# Support Packages

***All users***

Unzip **a3\_help.zip,** copy the folders to your catkin\_ws/src

***FEIT PCs***

Unzip **local\_map.zip,** copy the folders to your catkin\_ws/src

Unzip **map\_ray\_caster.zip,** copy the folders to your catkin\_ws/src

Unzip **lama\_msgs.zip,** copy the folders to your catkin\_ws/src

***Standalone PCs one-off installation***

sudo apt-get install ros-indigo-local-map

# Running simulation

Run the **stage** simulation as:

rosrun stage\_ros stageros $(rospack find a3\_help)/worlds/uoa\_robotics\_lab.world

This will open Stage.

Pressing D - shows data (blue is laser data)

Pressing R - shows the perspective of camera view

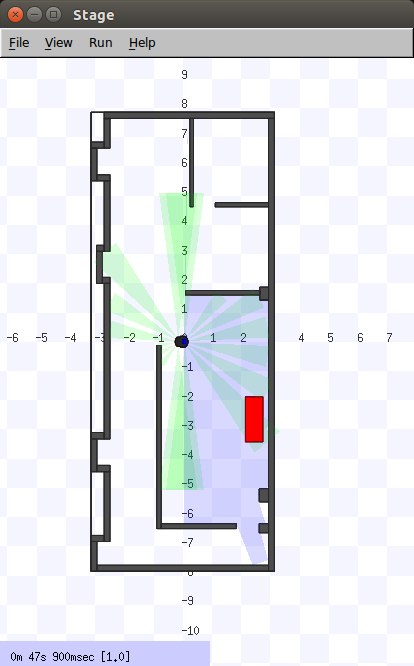


Figure 0- Stage

# Robot Steering Control

Either using rqt (Robot Steering GUI)

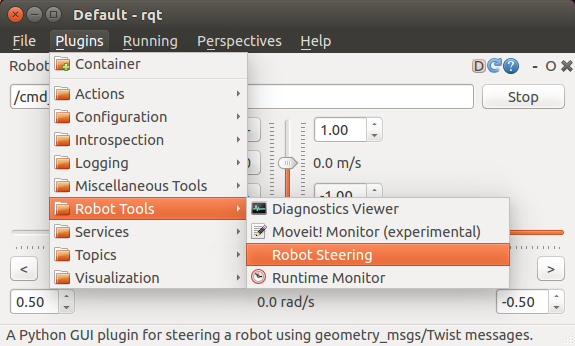


Figure 0- RQT

A sample keyboard controller **TBK** is provided in **a3\_help** package

*rosrun a3\_help TBK*

# OgMap Creation

A sample node **local\_map** is provided

rosrun local\_map local\_map /local\_map/scan:=/base\_scan\_1 \_map\_width:=300  \_map\_height:=300 \_map\_resolution=0.1

If working correctly should state:

[ INFO] [1496387191.945375296, 33.900000000]: **Found world frame odom**

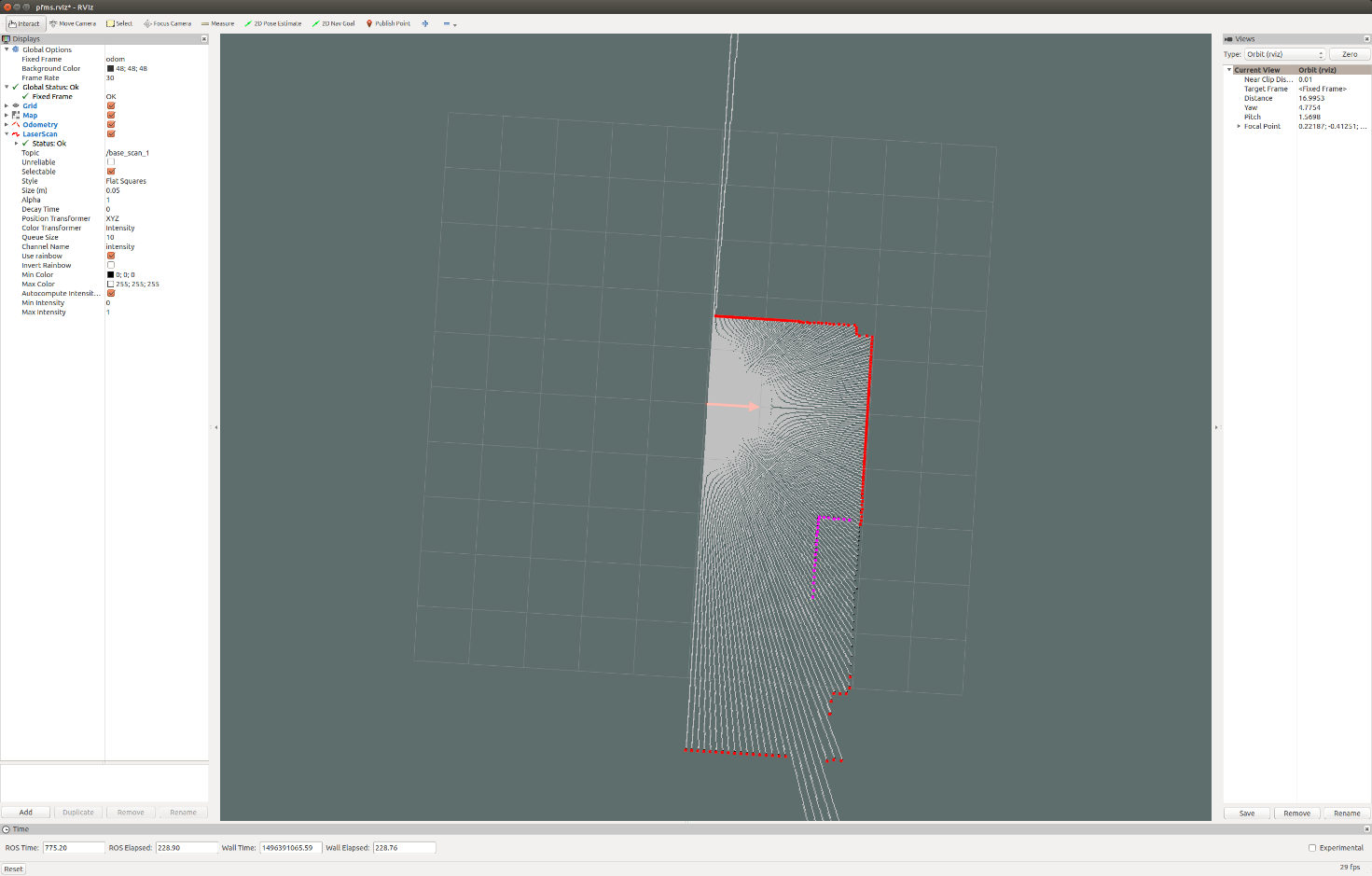
Any other world frame please CTRL+C and restart component

# Visualiser

We will use rqt for visualising

rosrun rviz rviz -d $(rospack find a3\_help)/rviz/pfms.rviz

The laser is noted in RED, the robot pose is an arrow



# OgMap Image Creation

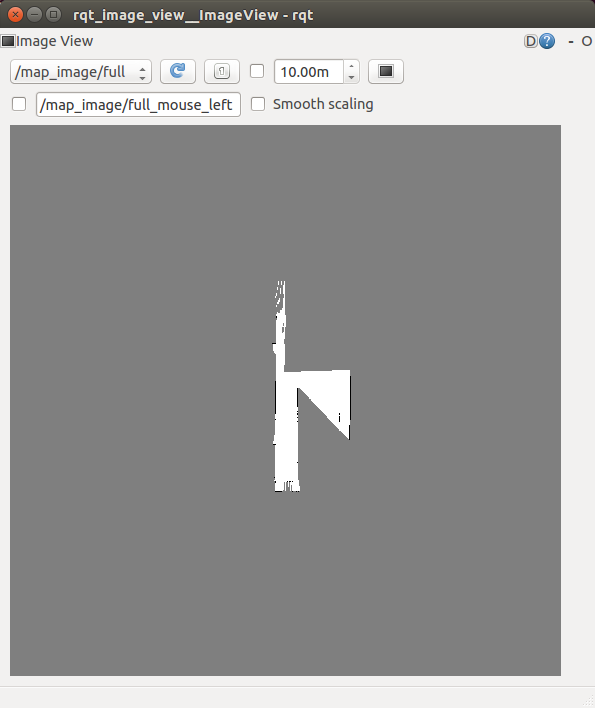
The local\_map node produces an OccupancyGrid structure, used in ROS. To produce OpenCV images we have provides a separate node.

rosrun a3\_help map\_to\_image\_node /map:=/local\_map/local\_map /pose:=/odom

OUTPUT

[ INFO] [1496392667.559820901]: Map to Image node started.

The map can be visualised in rqt\_image\_view



# Sample Code

We have provided sample code that subscribes to

* Robot Position’
* Laser Rangefinder
* OgMap

And provides

* Output Image
* Service (request\_goal)

The code launches all the callbacks for data and a separate thread for processing, protects data via mutexes (via a Data Structure) and performs some mundane processing of the OgMap image, which it displays out.

rosrun a3\_help sample

Calling the service can be done from command line:

rosservice call /request\_goal 1 1

# Unit Testing – Gtest

The test is in <package>/test and the CMakeLists.txt contains the setup for it (last few lines in the file). The example test draws a line through the image (top left to bottom right corner) and evaluate that the pixel in the middle is set to 255.

Compiling the test

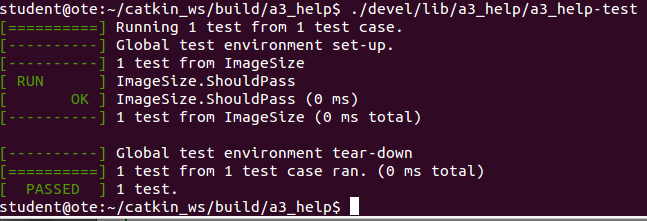
*catkin\_make tests*

Running the test

*./devel/libs/a3\_help/a3\_help-test*

More information is available on link

*http://ysonggit.github.io/coding/2014/12/19/use-gtest-in-ros-program.html*



# Documentation

The recommended way to create documentation is ROS is via ros\_doclite (<http://wiki.ros.org/rosdoc_lite>)

rosdoc\_lite <package\_path>

ie

cd ~/catkin\_ws

rosdoc\_lite src/a3\_help

will generate the doc in ~/catkin\_ws/doc

To open the documentation

firefox ~/catkin\_ws/doc/html/index.html