# Laboratory class #4: ROS Navigation

## 1. Navigation stack in ROS

In the laboratory class 2 we implemented the low level controller for a fully autonomous navigation system. In this lab we are going to integrate it with the main tool that ROS offers for navigation systems, the Navigation Stack. This stack defines the architecture. It includes localization and map management, as well as the different planning layers. The architecture is described in Figure 1.

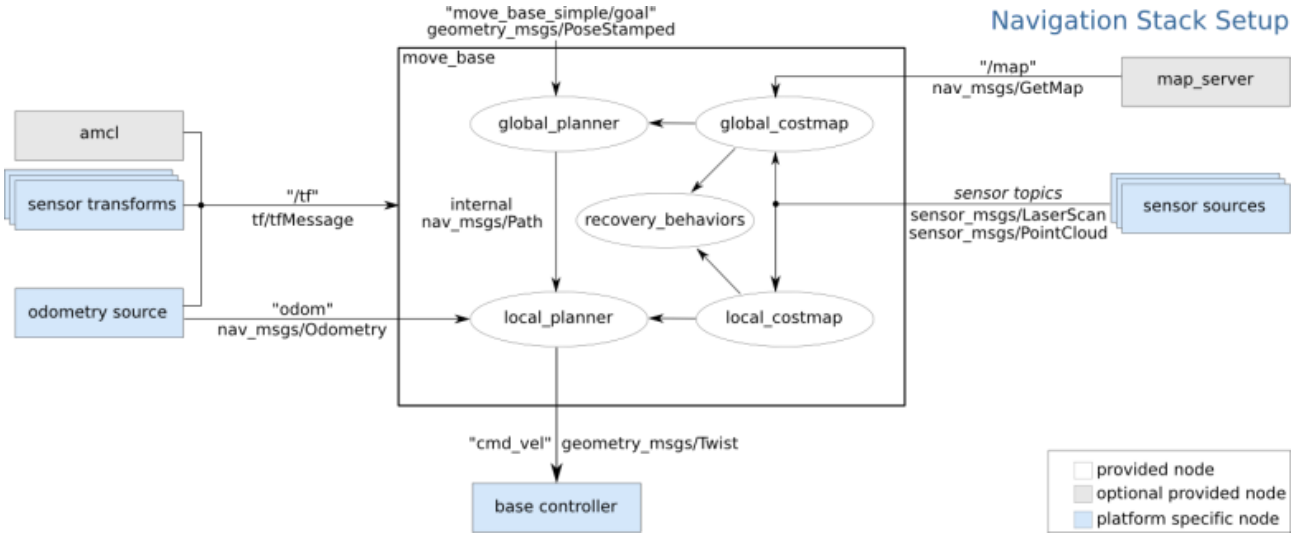


Figure 1. Navigation stack

### 1.1. Map_server

This node (http://wiki.ros.org/map_server) offers the map information. It requires an image file with the map and a configuration file with the metric information, including the size of each pixel and the location of the origin.

### 1.2. AMCL

This node (http://wiki.ros.org/amcl) is in charge of the robot localization using laser measurements and a map, implementing an Adaptive Monte Carlo Localization algorithm. It publishes the estimation of the robot (base_link frame) with respect to the map frame in the topic amcl_pose.

### 1.2. Move_base

This represents the core of the navigation stack as it handles all the planning layers, including the reactive navigation and high-level planner (http://wiki.ros.org/move_base). Given a goal, read from the topic /move_base_simple/goal it computes and publishes the command velocities (cmd_vel) to drive the robot towards the goal in the map. To compute the velocities, pose feedback from the AMCL, odometry data, and information from the robot sensors are needed. This node is highly configurable, since it enables the use of different global and local planners.

The ROS package needed to install the Navigation Stack is `navigation`. Open a terminal and type the following command:

```
>$ sudo apt install ros-noetic-navigation
```

*Exercise 1.*

Download arob_lab4 package from github https://github.com/luisriazuelo/pXX_arob_lab4.git and include it into your workspace.

After downloading the repository, you need to modify the name of the package folder (pXX_arob_lab4) and the contents of several files according to the pair number. Change pXX prefix, where pXX corresponds to the assigned pair number (e.g. p00, p01 .....) for this course. The assigned pair number can be found in the Moodle document "Laboratories Pairs".

To change the prefix pXX of all files automatically, you can use the following command in the terminal (you should replace the text string p?? with the corresponding pair number). You need to go to the parent directory of the package "pXX_arob_lab4" and execute:

```
>$ find . -type f -not -path '*/\.*' -exec sed -i 's/pXX/p??/g' {} +
```

Build the package and run the launch file arob-p4-navigation-basic.launch file. Notice that the first line of this file sets the parameter `use_sim_time` to true. This is to make sure all nodes running in our system use the simulation time published in the `/clock` topic by Stage simulator.

In another terminal publish a message in the topic /move_base_simple/goal and see how the robot moves to the goal:

```
>$ rostopic pub /move_base_simple/goal geometry_msgs/PoseStamped '{ header: { frame_id: "map" }, pose: { position: { x: -3, y: 0 }, orientation: { x: 0, y: 0, z: 0, w: 1 } } }'
```

## 2. RViz

RViz is a 3D visualization tool for ROS that subscribes to topics and visualizes the message contents using a series of plugins. On the right window is possible to select different views while from the top menu is possible to publish user information as for example target points. RViz permits to save and load setup as RViz configuration and has many plugins to facilitate the visualization (added from the left window). In order to install and execute RViz, execute the following commands:

```
>$ sudo apt install ros-noetic-rviz
```

```
>$ rosrun rviz rviz
```

If you want to add a new visualization, on the bottom left you click the 'Add' button and then you get a display window with the possible options. It is important to select an existing frame in the global view, for example odom.

*Exercise 2.*

Close everything and launch the file arob-p4-navigation-rviz.launch that includes an instance of rviz. Observe the value of the different topics sending different goals to the robot.

*Exercise 3.*

Now you have all the tools to analyze how some parameters can affect robot navigation. In particular, modify the value of the following parameters in the launch and configuration files:

- Try different global and local planners (planner_selection.yaml)
- Number of particles in AMCL

- Use the dynamic obstacle to make it difficult for the robot!

## 3. rosbag command-line tool

This is a set of tools for recording from and playing back to ROS topics. The tool rosbag subscribes to one or more ROS topics, and stores the serialized message data in a file called bag. A bag is a file format in ROS for storing message data. These bag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics.

The following tools can be use with rosbag:

- record: Record a bag file with the contents of specified topics
- play: Play back the contents of one or more bag files in a time-synchronized fashion.

### 3.1. rosbag record

Open a terminal and type the following command in order to see all the options provided by rosbag record:

```
>$ rosbag record -h
```

All available options for saving the information provided by ROS will be displayed on the screen. In order to store all the topics published you have to execute the following command:

```
>$ rosbag record -a
```

Press ctrl+c for stopping the execution and a file whose name is the time stamped with an extension .bag will be created. In order to see the información contained on that file, execute the following command:

```
>$ rosbag info name_of_the_file.bag
```

### 3.2. rosbag play

Open a terminal and type the following command in order to see all the options provided by rosbag play:

```
>$ rosbag play -h
```

All available options for playing the information provided by ROS will be displayed on the screen. In order to reproduce the information of all the topics stored, you have to execute the following command:

```
>$ rosbag play name_of_the_file.bag
```

It will automatically publish all the topic information in a time-synchronized fashion.

## 4. Integration with the navigation stack

It is possible to program your own global and local planners to be compatible with move_base (see http://wiki.ros.org/nav_core). The package `nav_core` provides the interfaces to implement your own algorithms for the global planner, local planner and recovery behavior. Additionally, and in order to be used in the `move_base` node, the new algorithms should be registered and exported as plugins in the ROS system (see Pluginlib (http://wiki.ros.org/pluginlib) package documentation).

In order to see a list of current local planners, global planners and recovery behaviors using the interfaces provided by `nav_core`, type in the terminal:

```
>$ rospack plugins --attrib=plugin nav_core
```

In this section we will integrate the low level controller implemented in lab2 as a plugin to be used in the move_base node as a local planner. A new class named LLCLocalPlanner is defined to implement the interface definition provided by nav_core for the local planner.

### *Exercise 4.*

Explore the files llc_local_planner.h (in folder include/arob_lab4) and llc_local_planner.cpp (in folder src) to understand the functions defined.

In order to use this new local planner in the navigation stack, we should register it as a plugin and export it in the ROS system. Follow the steps:

- Register and export the plugin.

  Include the following line after the include section in llc_local_planner.cpp file:

  ```
  PLUGINLIB_EXPORT_CLASS(pXX_llc_local_planner::LLCLocalPlanner,
  nav_core::BaseLocalPlanner)
  ```

  Change the previous pXX prefix, where pXX corresponds to the assigned pair number (e.g. p00, p01 .....) for this course.

- Pluginlib description file.

  Create a file named llc_local_planner_plugin.xml in the parent directory of the package, and include the following:

  ```
  <library path="lib/libpXX_llc_local_planner_lib">
        <class name="pXX_llc_local_planner/LLCLocalPlanner"
               type="pXX_llc_local_planner::LLCLocalPlanner"
               base_class_type="nav_core::BaseLocalPlanner">
               <description>This is a local planner plugin.</description>
        </class>
  </library>
  ```

  Change the previous pXX prefix, where pXX corresponds to the assigned pair number (e.g. p00, p01 .....) for this course.

- Register the plugin with the ROS package system.

  In the package.xml file you should include the following:

  ```
  <export>
        <nav_core plugin="${prefix}/llc_local_planner_plugin.xml" />
  </export>
  ```

  Given that the package providing the plugin (pXX_arob_lab4) depends on the package containing the plugin interface (nav_core), we should also include `<depend>nav_core</depend>` in the package.xml.

Build the package again. Now, if you type the command in the terminal:

```
>$ rospack plugins --attrib=plugin nav_core
```

you should see your plugin in the list.

*Exercise 5.*

Complete functions computeVelocityCommands() and isGoalReached() in llc_local_planner.cpp file to implement your low level controller. Do not forget to include the parameter values for your low level controller, which are defined in llc_local_planner_params.yaml.

*Exercise 6.*

Launch file arob-p4-navigation-plugin.launch with llc_local_planner as the local planner (planner_selection.yaml), and send different goals to evaluate the robot behavior.

*Exercise 7.*

The following exercise consists of testing the code implemented in the previous exercise on a real platform. To do this, you should send the package "pXX_arob_lab4" to one of the robots available for real-world testing. The code will be loaded using the method indicated during the practical session.

## 5. References

- ROS action tutorials: http://wiki.ros.org/actionlib_tutorials
- ROS navigation: http://wiki.ros.org/navigation