Assignment 2

The purpose of this assignment is to investigate how to steer a mobile robot towards a goal pose.

# Getting ready

Assumption is that you completed all ROS Getting Started tutorials (<http://www.ros.org/wiki/ROS/Tutorials>). If you did not, do it now.

Now get acquainted with Stage (<http://wiki.ros.org/stage>). Stage is a 2D mobile robot simulator. Best is to start it and play with it. To start Stage with an existing world do the following:

* In terminal 1: roscore
* In terminal 2: rosrun stage\_ros stageros `rospack find stage\_ros`/world/willow-erratic.world
* In terminal 3: rqt

In rqt start the Robot Steering plugin and use it to drive the robot. Try other plugins too, e.g. rviz.

In Stage you should certainly try various “View” options, in particular “Data” and “Trails”.

Now download the world file ‘*empty.world’*. This world should be used to perform the task as described below. The world can be started manually as follows: rosrun stage\_ros stageros empty.world

# The Task

In this task you are going to develop a couple of ‘steering nodes*’*. A ‘*steering node*’ makes the robot drive to the goal pose which it receives as a ‘[geometry\_msgs/PoseStamped](http://docs.ros.org/api/geometry_msgs/html/msg/PoseStamped.html)’ message on the topic ‘/goal’. It also subscribes to the topic ‘/odom’ to receive robot pose updates in the form of ‘[nav\_msgs/Odometry](http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html)’ messages and steers the robot to the goal via ‘[geometry\_msgs/Twist](http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html)’ messages which it publishes on ‘/cmd\_vel’.

More info on the stageros node (interface to Stage) can be found here: <http://wiki.ros.org/stage_ros>

/goal

/cmd\_vel

/odom

Create a new package “*assignment2*”. In this package write the following steering nodes in C++:

* ‘*pointshoot\_node*’
  + Rotate the robot in place to point towards the goal, then drive in a straight line towards the goal and finally rotate to comply with the goal orientation.
* ‘*servoing\_node’*
  + This node uses servoing (control laws) to reach the goal: linear speed proportional to remaining distance to goal, and angular speed proportional to angle error (see powerpoint slides)

Hints:

* You will find that representing the robot position error, i.e. the difference between its current pose and the goal pose, in polar coordinates facilitates the problem. You can represent the robot’s position in terms of a distance and an angle to the goal.
* Note that the orientation is defined in terms of a Quaternion, i.e. do not expect values in degrees or radians. Convert the Quaternion to Yaw if you need an angle representation.

ROS has support for datatypes such as Quaternions. These classes (e.g. *tf::Quaternion*) and helper functions, e.g. *tf:: getYaw* and *tf:: createQuaternionFromYaw*, can be found in the TF namespace: <http://docs.ros.org/api/tf/html/c++/namespacetf.html>. To use any of these put the following line in your code: *#include <tf/transform\_datatypes.h>*

* Note that all control formula’s assume that angles are normalized to the range [-pi,+pi]. ROS has support for calculations (such as normalization and difference) involving *angles*: <http://wiki.ros.org/angles>

General requirements:

* Place the ‘empty.world’ file in the world subdirectory of your package.
* Provide launch files for testing. Each such launch file should start the empty.world in Stage plus one of the steering nodes.

Now do the following research:

* Investigate and compare the performance of both steering nodes

# What to Submit

Your submission for this assignment will have two parts:

1. The source code, i.e. a zip (or rar) file containing the package
2. A document which should:
   1. explain how everything works and include computation graph,
   2. describe test scenario’s, and include test vectors (using rostopic pub on topic /goal)
   3. describe the results of your tests and include screendumps of trails in Stage.