Robot Practical Course Bachelor
Assignment #2

In this assignment, you are introduced to the robotic platform "turtlebot" which you will program in the rest of this course. You will get to know its sensors and their data visualization. In a second step you will write a program to move the Turtlebot and use the sensor data to stop before it collides with obstacles in the environment.

Real robots require to be handled with care.
Be prepared to rescue the robot from hitting obstacles.

Task 2.1 Starting the Turtle and ROS: You can login to the robot via SSH with the username `prac2022` on the hosts `donny`, `leo`, `mikey`, and `raph` from any TAMS pool computer. You will find a pre-built ROS workspace `~/ros` on each robot you should use to develop your own ROS packages.

2.1.1: Boot the Turtlebot & its laptop. Login to the system.

2.1.2: Start ROS and the nodes for a basic setup to use the mobile base and the camera. Good tutorials are available at the Turtlebot wiki page:
`http://wiki.ros.org/Robots/TurtleBot`
You do **not** need to read them now.
To account for custom changes on TAMS' turtlebots (i.e. an additional laser scanner), you have to start TAMS-specific launch files which are modified for our setup. To bringup the turtlebot, don't run any launch file from the package `turtlebot_bringup`. Instead run:

```
roslaunch tams_turtlebot_bringup tams_turtlebot.launch
```

2.1.3: Start the visualization tool RViz on your local computer.
Make sure to correctly export the `ROS_MASTER_URI` on your desktop computer before you start the RViz visualization tool.
It can be set using the following command on the desktop computer:

```
export ROS_MASTER_URI="http://ROBOT_NAME:11311"
```

This way, you tell the ROS nodes in that terminal where to look for the ROS master it should communicate with.
Afterwards, RViz can be started with:

```
rviz
```

As the robot does not know a map frame at the moment, change the Fixed Frame in RViz to base_footprint.
Try to visualize the robot model, get a live camera image, and drive around manually with the `turtlebot_teleop` node[1]. It can visualize the point cloud provided by the Kinect sensor. and many other measurements taken by the robot.

[1]The node can be launched using the `keyboard_teleop` launch file in the `turtlebot_teleop` package.

Task 2.2 Kinect Listener: Write a listener node to read the Kinect data

2.2.1: Find out the identifier and the message type of the Kinect "laser_scan" topic.

2.2.2: Look up the ROS tutorial for a simple subscriber. There is one for C/C++ and one for Python. You can choose the language you like. Make sure you are looking up the right version of ROS. You used this tutorial last time for the publisher.
C/C++: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(c++)`
Python: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(python)`

2.2.3: On the course website you will find the `deploy.sh` script. Make sure to set the `ROS_MASTER_URI`. The script copies your package to the robot and runs `catkin build` on the robot.

```
./deploy.sh
```

Task 2.3 Simple Movement: Write a node to move the Turtlebot.

2.3.1: Extend your node to move the Turtle bot.

2.3.2: Test your node on the robot. Make sure there is enough space around the robot.

Never leave the robot unattended!

Task 2.4 Detect obstacles: Write a program to move the TurtleBot and stop in front of an obstacle.

2.4.1: Use your code from the previous two tasks and extract the distance to an object directly to the front of the Kinect.

2.4.2: Now let the robot move forward until it detects an obstacle closer than 1 m. Use a soft obstacle to test your code.

Task 2.5 Optional: Wall following: Write a program to move the TurtleBot along a wall to explore the environment.