

Localization and navigation

Ivan Marković Matko Orsag Damjan Miklič
(Srećko Jurić-Kavelj)

University of Zagreb, Faculty of Electrical Engineering and Computing,
Department of Control and Computer Engineering

2018



University of Zagreb
Faculty of Electrical Engineering
and Computing



Before we begin

- Be sure to install the following packages for the class:

```
$ sudo apt-get update
$ sudo apt-get install ros- $\$ROS\_DISTRO$ -stage
$ sudo apt-get install \
  ros- $\$ROS\_DISTRO$ -teleop-twist-keyboard
$ sudo apt-get install ros- $\$ROS\_DISTRO$ -navigation
```

So you want an autonomous robot?

- A robot is in a completely unknown environment for the first time. What to do next?
- One of the most fundamental problems in mobile robotics is map building of an unknown environment
- Ends up being a prerequisite for autonomous mobile robots (without making structural changes in the environment)
- This problem is known as Simultaneous localization and mapping (SLAM)

So you want an autonomous robot?

- SLAM is difficult, and what if you are not a SLAM expert?
- Say thank you to researchers who open-sourced their solution
- At the moment popular SLAM implementations in are GMapping/OpenSLAM, Hector SLAM, Cartographer, and MRPT
- In this class we will setup a robot in Stage simulator with a laser as a sensor and perform localization with the AMCL algorithm and for homework use the Move Base package to navigate the robot in the environment

- Launch the simulator with a simple map and a robot inside

```
$ rosrn stage_ros stageros simple_pzros.world
```

- Familiarize yourself with the environment and the robot sensor
- Analyze the topics published by the simulator (`rostopic list`, `info`, ...)

Controlling the robot

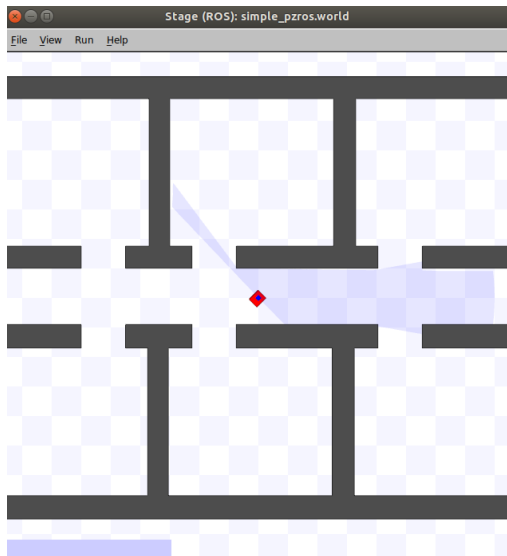
- One of the listed topics is /cmd_vel

```
$ rostopic info /cmd_vel
Type: geometry_msgs/Twist
...
```

- We will send messages to this topic in order to control the robot
- Analyze the message type. What kind of information does the message contain? (hint: `rosmmsg show`)
- The package `teleop_twist_keyboard` offers the option of controlling the robot via the keyboard

```
$ rosrn teleop_twist_keyboard teleop_twist_keyboard.py
```

Stage simulator with a robot equipped with a laser



- We assume that we have the map the environment. Do we have the map published?
- With the map we can start making the robot do useful tasks in the environment
- But we also need an algorithm which will tell us where the robot is located based on the map and received measurements
- For this purpose we will use adaptive Monte Carlo localization (AMCL) algorithm

- Let's look at the AMCL documentation
- It requires a laser scan, initial pose, the map, some transformations, and has many parameters to set up (most we can leave default, but some require attention)
- Laser scan and odometry are already being published by the simulator
- Map is just like any other topic AMCL will subscribe to, we will run our true map:

```
$ rosrun map_server map_server simple_room.yaml
```

- What about the transforms?

- We will not go at this point in the details of the `tf` package
- Launch `rviz`

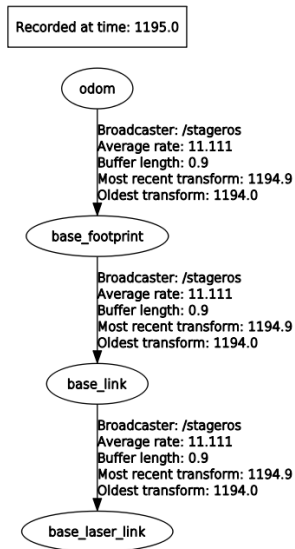
```
$ rosrun rviz rviz
```

- Add LaserScan (`/base_scan` topic), TF, and map (`/map` topic) display types
- Move the robot and see how TF coord. systems position change
- Now let's look at the available transforms:

```
$ rosrun rqt_tf_tree rqt_tf_tree
```

- Are all the required transforms available?

TF tree



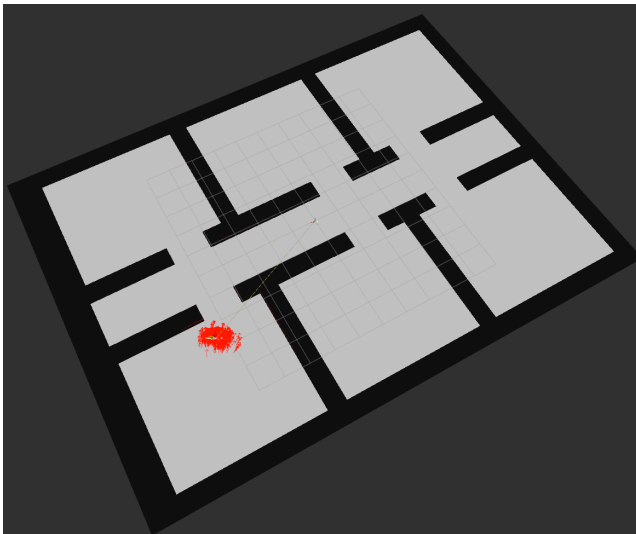
- We are missing the transform between the odometry and the map

```
$ rosrun tf static_transform_publisher \  
0 0 0 0 0 0 map odom 100
```

- We are ready to run the AMCL for which we will need to set the frames

```
$ rosrun amcl amcl _use_sim_time:=true \  
_use_map_topic:=true scan:=/base_scan \  
_odom_frame_id:=/odom _base_frame_id:=/base_link \  
_global_frame_id:=/map _tf_broadcast:=false
```

- In the RViz add PoseArray to visualize particlecloud topic and click 2D Pose Estimate and then click on the map to set up the initial position
- Drive the robot around a bit and see how the algorithm converges to the location



- Now that you have the map and a way to localize your robot in the map we can start with robot navigation
- In essence, the purpose of navigation is to move the robot from point A to point B
- But this stems a plethora of problems: finding the optimal global path, acting locally, avoiding static and dynamic obstacles, recalculating paths due to changes in the environment, controlling the robot etc.
- We will use the Move Base package

Starting multiple ROS nodes

- As our applications get more complex, we will need to start multiple nodes, set many parameters etc.
- Solution is to use `roslaunch`, a tool for starting multiple nodes and setting multiple parameters
- Many packages come with already set-up “launch files”

```
$ roslaunch package_name file.launch
```

- `roslaunch` uses XML files (e.g. `file.launch`) that describe the nodes that should be run, parameters that should be set, and other attributes
- Location of the launch file is resolved by the argument `package_name`; however, this is not necessary

roslaunch XML syntax (simple example)

```
<launch>  
  <node name="shove" pkg="turtlecontrol" type="push.py" />  
  <remap from="turtle1/cmd_vel" to="cmd_vel"/>  
</launch>
```

- Starts a node shove using the push.py executable from the turtlecontrol package and call it shove
- In the example the simulator is subscribed to turtle1/cmd_vel, but our node publishes cmd_vel, so we need to remap this
- Visit this link for more info

For homework you will need to use the Move Base package to make the robot autonomously navigate in the map. Unpack the prepared archive where you will see the launch file (`move_base_pzros.launch`) running the required node and loading all the necessary parameter files.

Assignments

- 1 Study the Move Base documentation and submit the answers to the following questions:
 - What is a holonomic robot?
 - What is a costmap and why is it needed?
 - What is the difference between the global and local navigation plan?

Assignments (continued)

- 2 Fill in the required parameters in place of the question marks (???) in the provided parameter files. Be sure that Stage simulator, AMCL, and RViz are running, and that robot is properly localized. Run the launch file and in RViZ select the 2D Nav Goal and give the robot a destination. If all is correct the robot should autonomously navigate. In RViz show the following and submit it as screenshots:
 - Costmap generated by the Move Base package.
 - Global and local planned path (both need to be clearly visible).

- <http://www.ros.org/wiki/gmapping>
- <http://openslam.org/>
- <http://www.ros.org/wiki/amcl>
- http://www.ros.org/wiki/move_base