Assignment 6

The purpose of this assignment is to get acquainted with the *navigation stack*. The navigation stack serves to drive a mobile base autonomously from one location to another while safely avoiding obstacles.

# Learning goal

This assignment is primarily focussed on the following learning goal:

* apply ‘Probablistic Robotics’ concepts like ‘Occupancy Grid Map’, ‘Bayes (Kalman, Particle) Filter’, and ‘SLAM’ by making a mobile robot drive autonomously, using existing ROS localization, mapping & navigation nodes

# Getting ready

We assume you are already acquainted with the Stage simulator.

Some startup code is provided for you on GitHub. To download this code you will have to install the [GIT](http://git-scm.com/) version control software:

* *sudo apt-get install git*

In order to perform the navigation experiments in this assignment you need to install the simulation environment that we prepared in your ROS workspace src directory. Assuming your workspace is ~/catkin\_ws the installation goes as follows:

* *cd ~/catkin\_ws/src*
* *git clone --depth 1* [*https://github.com/fhict/mines-ros.git*](https://github.com/fhict/mines-ros.git)
* *cd ~/catkin\_ws*
* *catkin\_make*

Also take care that the navigation stack and associated mapping and localization tools are installed:

* *sudo apt-get install ros-indigo-gmapping*
* *sudo apt-get install ros-indigo-amcl*
* *sudo apt-get install ros-indigo-move-base*
* *sudo apt-get install ros-indigo-map-server*

To teleoperate your robot in the simulated world the *arbotix\_gui* is a nice tool to use. You can install it as follows:

* *sudo apt-get install ros-indigo-arbotix*

Finally, as a warming-up for your own experiments, watch the following tutorial video’s to get more feeling for the ROS navigation stack (<http://wiki.ros.org/navigation>), localization on a known map *AMCL* (Adaptive Monte Carlo Localization) and map building using *gmapping* (SLAM):

* ROS Navigation: <http://www.youtube.com/watch?v=qziUJcUDfBc>
* *TurtleBot Odometry Primer*: <http://www.youtube.com/watch?v=3S8MXsnNe3U>
* *TurleBot Localization Primer*: <http://www.youtube.com/watch?v=Mv1mbsMfbmI>
* *SLAM 1: Testing ROS with the gmapping package*: <http://www.youtube.com/watch?v=khSrWtB0Xik>

# The Tasks

You are supposed to run a robot in a simulated world, generate a map of the world and do navigation experiments.

## Generate a map

You will generate a map similar to what is described in the following tutorial:

* *SLAM Map Building with TurtleBot* <http://wiki.ros.org/turtlebot_navigation/Tutorials/Build%20a%20map%20with%20SLAM>

However, instead of a real robot in the real world we will be using a simulated one in a simple, simulated world.

In the following we assume you have installed our simulation environment, as described earlier on in the “Getting Ready” section.

We will be using 2 different (simulated) robots:

* a Roomba, equipped with a Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder
* a TurtleBot, basically a Roomba, equipped with a Kinect sensor

Your task is to generate 2 maps of the same world, each one with a different robot.

Let us start with the Roomba/Hokuyo robot and launch the simulated robot world, rviz visualization node, gmapping SLAM node, and move\_base navigation node as follows:

* *roslaunch stage\_navigation laser\_gmapping.launch*

Select the Stage world simulator window and make the laserbeams visible:

* *<shift>-D*
* In the Visualize window that pops-up select “*Ranger beams*”

In another terminal start the arbotix\_gui using the following command:

* *arbotix\_gui*

Now drive the robot around and watch gmapping building the map. In the map obstacles are drawn in black, free space is white and unknown territory is grey. Remember gmapping is using probabilistic reasoning: the state of the map is how gmapping thinks it is en the location of the robot is where it thinks it probably is. The Stage simulator windows shows how the world really looks like and where the robot really is.

Note that there are other ways you can also use to drive your robot around in the simulated world:

* Select the robot in Stage and drag it around (“lead it by the nose” as roboticist would say).
* Use the Navigation stack (i.e. the *move\_base* node, which also has been started by the launch file) to make the robot drive to the border of the known territory in the map. Just repeatedly give move\_base a new goal using the “*2D Nav Goal*” button in rviz.

Hints:

* To get a good map you have to drive around, turn in place, and also go back where you started, several times, because gmapping needs to recognize places it already visited before. Take care not to bump into walls because this could make gmapping lose track of the world.
* Look at the parameters of gmapping ( <http://wiki.ros.org/gmapping> ). Maybe you can improve your maps by changing one or more parameters in the gmapping launch file: *stage\_navigation/move\_base\_config/slam\_gmapping.xml*. See for instance what the parameters *maxUrange* and *minimumScore* can do for you.

If you are satisfied with the map you can save it using following command:

* *rosrun map\_server map\_saver -f laser\_world\_map*

Now do the same thing with the simulated Turtlebot, which is equiped with a Kinect sensor, used here as a sort of laser scanner (i.e. the depth image of the sensor is condensed into a laserscan message):

* *roslaunch stage\_navigation kinect\_gmapping.launch*

Again drive the robot around and, when you are satisfied, save the map:

* *rosrun map\_server map\_saver -f kinect\_world\_map*

Finally compare the results of both robots:

* Did you experience differences in mapbuilding using the two different robots?
  + If there was a difference, which one gave the best map?
  + If there was a difference, which one was fastest in map building?
* Explain the differences, if there were any.

## Do autonomous navigation experiments in the Map

You will use the maps that we generated before to do navigation experiments.

First study the ROS *navigation stack* (<http://www.ros.org/wiki/navigation>) and its tutorials. In particular the following tutorials are worth studying in detail:

* *Setup and Configuration of the Navigation Stack on a Robot*  
  <http://wiki.ros.org/navigation/Tutorials/RobotSetup>
* *Using rviz with the Navigation Stack*  
  <http://wiki.ros.org/navigation/Tutorials/Using%20rviz%20with%20the%20Navigation%20Stack>  
  This will allow you to set initial 2D pose estimate and set 2D nav goals using the rviz GUI.

The experiments we will do are similar to the one outlined in the tutorial:

* *Autonomous Navigation of a Known Map with TurtleBot*<http://wiki.ros.org/turtlebot_navigation/Tutorials/Autonomously%20navigate%20in%20a%20known%20map>

However, we will use a simulated robot in a simulated word.

Let us start with the Roomba/Hokuyo robot and launch the simulated robot world, rviz visualization node, the AMCL (Adaptive Monte Carlo Localization) node, and move\_base navigation node as follows:

* *roslaunch stage\_navigation laser\_amcl.launch map\_file:=laser\_world\_map.yaml*

Select the Stage world simulator window and make the laserbeams visible:

* *<shift>D*
* In the Visualize window that pops-up select “*Ranger beams*”

Now use the “*2D Nav Goal*” button in rviz to give the robot some interesting goals:

* Send the robot to the upper left corner of the map
* When arrived, send the robot to the upper right corner
* When arrived, send the robot to the lower left corner
* When arrived, send the robot to the lower right corner
* When arrived, send the robot to the upper left corner

Note: rviz does not use actionlib (it just publishes the goal on the topic *move\_base\_simple/goal*) and therefor does not know when the goal is reached.

Observe what happens. Time how long it takes. Make screendumps of interesting things

Now do the same experiment with a different robot, the TurtleBot/Kinect robot, but use the same map:

* *roslaunch stage\_navigation kinect\_amcl.launch map\_file:=laser\_world\_map.yaml*

Again observe what is happening and make interesting screendumps.

## Write navigator node

Last task is write a node to submit a sequence of goals to the navigation stack.

How to do that is described here:

* *Sending Goals to the Navigation Stack*

<http://wiki.ros.org/navigation/Tutorials/SendingSimpleGoals>  
This will allow you to send navigation goals and wait for completion using the actionlib library.

Create a new package “*assignment6*”.

Put the maps that you generated earlier in the subdirectory “*maps”* of this package.

Write a ROS node called “*navigator*”. This node must use actionlib to send a sequence of navigation goals to the navigation stack (i.e. the move\_base node). For each goal it should wait until the goal is reached. The sequence of goals is the same as described in the manual experiment we did before where we issued goals using the 2D Nav Goal button of rviz.

Of course provide a launch file “*assignment6.launch*”. Include the laser\_amcl.launch file from the stage\_navigation package.

# What to Submit

Your submission for this assignment will have two parts:

1. A zip file containing your package.
2. A document which explains how everything works and describes how to run it. The document should also describe your investigations. Include screendumps for proof.