

Practical Nr. 2

The task of this exercise depends on the topic that you chose. Here I will give general guidance of the things you are expected to do in order to complete P2.

2D navigation

1. Preparation:

Install the necessary packages:

```
sudo apt install ros-kinetic-amcl
sudo apt install ros-kinetic-fake-localization
```

2. Creating launch files:

In the `simplebot_navigation` package create a new `move_base` launch file `move_base.launch`. This file will be used to start the global navigation stack configuration (copy its contents from `move_base_local.launch`). Change `navigation.launch` to include this newly created file.

3. Adding localization node:

Now you need your robot to be able to publish transform from `/map` to `/odom`. This can be done by either [amcl](#) package node or [fake_localization](#) package node. You have to choose one of these nodes and add it to the `move_base.launch` file.

4. Change global costmap configuration:

In the `config` folder create `costmap_global.yaml` file and copy its contents from `costmap_global_odom.yaml`. Make sure that `move_base.launch` loads this new configuration file. Now do the following changes to `costmap_global.yaml`:

- Change global frame to `/map`
- Remove `width`, `height`, `resolution`, `origin_x` and `origin_y` parameters
- Add static layer plugin:

```
- {name: static_layer,      type: "costmap_2d::StaticLayer"}
```

5. Add map_server node:

Add a [map_server](#) package node to `move_base.launch` file. The server should load the map located in `simplebot_navigation/maps/willow_garage.yaml`.

6. Change the loaded world file:

In `simplebot_description/launch/simplebot_gazebo.launch` change the `world_name` parameter to:

```
$(find simplebot_description)/model/willow_office.world
```

7. Open `rviz` and visualize all the things that we had in L2. Change the global frame to `map`.

8. Set initial robot pose using `2D Pose Estimate` tool from top toolbar in `rviz`.
9. Try setting a goal for a robot.

Exploration

There are four packages that you can use to implement exploration behaviour, namely:

```
frontier\_exploration  
hector\_exploration  
explore\_lite  
rrt\_exploration
```

All of these packages have an example or a demo that can be used to test how this package works. Your task is to choose one of these packages and test their functionality on at least two different maps.

Controller

Localization

The aim of this task is to fuse calculated robot odometry with IMU data. The resulting fused odometry measurements should be closer to ground truth when robot wheels slip. To simulate wheel slipping you can increase the `mu1` and `mu2` parameters of `caster_link` in `simplebot.urdf.xacro` file.

The sensor fusion is performed using [robot_localization](#) package. The documentation and detailed instructions on how to set this package up can be found [here](#). There is also this [ROSCon talk](#) about `robot_localization` by its main developer. You might find it helpful in clarifying some of the steps of setting the package up.

This package can be installed using the following command:

```
sudo apt install ros-kinetic-robot-localization
```

When you finish setting the package up your task is to drive the robot in a circle or rectangle and plot the two different trajectories, namely odometry and fused odometry. In principle you should expect to see that fused trajectory is closer to the ground truth. The effects should be more visible when you have more wheel slipping, i.e. larger `mu1` and `mu2`.

3D navigation

There are two scenarios for 3D navigation. First, controlling a robotic arm and avoiding obstacles. Second, navigation of quadcopter robot. You have to choose one of these tasks and run tutorials that demonstrate this functionality.

1. For robotic arm control:

Use [MoveIt](#) package to plan paths of robotic arms. MoveIt can be installed using this command:

```
sudo apt-get install ros-kinetic-moveit
```

More info on installation can be found [here](#). After you have installed the packages you should look at MoveIt [tutorials](#). For your experiments you can use any of the [supported robot](#) simulated in Gazebo. The aim of your task is to show that the robot is able to avoid known obstacles. You could also try to add a simulated depth sensor such as Kinect or RealSense and avoiding dynamic obstacles.

2. For quadcopter:

Use a simulated quadrotor model such as [hector_quadrotor](#). The gazebo files are located in [hector_quadrotor_gazebo](#) package. The necessary packages can be installed using the following command:

```
sudo apt install ros-kinetic-hector-quadrotor
```

After you are able to start the quadcopter simulation in gazebo the next step is to do quadcopter navigation. One package implementing this functionality can be found [here](#). Your aim is to be able to set 3D target coordinate for your quadrotor simulation.