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

2013

University of Zagreb
Faculty of Electrical Engineering and Computing

Before we begin

- Be sure to install the following for the class

```
$ sudo apt-get install ros-fuerte-turtlebot-simulator
$ sudo apt-get install  ros-fuerte-turtlebot-apps
```

- Check if you can run Gazebo

```
$ rosrun gazebo gazebo
$ rosrun gazebo gui
```

- If Gazebo throws an error, try to install proprietary driver (from the manufacturer) for your graphics card

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 Turtlebot in Gazebo with a simulated laser and build a map of our environment with the GMapping algorithm

Creating a package

- Create a package that depends on Turtlebot's packages for simulation in Gazebo and teleoperation

```
$ roscreate-pkg ros_liv_turtlebot turtlebot_gazebo \
turtlebot_teleop gmapping amcl move_base
```

- Make sure that the package is in ROS path

```
$ source ~/ros/setup.bash
$ rospack profile
```

- Try to `roscd` into it with autocomplete

Creating launch files

- In the ros_liv_turtlebot folder make a launch folder where you need to create office_turtlebot.launch file

```
<launch>
  <!-- we run gazebo with a world argument where path to
  gazebo_worlds is automatically located. We also run the
  gazebo gui -->
  <param name="/use_sim_time" value="true" />
  <node name="gazebo" pkg="gazebo" type="gazebo"
    args="-u $(find gazebo_worlds)/worlds/
    simple_office.world" />
  <node name="gazebo_gui" pkg="gazebo" type="gui" />
```

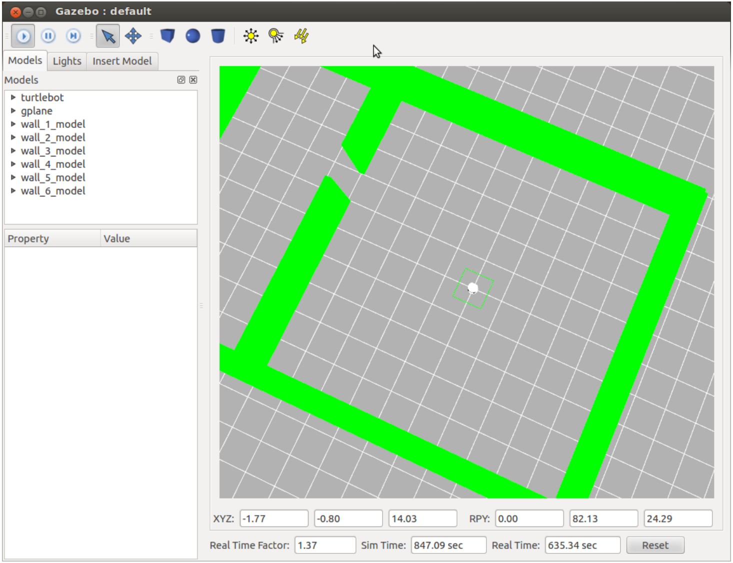
Creating launch files

```
  <!-- we include preexisting launch files which will set
   up the robot model and the node for teleop (in one line!) -
  <include file="$(find turtlebot_gazebo)/
    launch/robot.launch" />
  <include file="$(find turtlebot_teleop)/
    keyboard_teleop.launch" />
</launch>
```

- Now we are ready to launch gazebo with the Turtlebot in it

```
$ roslaunch ros_liv_turtlebot office_turtlebot.launch
```

Turtlebot in the office



GMapping

- Go to GMapping package documentation and see the requirements
- Gmapping requires the following:
  - Odometry data (implicitly) and laser scans
  - Transformations from `laser`→`base_link` and from `base_link`→`odometry`
  - Several parameters (or you can leave them default)
- See which topics are available

```
$ rostopic list
```

- You will see a lot of topics, among which there is the `/scan` topic

GMapping

- See info on the /scan topic

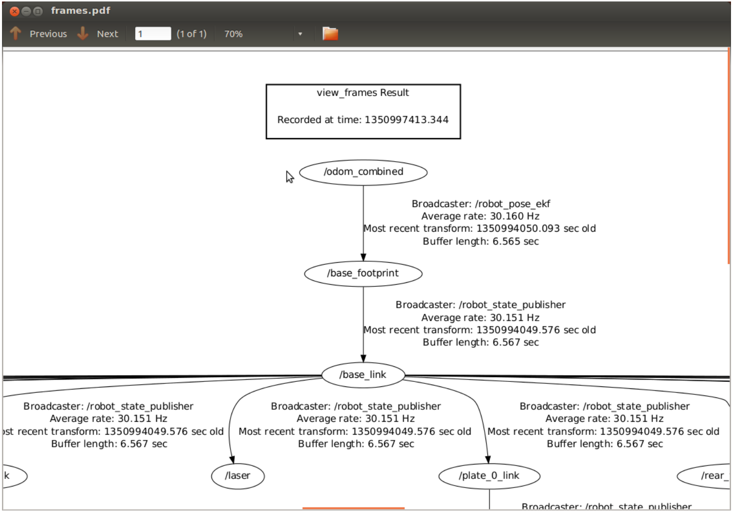
```
$ rostopic info /scan
Type: sensor_msgs/LaserScan
...
```

- This is exactly what GMapping requires!
- Let's see which transforms currently we have

```
$ rosrun tf view_frames
```

- We get a pdf file as a result with the whole transform tree

TF tree



frames.pdf
Previous
Next
1
(1 of 1)
70%
view_frames Result
Recorded at time: 1350997413.344
/odom_combined
Broadcaster: /robot_pose_ekf
Average rate: 30.160 Hz
Most recent transform: 1350994050.093 sec old
Buffer length: 6.565 sec
/base_footprint
Broadcaster: /robot_state_publisher
Average rate: 30.151 Hz
Most recent transform: 1350994049.576 sec old
Buffer length: 6.567 sec
/base_link
Broadcaster: /robot_state_publisher
Average rate: 30.151 Hz
Most recent transform: 1350994049.576 sec old
Buffer length: 6.567 sec
/laser
/plate_0_link

GMapping

- The odometry is being published as `/odom_combined` topic by the `/robot_pose_ekf` node
- Node which fuses measurements from wheel encoders, IMU, and visual odometry to provide odometry data
- Now run the GMapping node

```
$ rosrun gmapping slam_gmapping \
_odom_frame:=odom_combined
```

- Let's visualize this in RViz

```
$ rosrun rviz rviz -d `rospack find \
turtlebot_navigation`/nav_rviz.vcg
```

- Drive the robot around and see how map is being built

GMapping

- We can see that the robot is detecting itself with the simulated laser scan from the RGBD sensor
- A proper way to remove this would be to write a filter which would remove scans based on the robot's configuration
- For simplicity we will ignore all the measurements which are closer than 0.3 m

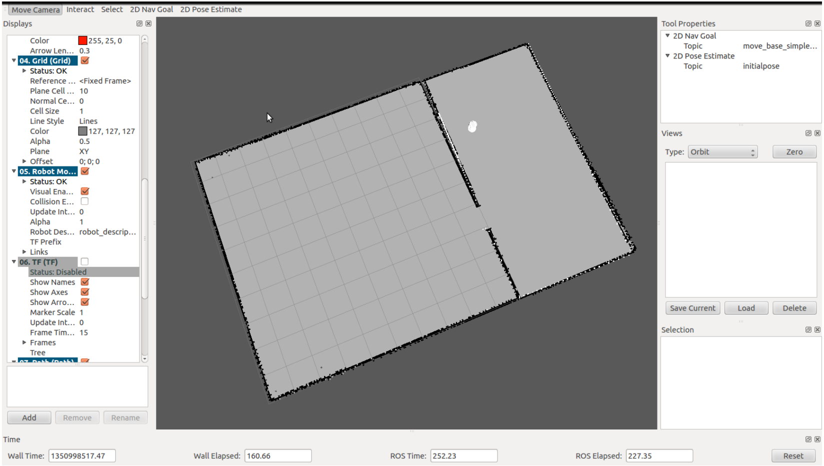
Exercize

Find the location of the file that describes the simulated sensor. The robot description is in the URDF files. Hint: `robot.launch` is the one that sets up everything needed for robot simulation.

- Save the map

```
$ rosrun map_server map_saver
```

Office map



Move Camera
Interact
Select
2D Nav Goal
2D Pose Estimate
Displays
Color 255, 25, 0
Arrow Len… 0.3
04. Grid (Grid)
Status: OK
Reference … <Fixed Frame>
Plane Cell … 10
Normal Ce… 0
Cell Size 1
Line Style Lines
Color 127, 127, 127
Alpha 0.5
Plane XY
Offset 0; 0; 0
05. Robot Mo…
Status: OK
Visual Ena…
Collision E…
Update Int… 0
Alpha 1
Robot Des… robot_descrip…
TF Prefix
Links
06. TF (TF)
Status: Disabled
Show Names
Show Axes
Show Arro…
Marker Scale 1
Update Int… 0
Frame Tim… 15
Frames
Tree
Add
Remove
Rename
Tool Properties
2D Nav Goal
Topic move_base_simple…
2D Pose Estimate
Topic initialpose
Views
Type: Orbit
Zero
Save Current
Load
Delete
Selection
Time
Wall Time: 1350998517.47
Wall Elapsed: 160.66
ROS Time: 252.23
ROS Elapsed: 227.35
Reset

Localization

- Now that you have the map of your environment you can make the robot drive autonomously with a purpose
- But first we need an algorithm which will tell us where the robot is 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)
- Launch the Turtlebot in the office
- The laser scan is already being published by Gazebo
- The map is just like any other topic to which AMCL will subsribe to

```
$ rosrun map_server map_server \
 `rospack find ros_liv_turtlebot`/map/map.yaml
```

AMCL

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

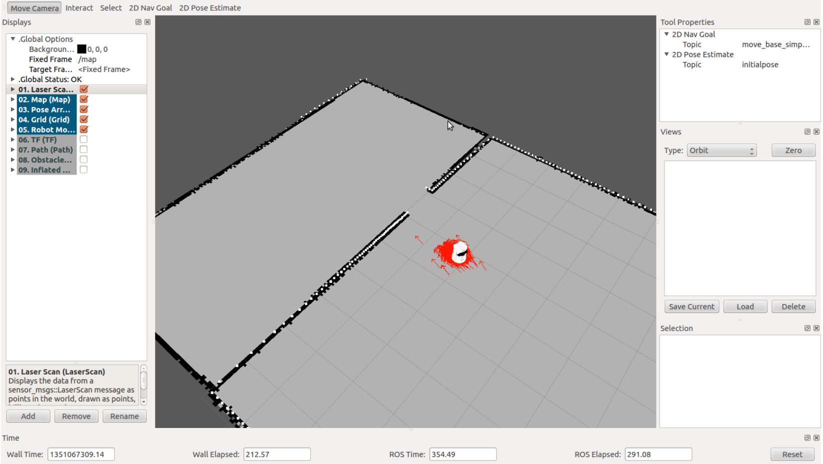
```
$ rosrun amcl amcl _odom_frame_id:=odom_combined
```

- Launch the RViz for visualization
- In the RViz 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

Homework

Write a launch file that will run the `map server` and the `amcl` just as we did manually in the console.

Office localization



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

- Copy the launch file from `turtlebot_navigation` package to `ros_liv_turtlebot/launch` folder (`config/move_base_turtlebot.launch`)
- This file is set to run the Move Base package with some Turtlebot specific configuration files
- Add the following line to the copied launch file

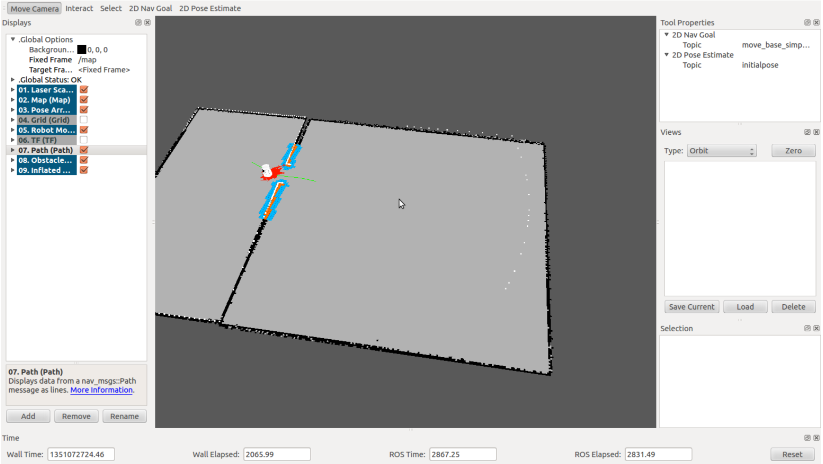
```
<param name="local_costmap/global_frame"
 value="/odom_combined"/>
```

Move Base

- Be sure that Gazebo, map server, AMCL, and RViz are running
- Localize the robot the in map
- Launch the edited file

  ```
  $ roslaunch ros_liv_turtlebot move_base_turtlebot.launch
  ```
- In RViz click on the 2D `Nav Goal`, set it, and see the robot move!

Move Base


Move Camera
Interact
Select
2D Nav Goal
2D Pose Estimate
Displays
.Global Options
Backgroun... 0, 0, 0
Fixed Frame /map
Target Fra... <Fixed Frame>
.Global Status: OK
01. Laser Sca...
02. Map (Map)
03. Pose Arr...
04. Grid (Grid)
05. Robot Mo...
06. TF (TF)
07. Path (Path)
08. Obstacle...
09. Inflated ...
07. Path (Path)
Displays data from a nav_msgs::Path message as lines. More Information.
Add
Remove
Rename
Tool Properties
2D Nav Goal
Topic move_base_simp...
2D Pose Estimate
Topic initialpose
Views
Type: Orbit
Zero
Save Current
Load
Delete
Selection
Time
Wall Time: 1351072724.46
Wall Elapsed: 2065.99
ROS Time: 2867.25
ROS Elapsed: 2831.49
Reset

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