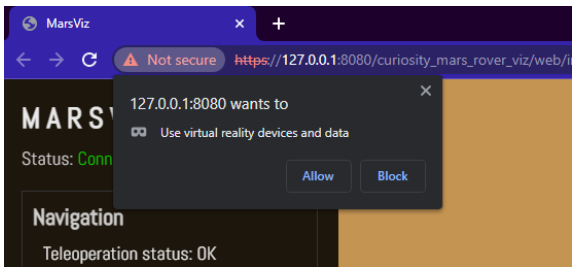
1. Click 'Send Nav Goal' in the control panel at the side.
2. Hover over the terrain with your mouse. You will see a blue arrow appear.
3. Click somewhere to place the arrow. You can now rotate it into the desired orientation.
4. Click a second time to send the navigation goal.

A blue line will appear to indicate where the rover is navigating to. You can also easily cancel the goal by pressing the same button as before, which will now read 'Cancel Nav Goal':



The *Mast* section of the control panel can be used to automatically take panoramas. Simply click the 'Take Panorama' button and the mast will begin to rotate and take photos. You can also cancel the panorama before it is finished, just like with the autonomous navigation system. The *Arm* section of the control panel also allows each individual arm joint to be positioned manually.

Finally, MarsViz can also be used in virtual reality if you have a headset compatible with WebXR. To take advantage of this feature, you will need to make sure that your headset's runtime software is already running (for example, SteamVR or Oculus Home). If you click the 'VR' button in the bottom right corner of MarsViz and your browser supports WebXR, you will get a prompt for website permissions like so:



If you click 'Allow' and put on your headset, you will now be able to see MarsViz in virtual reality. The VR version has the following capabilities:

- Using a motion controller in your left hand, you can hold the trigger button and a blue circle will appear. You can point at a location on the ground and the circle will move to that point. When you release the trigger, the rover will begin to autonomously navigate to the blue circle.
- Using a motion controller in your right hand, you can follow the same process and a green circle will appear. Instead of moving the rover, releasing the trigger will instantly teleport you to the location you pointed at.
- Pushing the thumbstick/touchpad on a motion controller will manually drive the rover in the direction that you push.
- Pressing one of the front buttons on your left-hand motion controller will raise or lower the mast, while pressing these buttons on the right-hand motion controller will open or close the robotic arm.