Follower ROVER
Name: Iesus René Dávila Aguilar

Content

Description

The project developed in ROS Melodic consists of the development of an algorithm that allows a ROVER to follow another ROVER that is in front. The algorithm does not focus on a particular robot, that is, the algorithm can be run on any ROVER, thus creating a chain of robot followers.

Important information

- Subscriber and Publisher is a type of communication in ROS.
- The cmd_vel topic receives linear and angular velocity.
- The LIDAR topic provides a list of 180 data, 1 data per angle.
- It only subscribes to the LIDAR topic and publishes to the cmd_vel topic.
- It does not use Machine Learning or any computer vision cameras.
- The algorithm separates the data into 5 regions, it always tries to have the main ROVER be in the front region (region 3).

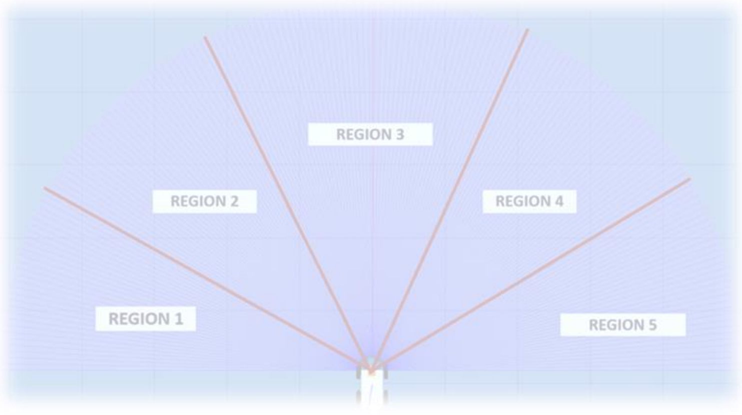


Figure 1 Region separations.

Characteristics

1. It is independent whether the main ROVER executes autonomous navigation or navigation through teleoperation.

2. It can be run on N ROVER at the same time.
3. Implements speed control by distance to the main ROVER.
4. The follower ROVER always tries to position itself behind the main ROVER.
5. The follower ROVER is located at a maximum distance of 0.5 meters from the main ROVER.

Work cycle

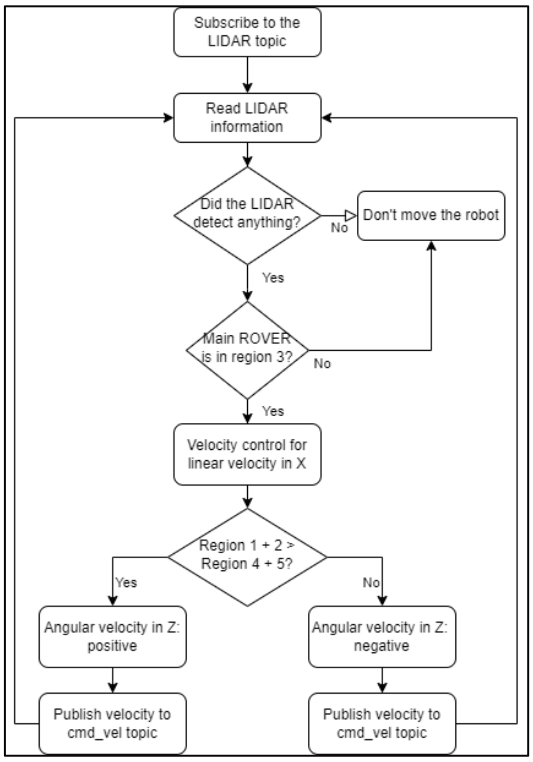

Subscribe to the LIDAR topic
Read LIDAR information
Did the LIDAR detect anything?
No
Don't move the robot
Yes
Main ROVER is in region 3?
No
Yes
Velocity control for linear velocity in X
Region 1 + 2 > Region 4 + 5?
Yes
No
Angular velocity in Z: positive
Angular velocity in Z: negative
Publish velocity to cmd_vel topic
Publish velocity to cmd_vel topic

Figure 2 Work cycle.

Operations for simulation execution

Operation for 1 follower ROVER and 1 main ROVER (teleoperation control)

1. Run the simulation: roslaunch main multi_rover_house.launch

   There are more worlds where to execute them are as follows:
   - roslaunch main multi_rover_empty.launch
   - roslaunch main multi_rover_world.launch

Follower ROVER
Name: Iesus René Dávila Aguilar

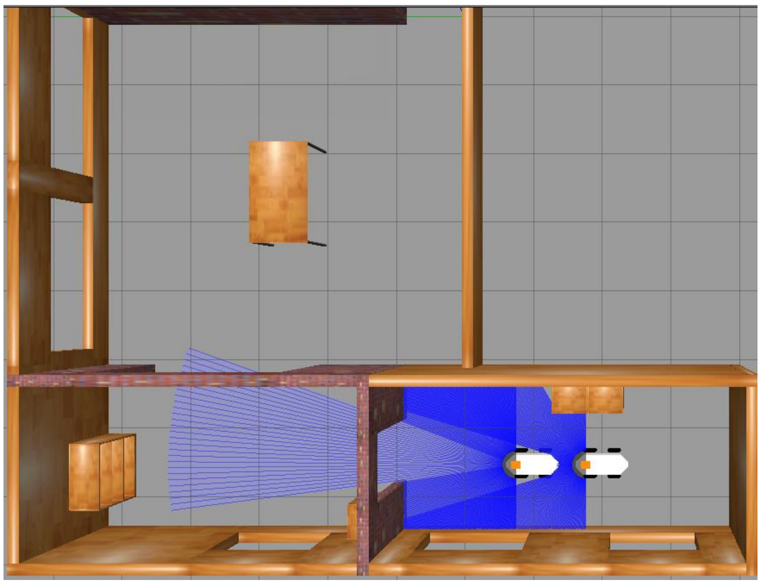
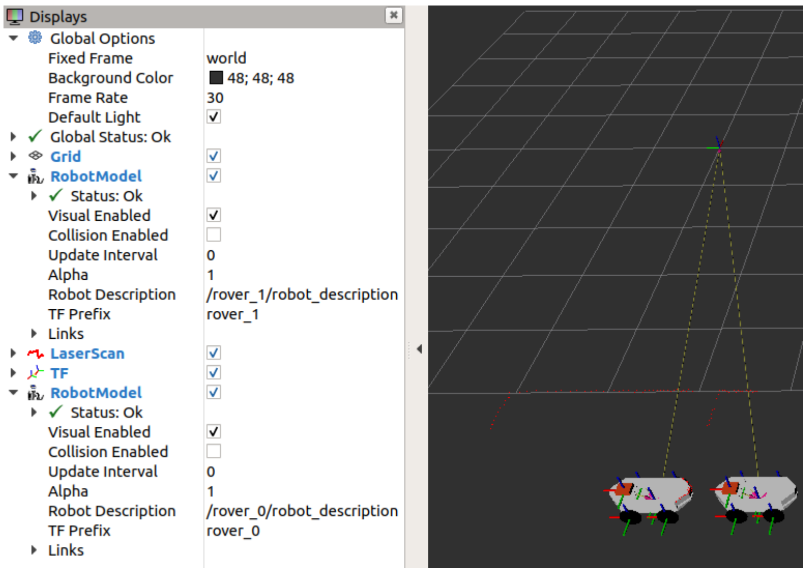


Figure 3 Multirobot simulation without autonomous navigation.



Displays	
Global Options	
Fixed Frame	world
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
✓ Global Status: Ok	
Grid	✓
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0
Alpha	1
Robot Description	/rover_1/robot_description
TF Prefix	rover_1
Links	
LaserScan	✓
TF	✓
RobotModel	✓
✓ Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Update Interval	0
Alpha	1
Robot Description	/rover_0/robot_description
TF Prefix	rover_0
Links	

Figure 4 Multirobot RViz without autonomous navigation.

2. Run the follower algorithm (example with the robot named rover_1): rosrun follower_rover follower_2.py NOMBRE_ROVER

Follower ROVER
Name: Iesus René Dávila Aguilar

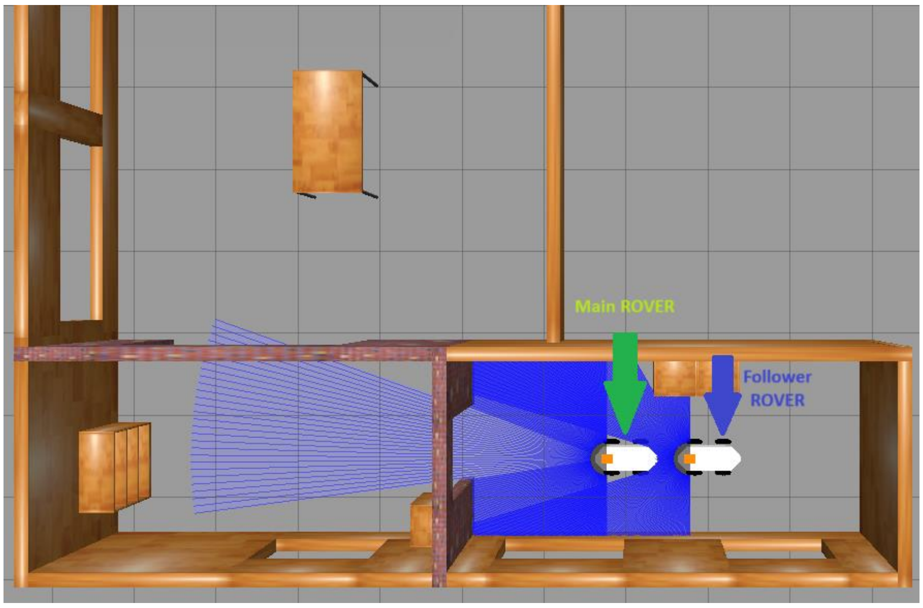
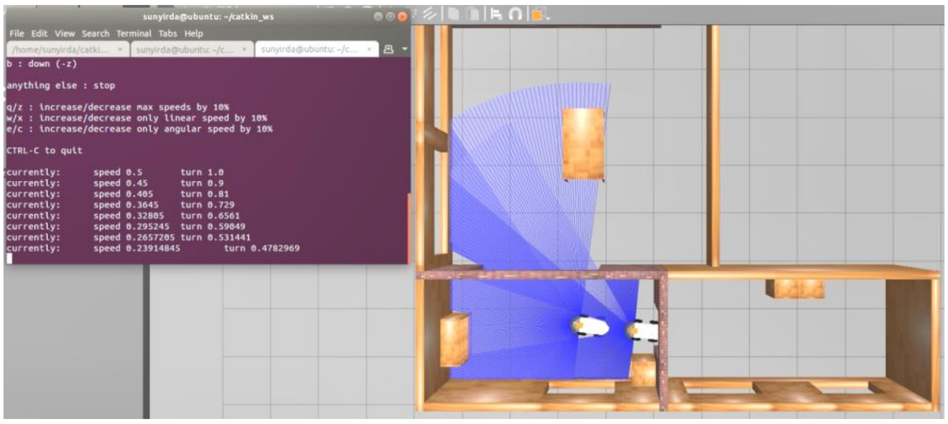


Figure 5 Follower ROVER

3. Execute the teleoperation (example with the robot named rover_0): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=NAME_ROBOT/cmd_vel



```
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
currently:      speed 0.45      turn 0.9
currently:      speed 0.405     turn 0.81
currently:      speed 0.3645    turn 0.729
currently:      speed 0.32805   turn 0.6561
currently:      speed 0.295245  turn 0.59049
currently:      speed 0.2657205 turn 0.531441
currently:      speed 0.23914845        turn 0.4782969
```

Figure 6 rover_0 teleoperation

Operation for 1 follower ROVER and 1 main ROVER (autonomous navigation)

1. Run the simulation: roslaunch main multi_rover_house.launch navigation:=true

   There are more worlds where to execute them are as follows:
   - roslaunch main multi_rover_empty.launch navigation:=true
   - roslaunch main multi_rover_world.launch navigation:=true

Follower ROVER
Name: Iesus René Dávila Aguilar

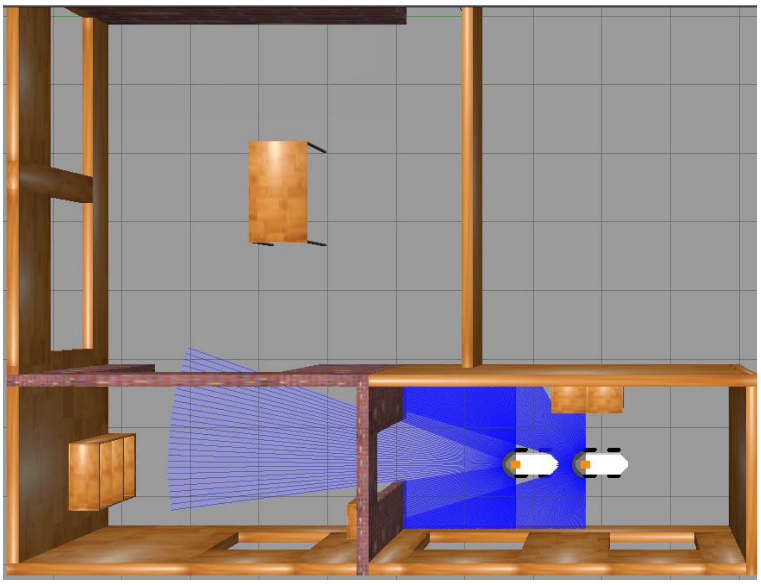
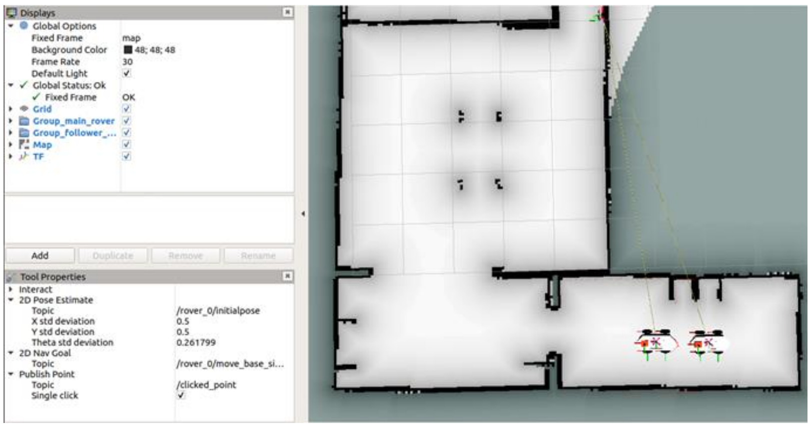


Figure 7 Multi-robot simulation with autonomous navigation.



```
Displays
▾ Global Options
    Fixed Frame             map
    Background Color        ■ 48; 48; 48
    Frame Rate              30
    Default Light           ✓
▾ ✓ Global Status: Ok
    ✓ Fixed Frame           OK
▸ Grid                      ✓
▸ Group_main_rover          ✓
▸ Group_follower_...        ✓
▸ Map                       ✓
▸ TF                        ✓

Tool Properties
▸ Interact
▾ 2D Pose Estimate
    Topic                   /rover_0/initialpose
    X std deviation         0.5
    Y std deviation         0.5
    Theta std deviation     0.261799
▾ 2D Nav Goal
    Topic                   /rover_0/move_base_si...
▾ Publish Point
    Topic                   /clicked_point
    Single click            ✓
```

Figure 8 RViz multirobot with autonomous navigation.

2. Run the follower algorithm (example with the robot named rover_1): rosrun follower_rover follower_2.py NOMBRE_ROVER

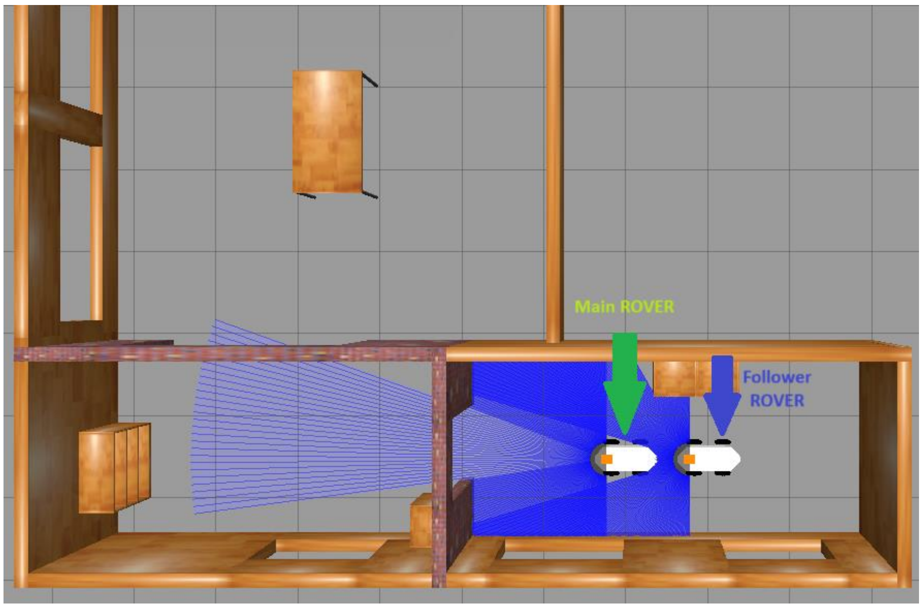


Figure 9 Follower ROVER.

Operation for 2 follower ROVERs and 1 main ROVER (teleoperation control)

1. Run the simulation: roslaunch main multi_rover_house.launch

   There are more worlds where to execute them are as follows:
   - roslaunch main multi_rover_empty.launch
   - roslaunch main multi_rover_world.launch

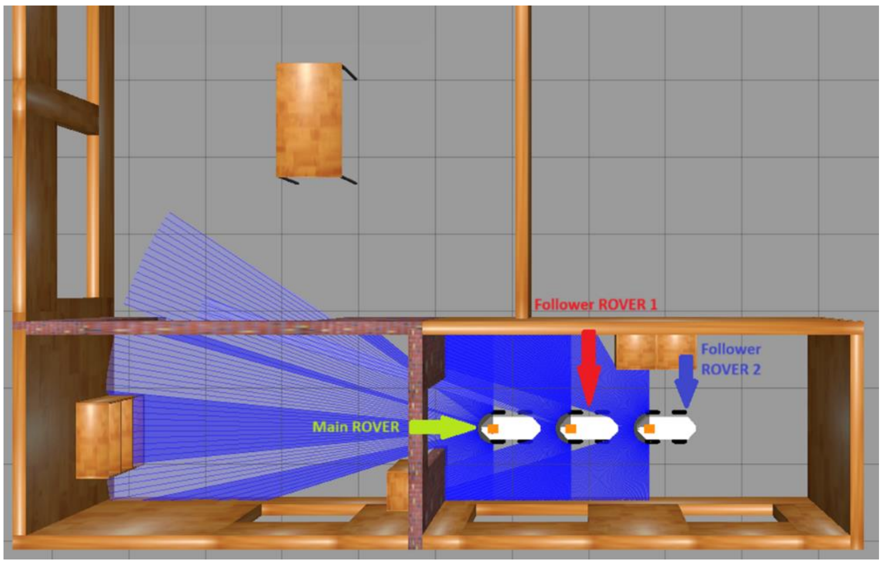


Figure 10 A main rover without autonomous navigation and 2 followers.

2. Run the follower algorithm for rover_1: rosrun follower_rover follower_2.py rover_1

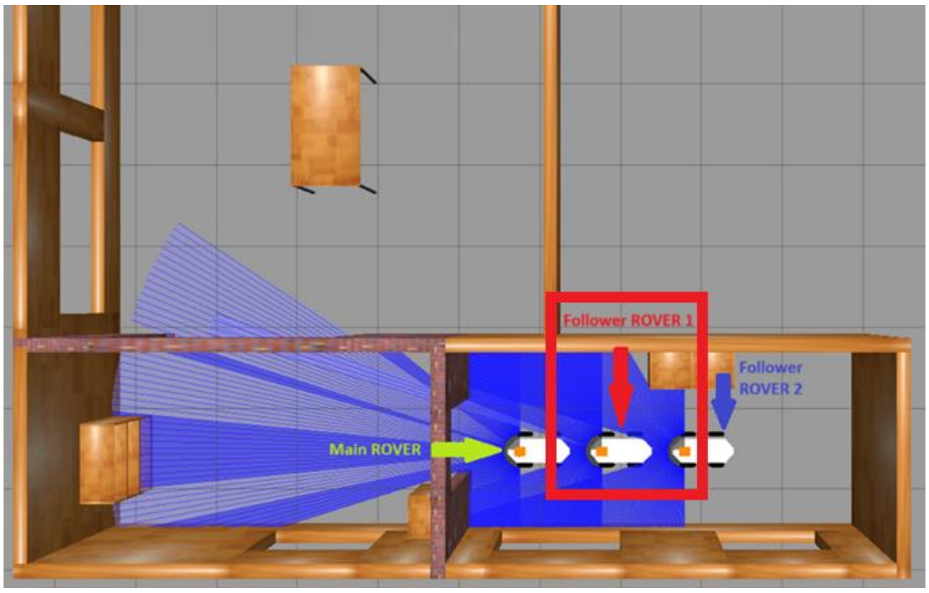


Figure 11 Follower ROVER 1.

3. Run the follower algorithm for rover_2: rosrun follower_rover follower_2.py rover_2

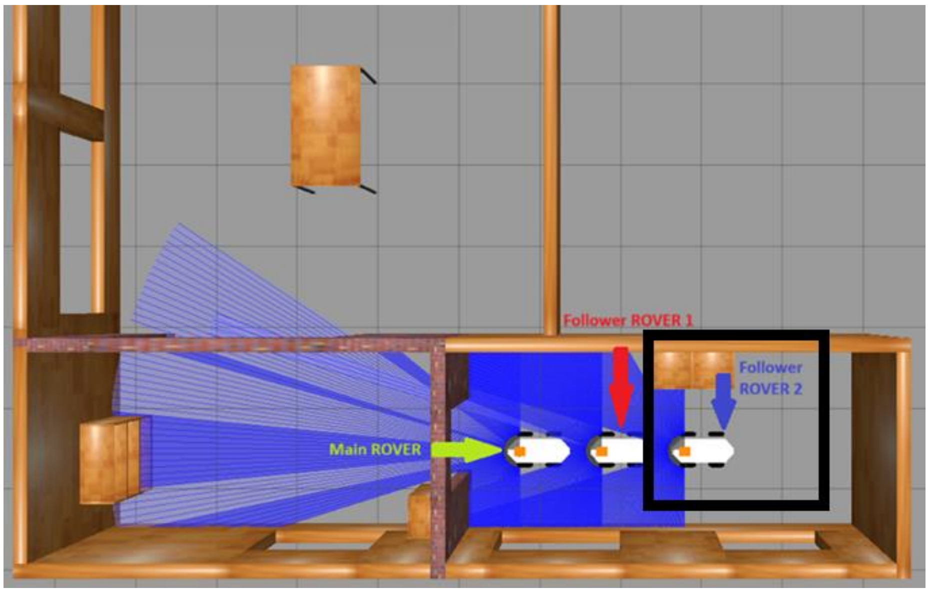
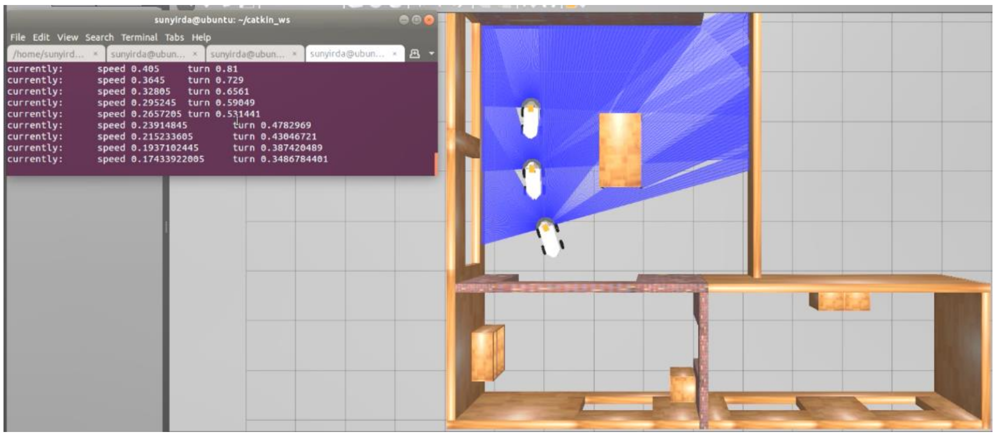


Figure 12 Follower ROVER 2.

4. Execute the teleoperation (example with the robot named rover_0): rosrun teleop_twist_keyboar teleop_twist_keyboard.py cmd_vel:=NAME_ROBOT/cmd_vel

Follower ROVER
Name: Iesus René Dávila Aguilar


sunyirda@ubuntu: ~/catkin_ws
File Edit View Search Terminal Tabs Help
/home/sunyird...
sunyirda@ubun...
sunyirda@ubun...
sunyirda@ubun...

```
currently:      speed 0.405     turn 0.81
currently:      speed 0.3645    turn 0.729
currently:      speed 0.32805   turn 0.6561
currently:      speed 0.295245  turn 0.59049
currently:      speed 0.2657205 turn 0.531441
currently:      speed 0.23914845        turn 0.4782969
currently:      speed 0.215233605       turn 0.43046721
currently:      speed 0.1937102445      turn 0.387420489
currently:      speed 0.17433922005     turn 0.3486784401
```

Figure 13 rover_0 teleoperation.

Operation for 2 ROVER followers and 1 main ROVER (autonomous navigation)

1. Run the simulation: roslaunch main multi_rover_house.launch navigation:=true

   There are more worlds where to execute them are as follows:
   - roslaunch main multi_rover_empty.launch navigation:=true
   - roslaunch main multi_rover_world.launch navigation:=true

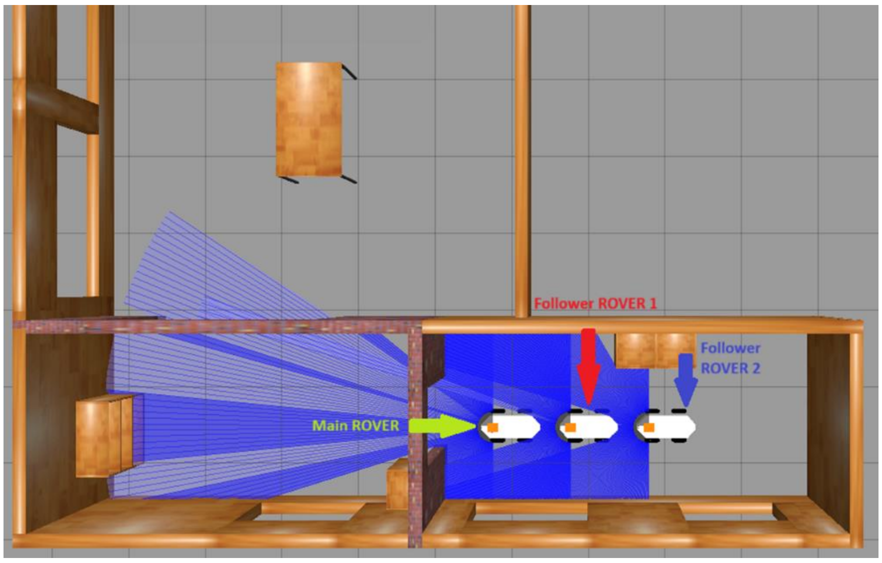


Figure 14 A main rover with autonomous navigation and 2 followers.

2. 2. Run the follower algorithm for rover_1: rosrun follower_rover follower_2.py rover_1

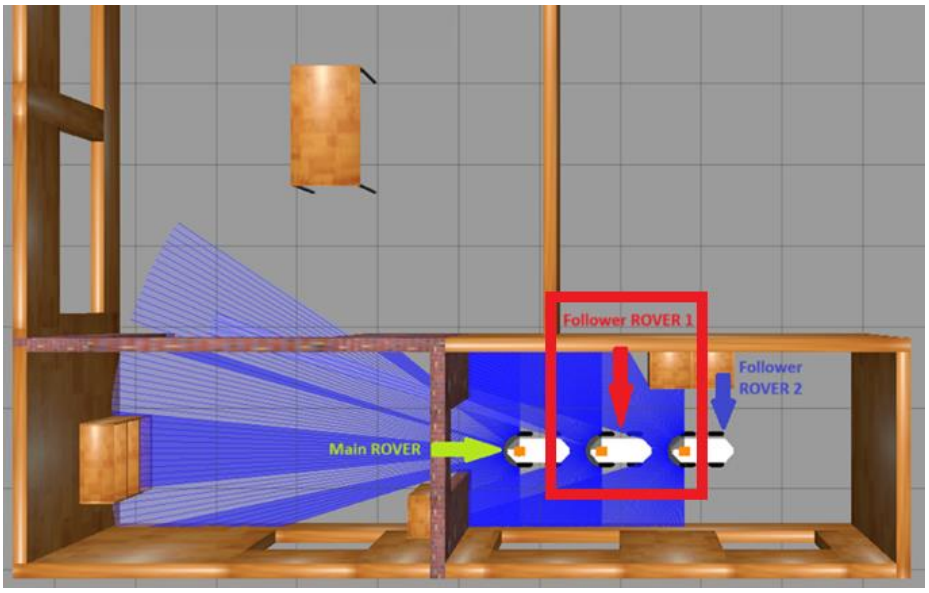


Figure 15 Follower ROVER 1.

3. Run the follower algorithm for rover_2: rosrun follower_rover follower_2.py rover_2

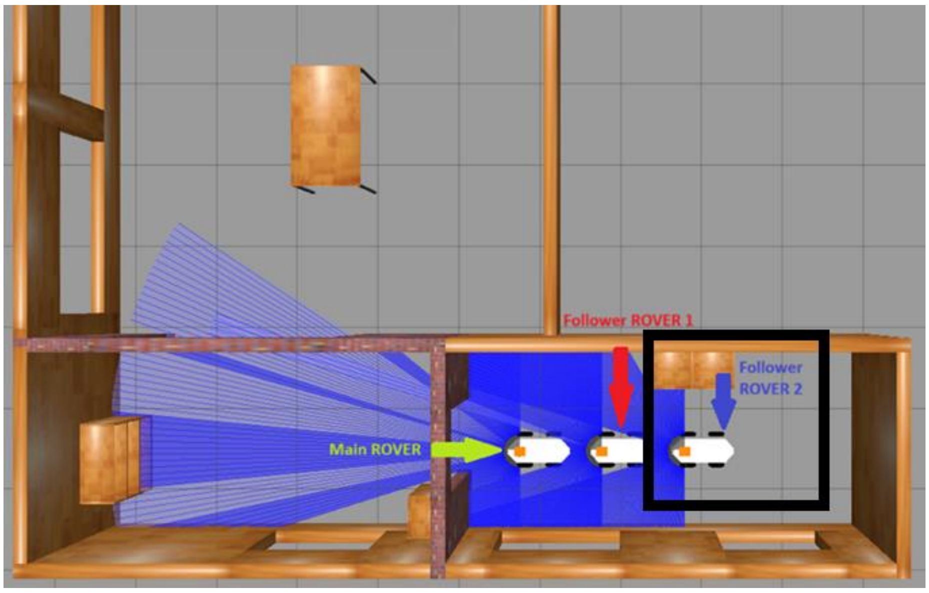


Figure 16 Follower ROVER 2.

Operation for real life execution

1. Run everything necessary for the main ROVER. The algorithm is independent of what the main ROVER executes and how it executes it.
2. Run everything necessary for the ROVER follower.
3. Run the follower algorithm for the corresponding ROVER: rosrun follower_rover follower_2.py ROVER_NAME
4. If you want to run the code on more than 1 robot, repeat steps 2 and 3 for the other followers.