Map building and navigation

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

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

2017

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-stdr-simulator
$ sudo apt-get install  \
  ros-$ROS_DISTRO-teleop-twist-keyboard
$ sudo apt-get install ros-$ROS_DISTRO-amcl
$ sudo apt-get install ros-$ROS_DISTRO-move-base
$ sudo apt-get install ros-$ROS_DISTRO-gmapping
```

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 two most commonly used SLAM implementations in 2D are GMapping/OpenSLAM and Hector SLAM
- In this class we will setup a robot in the STDR simulator with a laser as a sensor and build a map of our environment with the GMapping algorithm (homework), perform localization with the AMCL algorithm and use the `move_base` package to navigate the robot in the environment

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
- 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

Simulator

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

```
$ roslaunch stdr_launchers \
  server_with_map_and_gui_plus_robot.launch
```

- Familiarize yourself with the environment and the robot sensors
- Launch `rviz`

```
$ roslaunch stdr_launchers rviz.launch
```

- Analyze the topics published by the simulator (`rostopic list`, `info`, `...`)

Controling the robot

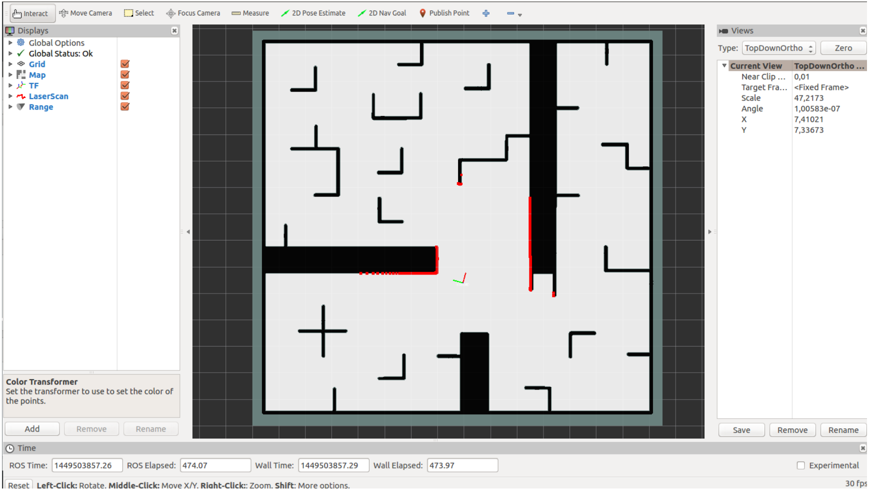
- One of the listed topics is /robot0/cmd_vel

```
$ rostopic info /robot0/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: rosmsg show)
- The package teleop_twist_keyboard offers the option of controlling the robot via the keyboard

```
$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py \
cmd_vel:=robot0/cmd_vel
```

RViz showing map, robot and laser scan



Localization

- We assume that we have the map the environment. Where is this map published?
- Having the map, we can start making the robot do useful tasks in the environment
- But first we 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

AMCL

- Let's look at the AMCL documetation
- 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 is already being published by the simulator
- Map is just like any other topic AMCL will subsribe to, usually it is ran as follows but the simulator is already publishing the map

```
$ rosrun map_server map_server map.yaml
```

- What about the transforms?

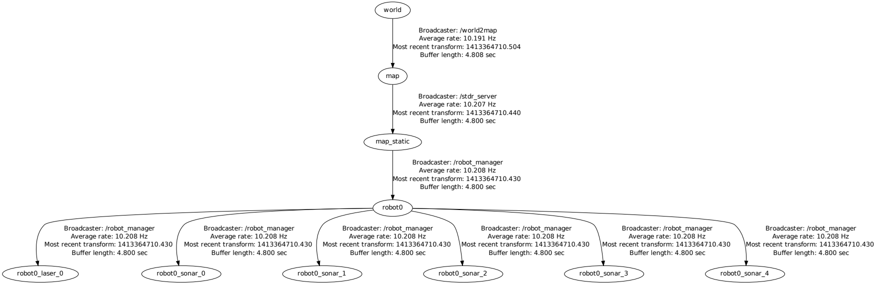
AMCL

- We will not go at this point in the details of the `tf` package
- Take a look in RViz at defined coordinate systems (TF checkbox should be switched on)
- Now lets look at the available transforms:

```
$ rosrun rqt_tf_tree rqt_tf_tree
```

- Are all the required transforms available?

TF tree


world
Broadcaster: /world2map
Average rate: 10.191 Hz
Most recent transform: 1413364710.504
Buffer length: 4.808 sec
map
Broadcaster: /stdr_server
Average rate: 10.207 Hz
Most recent transform: 1413364710.440
Buffer length: 4.800 sec
map_static
Broadcaster: /robot_manager
Average rate: 10.208 Hz
Most recent transform: 1413364710.430
Buffer length: 4.800 sec
robot0
Broadcaster: /robot_manager
Average rate: 10.208 Hz
Most recent transform: 1413364710.430
Buffer length: 4.800 sec
robot0_laser_0
robot0_sonar_0
robot0_sonar_1
robot0_sonar_2
robot0_sonar_3
robot0_sonar_4

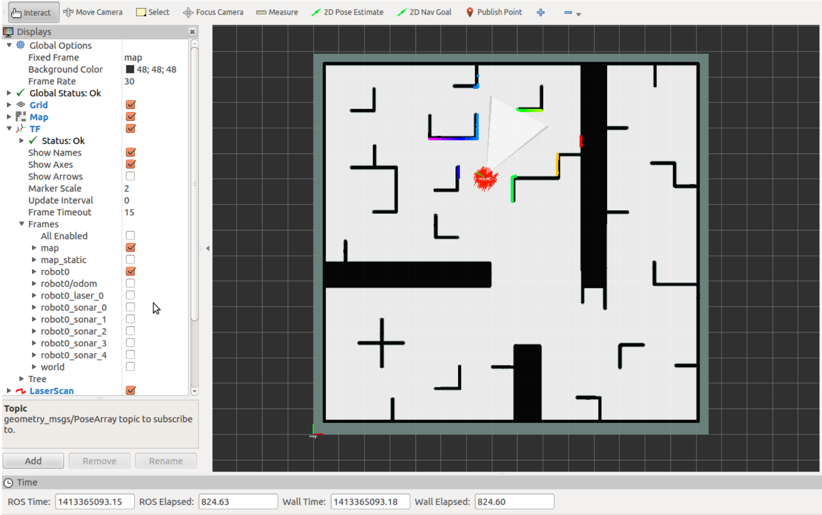
AMCL

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

```
$ rosrun amcl amcl _use_map_topic:=true \
scan:=/robot0/laser_0 _odom_frame_id:=/world \
_base_frame_id:=/robot0 _global_frame_id:=/world
```

- We are taking a shortcut here by saying that our odometry frame is the world frame (this causes error but for the moment it is safe to ignore it)
- 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

AMCL


Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
map
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Grid
Map
TF
Status: Ok
Show Names
Show Axes
Show Arrows
Marker Scale
2
Update Interval
0
Frame Timeout
15
Frames
All Enabled
map
map_static
robot0
robot0/odom
robot0_laser_0
robot0_sonar_0
robot0_sonar_1
robot0_sonar_2
robot0_sonar_3
robot0_sonar_4
world
Tree
LaserScan
Topic
geometry_msgs/PoseArray topic to subscribe to.
Add
Remove
Rename
Time
ROS Time: 1413365093.15
ROS Elapsed: 824.63
Wall Time: 1413365093.18
Wall Elapsed: 824.60

Navigation

- Now that you have the map and a way 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

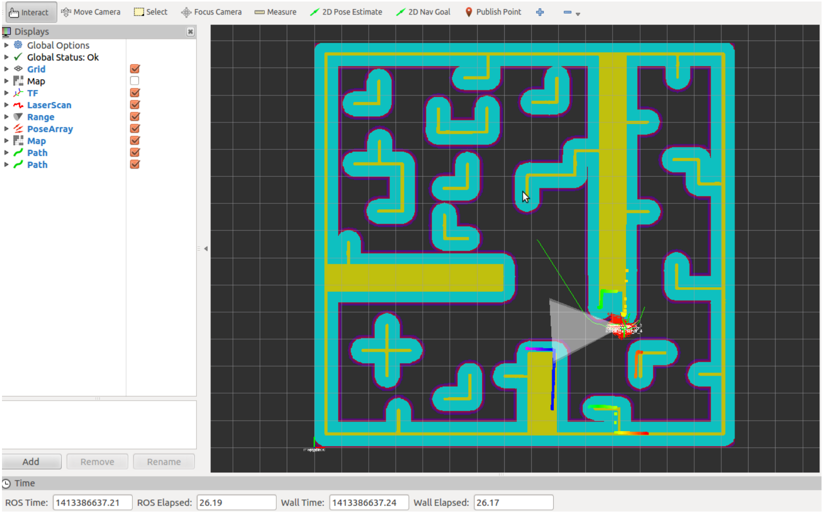
Move Base

- Unpack the prepared configuration files and the `launch`; update the hardcoded lines in the launch file to point at the unpacked folder
- Be sure that STDR simulator, AMCL, and RViz are running
- Open the prepared RViz visualization file
- Make sure that the robot is properly localized
- Launch the prepared `launch` file

```
$ roslaunch move_base_stdr_sim.launch
```

- In RViz click on the `2D Nav Goal`, set it, and see the robot move!

Move Base


Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Global Status: Ok
Grid
Map
TF
LaserScan
Range
PoseArray
Map
Path
Path
Add
Remove
Rename
Time
ROS Time: 1413386637.21
ROS Elapsed: 26.19
Wall Time: 1413386637.24
Wall Elapsed: 26.17

Homework

For homework you will need to build the map of the environment using the GMapping package.

Assignments

1. Study the GMapping documentation. What topics does GMapping subsribe to?
2. You will have to remap the scan topic, and odometry and base link frames (Hint: look for the exact topic names in the GMapping documentation or GitHub source code or both)
3. STDR simulator publishes map on the `map` topic. Make sure that GMapping does not do the same or otherwise the GMapping node will kill the STDR node!
4. Manually drive the robot until you have a complete map. Create an image consisting of four snapshots of the map in RViz taken at different times. Submit the snapshot images as your homework.

Homework

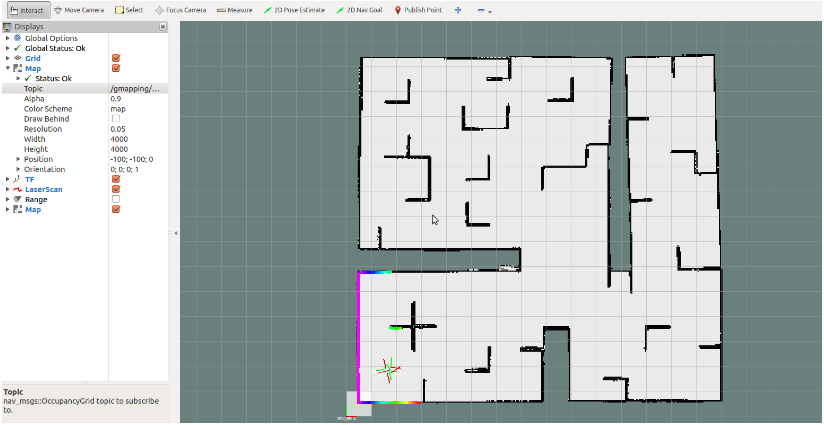


Figure: An example of a map produced by the GMapping package

Useful links

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