Assignment 3 – Support

Updated: 11 Oct 2021

Table of Contents

Support Package Installation

one-off installation of ROS dependecies

```
sudo apt-get install ros- $ROS_DISTRO-stage-ros ros- $ROS_DISTRO-rosdoc-lite ros- $ROS_DISTRO-map-server
```

symbolically link the folder support folder to your catkin_ws

```
cd ~/catkin_ws/src
ln -s <your_git_repo>/skeleton/pfms_support
```

What is ln -s <origin> <destination>

A symbolic link, is a term for any file that contains a reference to another file or directory in the form of an absolute or relative path and that affects pathname resolution.
When the destination is not specified, the current location and existing name is used

To build the support package

```
cd ~/catkin_ws
catkin_make
```

What is catkin_make

catkin_make is a convenience tool for building code in a catkin workspace. catkin_make follows the standard layout of a catkin workspace.

You can only execute catkin_make from within the catkin_ws directory.

Given the many dependencies between packages in ROS and other components of the system, catkin_make resolves the dependencies and performs the build of componenets / messages / services in strategic order. The catkin worskapce has specific setup where source files are in **src** directory with build / install and devel also used.

Running simulation

If you have not yet run **roscore**

```
roscore
```

What is roscore?

roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate. It is launched using the **roscore** command. You should run it in a seperate terminal.

Now launch all of the components needed for the simulation

```
roslaunch project_setup project_setup.launch
```

What are we launching

The launch file is located in the project_setup/launch folder and is named project_setup.launch. In total 2 components are being launched:

- the simulator (**stage_ros**) with a world file that is the environment the robot is being simulated in
- the **local_map** builder which uses the laser to build a occupancy grid map as the robot moves about in the environment. The local map is 20x20 m and each cell in map is 0.1m

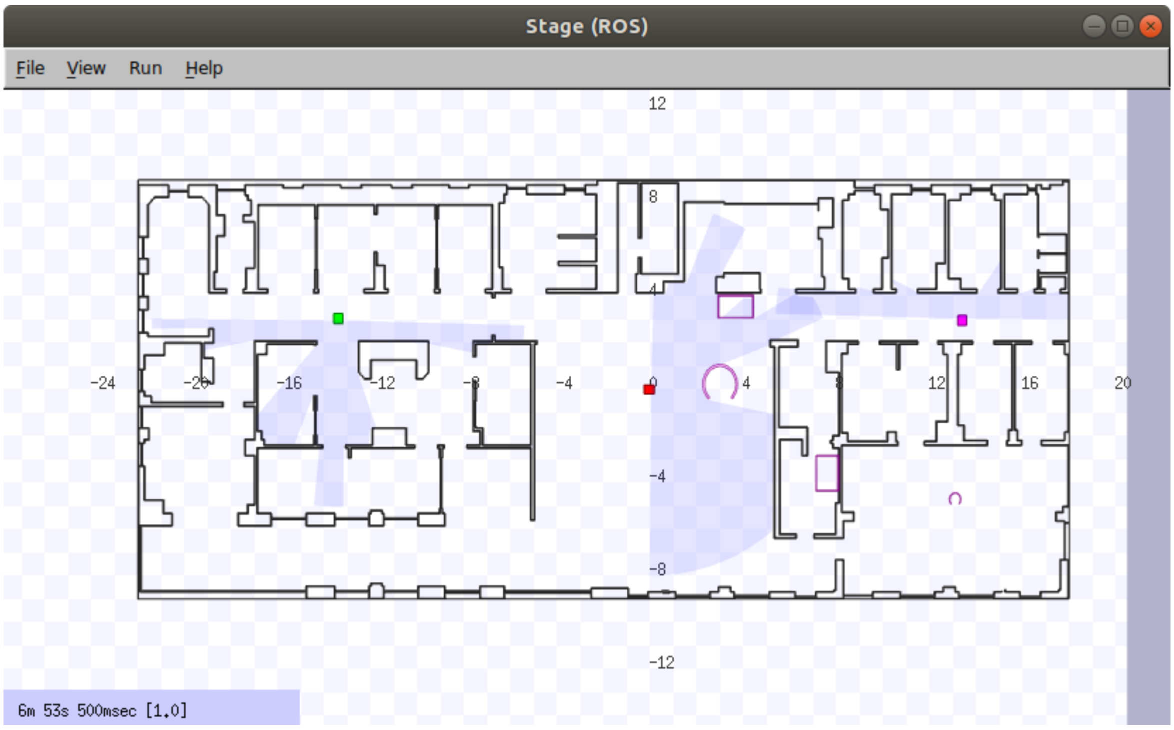
---

What is roslaunch?

roslaunch is a tool for easily launching multiple ROS nodes locally and remotely via SSH, as well as setting parameters on the Parameter Server. It includes options to automatically respawn processes that have already died. roslaunch takes in one or more XML configuration files (with the .launch extension) that specify the parameters to set and nodes to launch, as well as the machines that they should be run on..roslaunch will start roscore automatically if not already running.

Stage

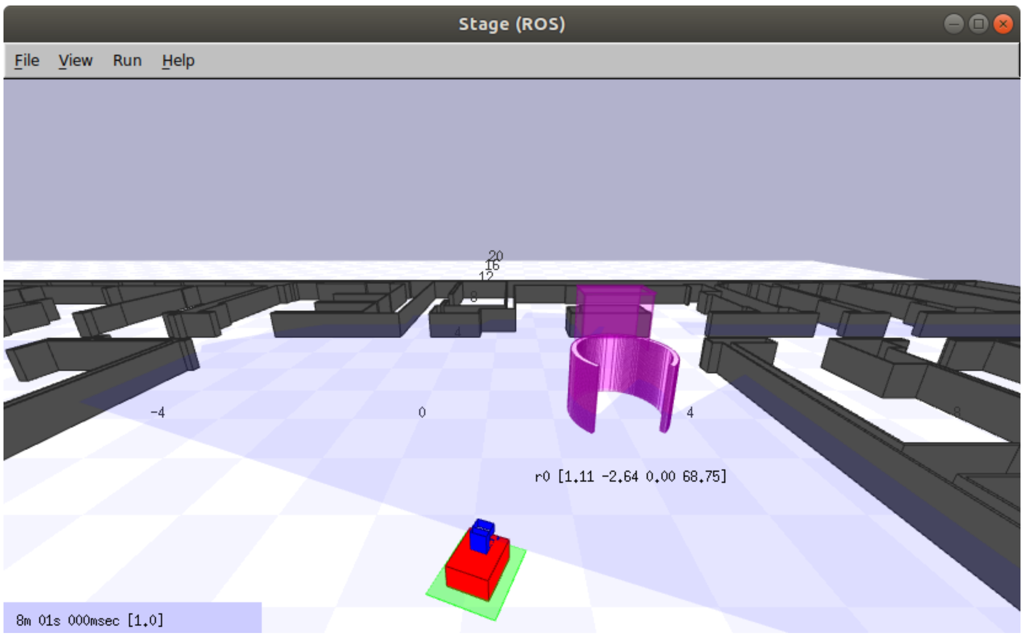
Stage is the simulator used
Pressing D - shows data (blue is laser data)
Pressing R - shows the perspective of camera view



You could ask: why do we need an image to run stage, is that kind of pointless as we already known the whole environment

The image of the environment is loaded into stage and thereafter stage only provides raw readings as if a robot equipped with sensor was placed in this environment (raw odometry, raw laser). We assume the odometry is PERFECT, and can be used for localisation, to simplify the task. Students are well aware this is not the fact (remember Mechatornics 2!)

What does the simulation provide



The simulation provides a number of topics (rostopic list) shows them.

To view type rostopic info <topic_name> (example: rostopic info /robot_0/odom)

To view information on message rosmsg info <message_name>

(Example :rosmsg info nav_msgs/Odometry message)

The ones to be used for the assignment are related to robots (there are 3) and the local_map, examples below:

rostopic : /robot_0/odom
Robot 0 Position (in world coordinates – reference is centre of map)

rostopic: /robot_0/base_scan
Laser Rangefinder located at centre of robot 0
The high intensity returns belong to cabinets in the environment.

rostopic: /robot_0/cmd_vel
Linear and angular velocity to control robot_0

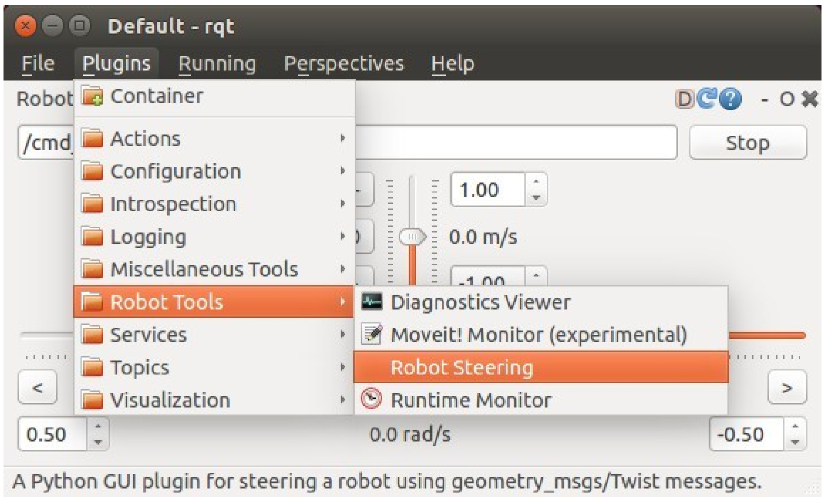
rostopic: /local_map/local_map
Occupancy Grid map built around the robot (in local coordinates)
The local map is encoded as an , (the height,wisth and resolution can be obtained from the info MetaData. The occupancy per cell in in the map data, in row-major order, starting with (0,0).
Occupancy # probabilities are in the range [0,100]. Unknown is -1

Robot Steering Control

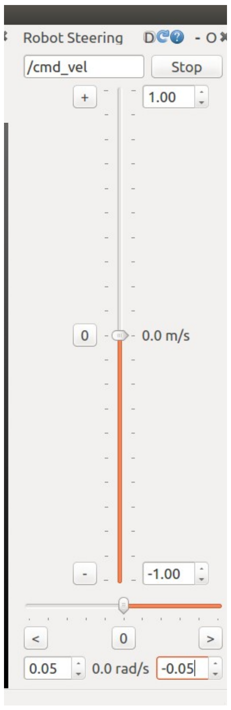
Either using rqt (Robot Steering GUI) or keyboard

Rqt

rqt



A Python GUI plugin for steering a robot using geometry_msgs/Twist messages.



Vertical lever is for linear and horizontal angular velocity. The textboxes are max value. Strongly suggest to change MAX angular velocity to a smaller value.

Keyboard control

A sample keyboard controller **TBK** is provided in **a4_setup** package
rosrun project_setup TBK cmd_vel:=/robot_0/cmd_vel

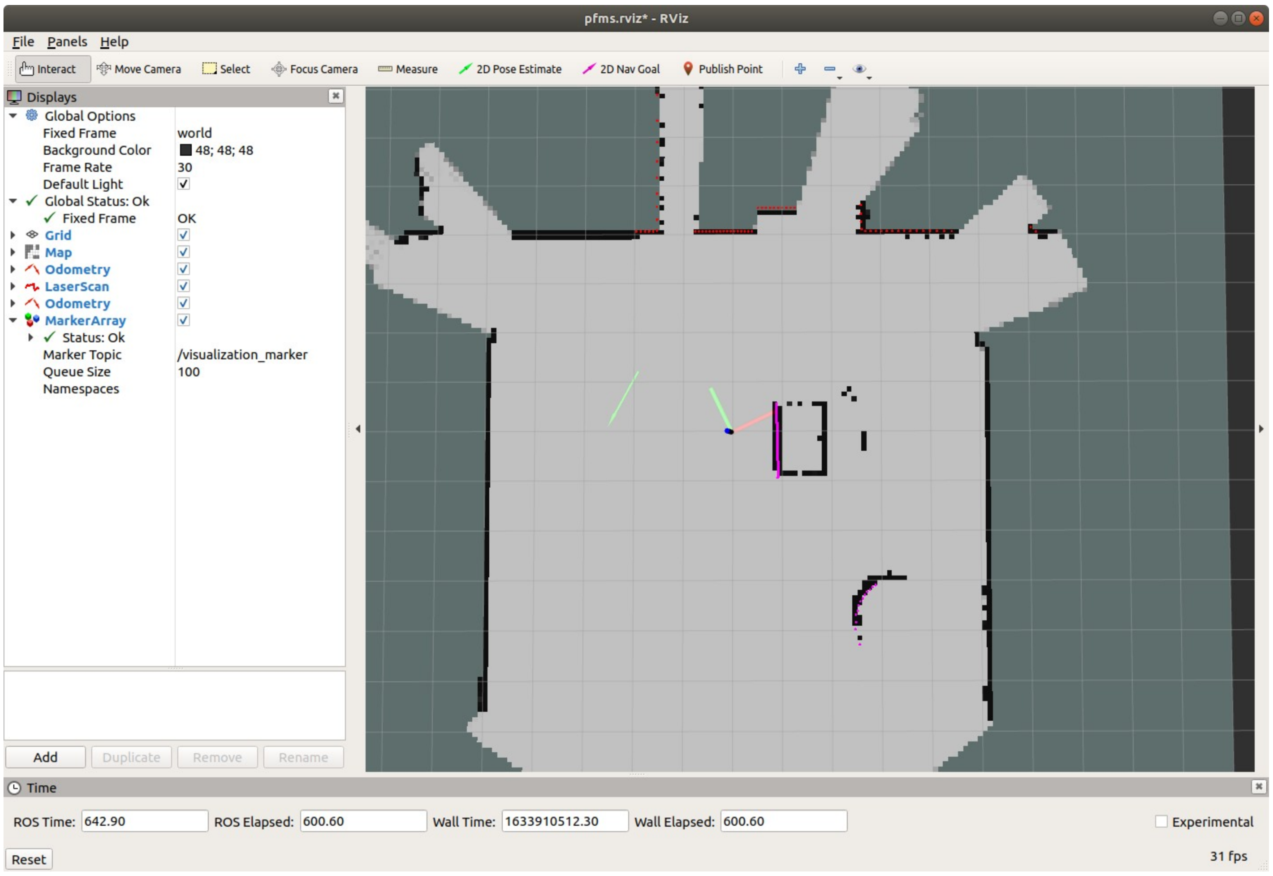
Visualiser

Optionally, we will use rqt for visualising

```
rviz -d $(rospack find project_setup)/rviz/pfms.rviz
```

The laser is noted in RED, the laser returns that are of high intensity are purple (cabinets are of high intensity – dome cabinets are not high intensity if you are inside them).
The robot_0 pose is an axis arrow (arrow head points in local x reference frame). The pose for robot_1 is noted by a green arrow. The occupancy grid map is shown with cell values (white free space, black occupied and grey unknown).

Below is an example of what the visualiser will show when moving robots about.


pfms.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Add Duplicate Remove Rename
Time
Reset
31 fps

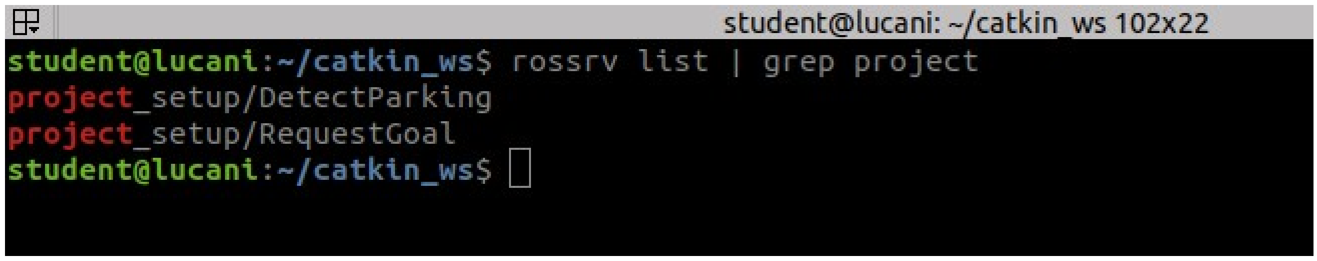
```
Global Options
  Fixed Frame          world
  Background Color     48; 48; 48
  Frame Rate           30
  Default Light        ✓
✓ Global Status: Ok
  ✓ Fixed Frame        OK
Grid                   ✓
Map                    ✓
Odometry               ✓
LaserScan              ✓
Odometry               ✓
MarkerArray            ✓
  ✓ Status: Ok
  Marker Topic         /visualization_marker
  Queue Size           100
  Namespaces
```

ROS Time: 642.90 ROS Elapsed: 600.60 Wall Time: 1633910512.30 Wall Elapsed: 600.60 Experimental

Calling Services

For the purpose of this assignment we have created additional services.

To examine the service message rossrv info <service_msg>, example below



```
student@lucani: ~/catkin_ws 102x22
student@lucani:~/catkin_ws$ rossrv list | grep project
project_setup/DetectParking
project_setup/RequestGoal
student@lucani:~/catkin_ws$ 
```

Once a node has been activated to accept incoming services of this type on the <incoming_service_name>, calling the service can be done from command line: rosservice call <incoming_service_name>

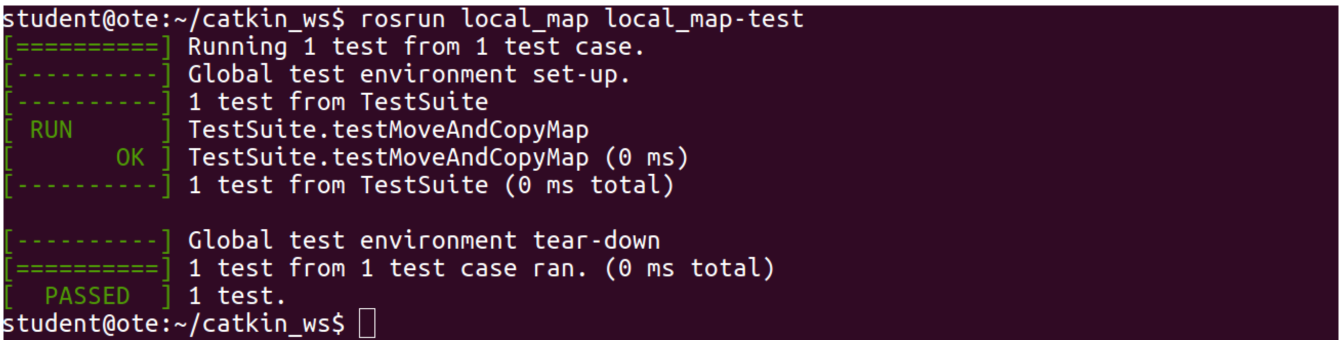
Unit Testing – Gtest

We will provide an example of gtest in week11.

Compiling the test (NOTICE the ‘s’ at the end : **tests, THIS IS IMPORTANT**) catkin_make tests

Running the test
rosrun <package_name> <unit_test_name>

Documentation



```
student@ote:~/catkin_ws$ rosrun local_map local_map-test
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from TestSuite
[ RUN      ] TestSuite.testMoveAndCopyMap
[       OK ] TestSuite.testMoveAndCopyMap (0 ms)
[----------] 1 test from TestSuite (0 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (0 ms total)
[  PASSED  ] 1 test.
student@ote:~/catkin_ws$ 
```

The recommended way to create documentation is ROS is

via ros_doclite rosdoc_lite <package_path>

ie
cd ~/catkin_ws
rosdoc_lite src/<package_name>

will generate the doc in

~/catkin_ws/doc To open the

documentation
firefox ~/catkin_ws/doc/html/index.html

Path Follower

A Path Follower has been supplied as an executable file (tested on Ubuntu 18.04 ROS Melodic and 20.04 ROS Noetic). A video of the installation and process of using the executable is online under Assessment3.

To install we need to copy your local _setup_util.py file into the a3_pf_install folder

```
cd <your_git_repo_folder>
cd skeleton/a3_pf_install
cp ~/catkin_ws/devel/_setup_util.py .
```

Then, source the setup.bash by adding the following line to your .bashrc

```
source <full_path_to_git>/skeleton/a3_pf_install/setup.bash
```

Once you open a new terminal you can run the path_following via

```
rosrun pfms_path_following pfms_path_following-velocity_control
```

You can supply a 2D Nav Goal for the robot to go to via rviz.
The path is done in 3 steps (1) turn to face that position it needs to go to (2) go straight until reaching position (3) turn to face the orinetation of the goal.