Workshop “ROS voor Engineers” - deel 2

auteur: Eric Dortmans (e.dortmans@fontys.nl)

Voorbereiding

Update de ros_examples stack die je in de vorige sessie hebt geinstalleerd:

```
cd ~/catkin_ws/src/ros_examples
git pull
cd ~/catkin_ws
catkin_make
```

Installeer de Turtlebot simulator stack

```
sudo apt-get install ros-indigo-turtlebot
sudo apt-get install ros-indigo-turtlebot-apps
sudo apt-get install ros-indigo-turtlebot-rviz-launchers
sudo apt-get install ros-indigo-turtlebot-create-desktop
sudo apt-get install ros-indigo-turtlebot-simulator
```

Besturen van een gesimuleerde robot in Stage

We gebruiken nu de Stage 2D simulator in plaats van een echte robot. De werking is hetzelfde.

Start een gesimuleerde robot in een gesimuleerde wereld:

```
roslaunch stage_worlds kinect_world.launch
```

Start de *arbotix gui* om met je muis de robot te besturen:

```
arbotix_gui
```

Bekijk in een ander window de odometry informatie zoals die door de robot wordt gepubliceerd:

```
rostopic echo /odom
```

Zo kun je zien hoe een *Odometry* message is opgebouwd:

```
rostopic type /odom | rosmsg show
```

Rij wat rond en kijk hoe de odom informatie verandert.

Start *rviz* om een en ander te visualiseren:

```
roslaunch stage_worlds rviz_stage.launch
```

TF

Bekijk de *TF tree*:

```
rosrun rqt_tf_tree rqt_tf_tree
```

Of run rqt en start de TF Tree plugin:

```
rqt &
```

```
Plugins > Visualization > TF Tree
```

Bekijk alles wat er gepubliceerd wordt op de TF tree

```
rosrun tf tf_monitor
```

Vraag specifieke, enkelvoudige transformaties op:

```
rosrun tf tf_echo /base_link /base_laser_link
rosrun tf tf_echo /base_footprint /base_link
rosrun tf tf_echo /odom /base_footprint
rosrun tf tf_echo /map /odom
```

Je kunt ook complexe, meervoudige transformaties opvragen, b.v.:

```
rosrun tf tf_echo /map /base_laser_link
```

Navigatie

Stop de Stage simulatie en RViz. Voor de zekerheid sluit alle windows.

We gaan nu experimenteren met de ROS Navigatie stack.

Start *gmapping* om een kaart te maken van de (gesimuleerde) robot wereld:

```
roslaunch stage_navigation kinect_gmapping.launch
```

Start de *arbotix gui* om met je muis de robot te besturen:

```
arbotix_gui
```

Je kunt overigens de robot in Stage ook verplaatsen door hem met je muis te selecteren en dan te "verslepen".

Rij de robot rond door de gesimuleerde wereld. Kom ook regelmatig terug waar je al geweest bent.
In het RViz window kun je dan zien hoe gmapping een kaart (map) van de wereld probeert te maken.

Je zult merken dat het niet meevalt om een goede map te maken!

We zullen een andere robot proberen. Stop de lopende simulatie en start de volgende:

```
roslaunch stage_navigation laser_gmapping.launch
```

Zie en merk je verschil? Wat neem je waar en hoe zou dat komen?

Als je tevreden bent met je map kun je hem opslaan:

```
rosrun map_server map_saver -f /tmp/world_map
```

Stop de lopende lsimulatie. Voor de zekerheid sluit alle terminal windows.

Nu we een kaart hebben kunnen we die door de robot laten gebruiken om voortaan zelf zijn weg te zoeken.
We gebruiken daarvoor de *amcl* localizatie node en de *move_base* navigatie node:

```
roslaunch stage_navigation kinect_amcl.launch map_file:=/tmp/world_map.yaml
```

Bekijk de Computation Graph:

```
rqt_graph
```

Gebruik the *2D Nav Goal* knop in RViz om robot een doel (goal) te geven. Click eerst op de knop en dan op een plaats in de map. Stuur hem bijvoorbeeld naar linksboven en vervolgens naar rechtsonder in de map.

Stop de draaiende Stage simulatie. Voor de zekergeheid sluit alle terminal windows.

Besturen van een gesimuleerde Turtlebot in Stage

We zullen nu hetzelfde doen maar met een robot die ook echt bestaat: de Turtlebot.

Start de Turtlebot sofware:

```
export TURTLEBOT_BASE=roomba
export TURTLEBOT_STACKS=circles
export TURTLEBOT_3D_SENSOR=kinect
export TURTLEBOT_STAGE_MAP_FILE=`rospack find turtlebot_stage`/maps/maze.yaml
export TURTLEBOT_STAGE_WORLD_FILE=`rospack find turtlebot_stage`/maps/stage/maze.world
roslaunch turtlebot_stage turtlebot_in_stage.launch
```

Bekijk de Computation Graph:

```
rqt_graph
```

Wat zou de cmd_vel_mux doen? Lees de parameter file (een YAML file) van de cmd_vel_mux:

```
cat `rospack find turtlebot_bringup`/param/mux.yaml
```

Gebruik het keyboard om de robot te besturen:

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

De robot kan ook autonoom navigeren. Gebruik de *2D Nav Goal* knop in RViz om robot een doel (goal) te geven.

Merk op dat de Turtlebot mooi gevisualiseerd wordt in RViz. Hoe dat moet gaan we volgende keer bekijken.

EXTRA: Rijden van een gesimuleerde Turtlebot in Gazebo

Gazebo is een 3D simulator. Dit vraagt natuurlijk meer van de grafische ondersteuning en van de performance van je laptop, maar is wel veel echter.

Start de Gazebo simulator:

```
export TURTLEBOT_BASE=roomba
export TURTLEBOT_STACKS=circles
export TURTLEBOT_3D_SENSOR=kinect
export TURTLEBOT_GAZEBO_WORLD_FILE=`rospack find turtlebot_gazebo`/worlds/playground.world
roslaunch turtlebot_gazebo turtlebot_world.launch
```

Run de *gmapping* node:

```
roslaunch turtlebot_gazebo gmapping_demo.launch
```

Visualizeer de robot tijdens navigatie:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Start de *arbotix_gui* om de robot te besturen:

```
arbotix_gui cmd_vel:=cmd_vel_mux/input/teleop
```

De simulatie bootst ook een echte kinect na. Voeg maar eens een Camera toe in RViz op het topic /camera/rgb/image_raw

EXTRA: Navigeren met een echte Turtlebot

Voor het installen van *ROS_IP* en *ROS_MASTER_URI* zie laatste opdracht van vorige sessie.

Start de robot, inclusief gmapping node.

Bekijk en bestuur de robot via RViz op je laptop:

```
roslaunch turtlebot_rviz_launchers view_navigation.launch
```

Voeg ook een Camera toe op het /camera/rgb/image_raw topic.

Referenties

- ROS Tutorials
- TF
- rqt
- navigation
- TurtleBot
- turtlebot_simulator
- turtlebot_navigation