ROS-Gazebo-ErleRover Installation and Usage Guide
Michigan State University

COLLEGE OF ENGINEERING

Glen Simon
Philip McKinley
Jared Moore
Anthony Clark

January 15, 2018

Contents

Chapter 1

Introduction

Chapter 2

Installation

These installation directions are derived from the instructions provided by Erle Robotics found at http://docs.erlerobotics.com/simulation/configuring_your_environment.

It is recommend that this software is used on a machine running Ubuntu 14.04 64 bits.

2.1 Configuring your Ubuntu Machine

These steps only have to be done once per machine.

2.1.1 Install base packages

```
sudo apt-get update
sudo apt-get install gawk make git curl cmake -y
```

2.1.2 Install dependencies for MAVProxy

```
sudo apt-get install g++ python-pip python-matplotlib python-serial python-wxgtk2.8 python-
    scipy -y
sudo apt-get install python-opencv python-numpy python-pyparsing ccache realpath libopencv-
    dev -y
```

2.1.3 Install MAVProxy

```
sudo pip install future
sudo apt-get install libxml2-dev libxslt1-dev -y
sudo pip2 install pymavlink catkin_pkg --upgrade
sudo pip install -I MAVProxy==1.5.2
```

2.1.4 Download and install ArUco

1. Download ArUco 1.3.0 from here https://sourceforge.net/projects/aruco/files/1.3.0/aruco-1.3.0.tgz/download

2. Install ArUco

```
cd ~/Downloads # Replace this with your Download directory
tar -xvzf aruco-1.3.0.tgz
cd aruco-1.3.0/
mkdir build && cd build
cmake ..
make
sudo make install
```

2.1.5 Install ROS Indigo

Setup your computer to accept software from packages.ros.org, setup your keys and install (make sure your Debian package index is up-to-date):

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt
    /sources.list.d/ros-latest.list'
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net --recv-key 0xB01FA116
sudo apt-get update
```

Install, ROS package, build, and communication libraries. No GUI tools.:

```
sudo apt-get install ros-indigo-ros-base -y
```

Initialize rosdep, before you can use ROS, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS.

```
sudo rosdep init
rosdep update
```

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

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

Get rosinstall and some additional dependencies

```
sudo apt-get  install python-rosinstall      \
ros-indigo-octomap-msgs \
ros-indigo-joy          \
ros-indigo-geodesy      \
ros-indigo-octomap-ros  \
ros-indigo-mavlink      \
ros-indigo-control-toolbox \
ros-indigo-transmission-interface \
ros-indigo-joint-limits-interface \
unzip -y
```

Get RQT graph

```
sudo apt-get install ros-indigo-rqt
sudo apt-get install ros-indigo-rqt-common-plugins
```

2.1.6 Install Gazebo

Setup your computer to accept software from packages.osrfoundation.org

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -
    cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
```

Setup keys

```
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
```

Install gazebo7

```
sudo apt-get update
sudo apt-get remove .*gazebo.* '.*sdformat.*' '.*ignition-math.*' && sudo apt-get update &&
    sudo apt-get install gazebo7 libgazebo7-dev drcsim7 -y
```

Chapter 3

Configuring User Workspace

These steps will have to be done for each user on a machine if they would like their own local copies of the source files.

3.1 Download Ardupilot

The ArduPilot project is an open source autopilot for drones. We'll be using its code to simulate the UAVs:

3.1.1 Compile a specific branch of ardupilot

```
mkdir -p ~/simulation; cd ~/simulation
git clone https://github.com/erlerobot/ardupilot -b gazebo
```

3.2 Download ErleRover_Scripts directory

This was created to ease the process of starting all of the required processes used to simulate the Erle Rover.

```
cd ~/simulation
git clone https://github.com/gsimon2/ErleRover-Scripts.git
```

3.2.1 Getting permission to push commits to the remote repo

This is a public repo and can freely be copied, but for access to submit changes please contact Glen Simon at glen.a.simon@gmail.com.

3.3 Download ros_gazebo_python directory, which contains the BasicBot work. Optional

```
cd ~/simulation
git clone https://github.com/jaredmoore/ros_gazebo_python.git
```

3.3.1 Getting permission to push commits to the remote repo

This is a public repo and can freely be copied, but for access to submit changes please contact Jared Moore at swiftfoottim@gmail.com.

3.4 Create ROS workspace

3.4.1 Make workspace

```
mkdir -p ~/simulation/ros_catkin_ws/src
```

3.4.2 Initialize the workspace

```
cd ~/simulation/ros_catkin_ws/src
catkin_init_workspace
cd ~/simulation/ros_catkin_ws
catkin_make
source devel/setup.bash
```

3.4.3 Download ros_catkin_ws_src which contains the development work for the MSU rover project

```
cd ~/simulation/ros_catkin_ws
git clone https://github.com/gsimon2/ros_catkin_ws_src.git
```

Delete default src directory and replace with the downloaded one

```
cd ~/simulation/ros_catkin_ws
rm -r src
mv ros_catkin_ws_src src
```

Getting permission to push commits to the remote repo
This is a public repo and can freely be copied, but for access to submit changes please contact Glen Simon at glen.a.simon@gmail.com.

3.4.4 Compile the ros_catkin_ws workspace

```
cd ~/simulation/ros_catkin_ws
source devel/setup.bash
catkin_make --pkg mav_msgs mavros_msgs gazebo_msgs
catkin_make -j 4
```

3.5 Download Gazebo models

```
mkdir -p ~/.gazebo/models
git clone https://github.com/erlerobot/erle_gazebo_models
mv erle_gazebo_models/* ~/.gazebo/models
```

3.6 Configuring .bashrc

3.6.1 Add ROS setup to bash

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched.

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

3.6.2 Add ros_catkin_ws setup to bash

For ROS to find the packages provided in ros_catkin_ws we need to source the setup file every time. This is easier if we also add this to the bash file.

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

Chapter 4

Usage

4.1 Basic Erle-Rover simulation

This process will bring up the Erle-Rover in a blank world and allow you to manually enter throttle and yaw commands via the MAVProxy terminal.

4.1.1 Manually starting all needed processes

The process of starting all processes can be found in more detail at:
http://docs.erlerobotics.com/simulation/vehicles/erle_rover/tutorial_1, but will be covered briefly here.

Executing APMrover2

This process requires two active terminals.

In terminal one enter:

```
source ~/simulation/ros_catkin_ws/devel/setup.bash
cd ~/simulation/ardupilot/APMrover2
../Tools/autotest/sim_vehicle.sh -j 4 -f Gazebo
# once MAVProxy has launched completely, load the parameters
param load /[path_to_your_home_directory]/simulation/ardupilot/Tools/Frame_params/3DR_Rover.
    param
# NOTE: replace [path_to_your_home_directory] with the actual path to your home directory.
# Example: param load /home/john/simulation/ardupilot/Tools/Frame_params/3DR_Rover.param
```

In terminal two enter:

```
source ~/simulation/ros_catkin_ws/devel/setup.bash
roslaunch ardupilot_sitl_gazebo_plugin rover_spawn.launch
```

This should start the Gazebo GUI and you should be able to see that the rover spawned in a blank world appearing similar to figure 4.1.

Controlling Erle-Rover using MAVProxy

Make the rover move forward. In the first terminal execute:

```
# in the MAVProxy prompt:
mode MANUAL
param set SYSID_MYGCS 255
rc 3 1900
```

Or backwards:

```
# in the MAVProxy prompt:
rc 3 1200
```

What we are doing here is override the 3rd channel of the RC, which corresponds to the throttle. Values go from 1100 to 1900. 1500 is to stop the throttle; so values above 1500 will make the rover move forward, and values above 1500 backwards. The same principle applies to the yaw, which is in the 1st channel of the RC. Values above 1500 will make it turn right, and below 1500 left. For instance:

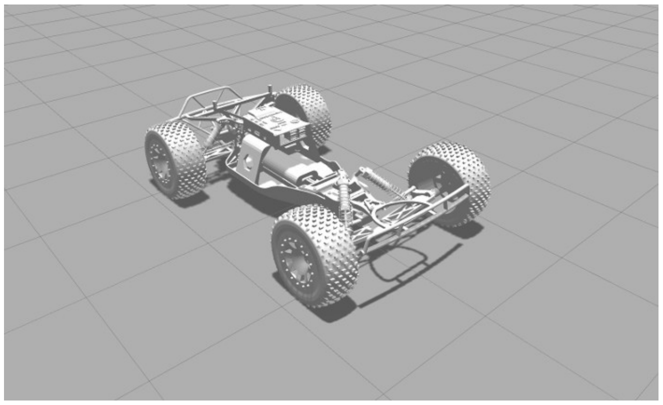


Figure 4.1: Erle-Rover model in Gazebo simulator

```
# in the MAVProxy prompt:
rc 1 1400
```

Note that first we had to use "param set SYSID_MYGCS 255". This tell MAVProxy where the source for rover commands are coming from. To control the rover manually via the MAVProxy terminal, this value must be set to 255. When using scripts to control the rover, this value must be set to 1.

4.1.2 Using scripts to start simulation

The above simulation can also be started using a script to make the start up process easier. To run the script enter the following commands:

```
cd ~/simulation/ErleRover-Scripts/
./basic_sim.sh
```

This script also runs the "start_FPV" script, which will bring up a window displaying a first person view from the rover's perspective. The "start_FPV" script can be ran on its own anytime a simulation is running to show the first person view by running the following commands:

```
cd ~/simulation/ErleRover-Scripts/
./start_FPV.sh
```

4.2 Starting MAVProxy via a script

MAVProxy is required when running any simulation of the rover and generally must manually be started prior to anything else. To make starting simulations easier, MAVProxy can be started via a script by using the following commands:

```
cd ~/simulation/ErleRover-Scripts/
./start_MAVProxy.sh
```

This will open an xterm window that sources the setup.bash file required to launch the Ardupilot SiTL software and execute Ardupilot's sim_vehicle script with the rover parameters already loaded into it. The sim_vehicle script will then launch MAVProxy in the xterm window and Ardupilot in a separate window, generally also xterm, but may vary depending on your system.

4.3 Basic obstacle avoidance simulation using Erle Robotics script

This process will bring up the rover in a basic maze and start a node running a basic obstacle avoidance algorithm provide by Erle Robotics which will lead the rover through the maze without crashing into the walls.

The process of starting all processes can be found in more detail at:
http://docs.erlerobotics.com/simulation/vehicles/erle_rover/tutorial_2, but will be covered briefly here.

4.3.1 Manually starting all needed processes

To run this simulation three processes must be started:

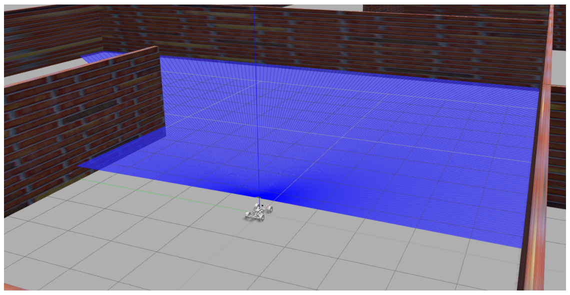
1. MAVProxy - This can be started manually, see terminal one in section 4.1.1, or with a script, see section 4.2.
2. The launch file - This can be done with the following command:

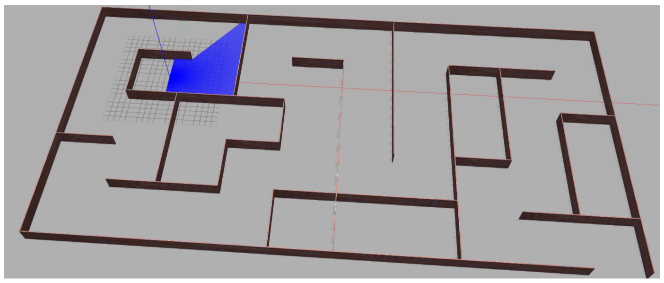
   ```
   roslaunch ardupilot_sitl_gazebo_plugin rover_maze.launch
   ```

3. The controller node - This can be done with the following command:

   ```
   rosrun erle_rover_explorer erle_rover_explorer.py
   ```

If this is done correctly you should see the rover spawn in a simple maze as can be seen in figure 4.2a. An overview of the maze can be seen in figure 4.2b.


(a) Erle-Rover in simple maze


(b) Simple maze overview

Notes

1. There is no stopping condition for this simulation and the rover will just continue to drive straight once it has exited the maze.
2. If the everything appears to have started correctly, but the rover is standing still in the maze, check to make sure MAVProxy is listening for commands on the right channel. For help with this issue see section 5.1.
3. The provided obstacle avoidance script also displays a window that represents the lidar scan and shows the current heading of the rover. An example of this can be seen in figure 4.3.
4. A first person view of the rover can be brought up by running the script. See section 4.1.2.

4.3.2 Using scripts to start simulation

The above simulation can also be started using a script to make the start up process easier. To run the script enter the following commands:

```
cd ~/simulation/ErleRover-Scripts/
./explorer_sim.sh
```

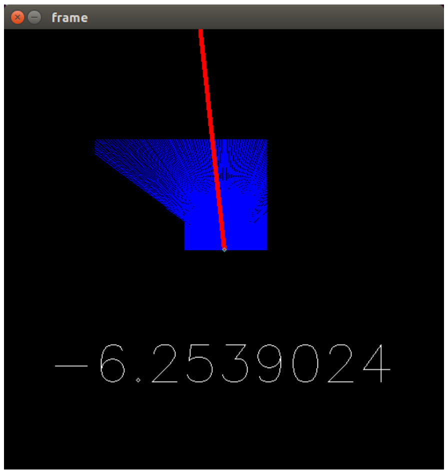


Figure 4.3: Lidar scan representation and current heading

4.4 Maze mapping using RViz

The scan results from either an on board lidar or sonar sensor can be fed to RViz to generate an image of the maze as it is known to the rover. To do so, follow these steps:

1. Start desired simulation - Launch all processes required for the simulation that you would like to map.
2.

Experiencing issues with hector_mapping mapping_default.launch currently. Will finish this section when I resolve issues.

4.5 Line follower simulation

Please follow directions provided at: http://docs.erlerobotics.com/simulation/vehicles/erle_rover/tutorial_3

4.6 Object finder simulation

Chapter 5

Troubleshooting

5.1 Rover is not responding to either manual or scripted commands

The most common issue for the rover not responding to commands as expected is that MAVProxy is listening for commands on the wrong channel. To check for this enter the following command into the MAVProxy terminal to check which channel is currently being listened to for commands:

```
param show SYSID_MYGCS
```

If the returned value is 1.0, MAVProxy is listening for commands from a scripted controller comminicating using a ROS topic.

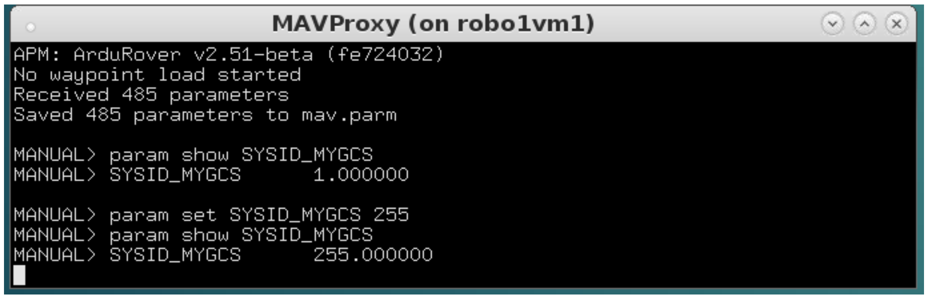
If the returned value is 255.0, MAVProxy is waiting for commands manually entered into the MAVProxy terminal.

To change the channel that MAVProxy is listening on use the following command:

```
param set SYSID_MYGCS {value}
```

where value is between 1 and 255.

Note: MAVProxy must be connected to the rover, either a physical rover or a ROS node acting as the rover, for these commands to work.



```
APM: ArduRover v2.51-beta (fe724032)
No waypoint load started
Received 485 parameters
Saved 485 parameters to mav.parm

MANUAL> param show SYSID_MYGCS
MANUAL> SYSID_MYGCS      1.000000

MANUAL> param set SYSID_MYGCS 255
MANUAL> param show SYSID_MYGCS
MANUAL> SYSID_MYGCS      255.000000
```

Figure 5.1: Checking MAVProxy command source