Assignment 3 ROS

Rafal Grasman

Menno Broeders

**Running our code**

**This document assumes that ~/ros\_workspace/ is in your ROS\_PATH variable!**

Extract the zip into ~/ros\_workspace/

*Detect the package:*

rosdep update

(if needed, run rosdep init beforehand)

*Create folder for this assignment:*

Mkdir ~/ros\_workspace/assignment3

*Build the binary:*

cd ~/ros\_workspace/assignment3

rosmake

*Then launch the package: (terminal1)*

roslaunch assignment3 assignment3-rectangle.launch

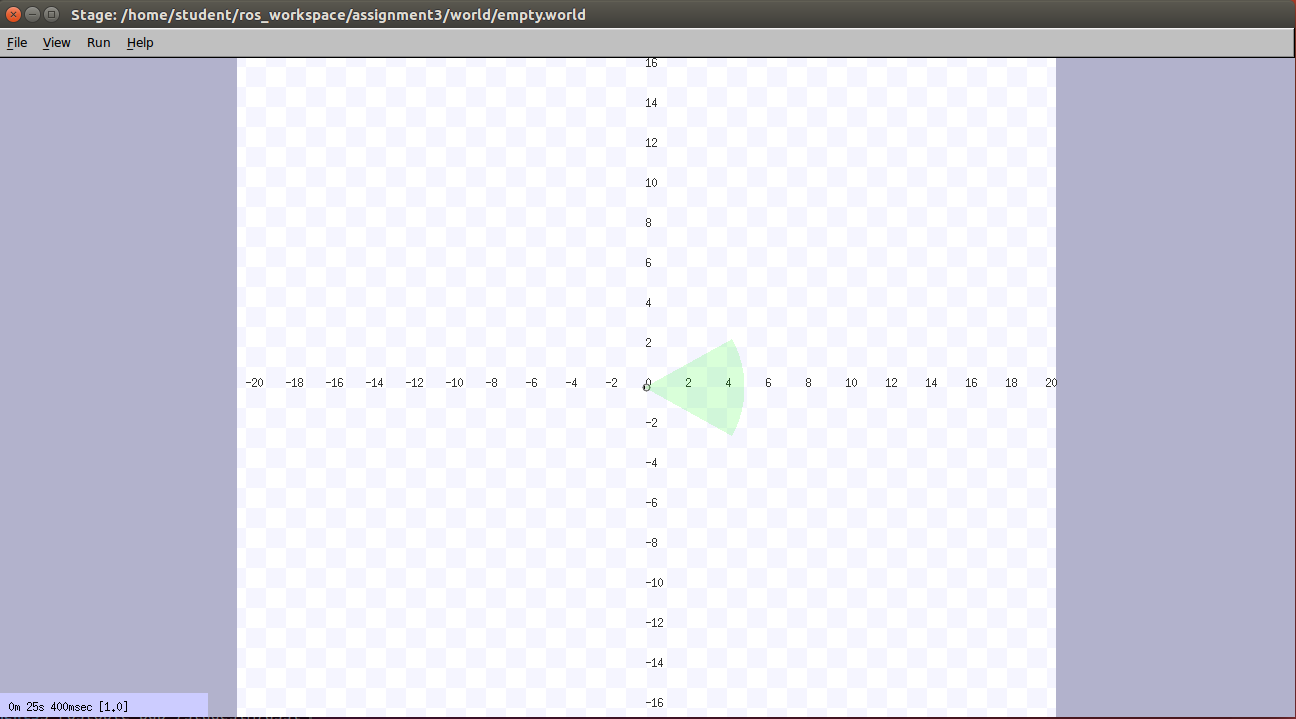
*or*

roslaunch assignment3 assignment3-triangle.launch

*Send a goal command: (terminal2)*

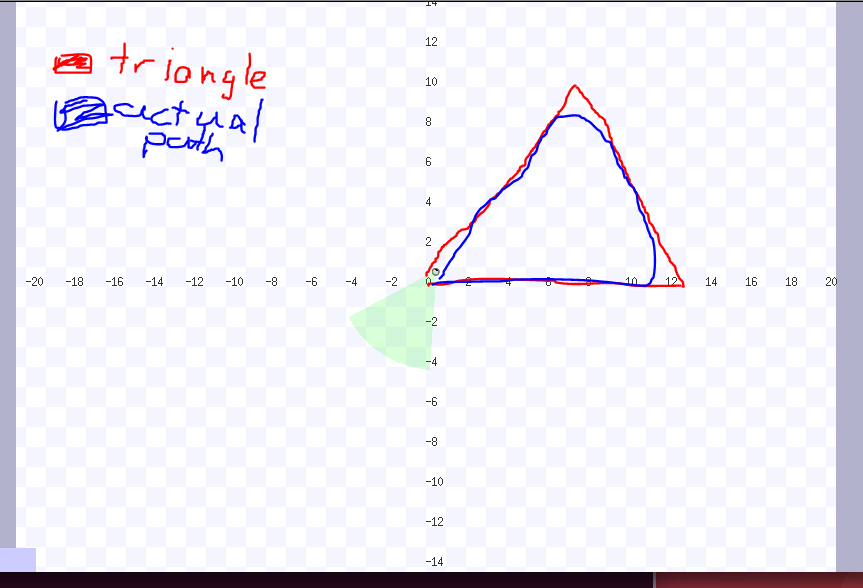
*Since the only varying factor is the size, we created a message called Goal, which holds one float. This float is used to define the size of each side of the rectangle or triangle.*

rostopic pub /stagesim/assignment3/goal assignment3/Goal 12

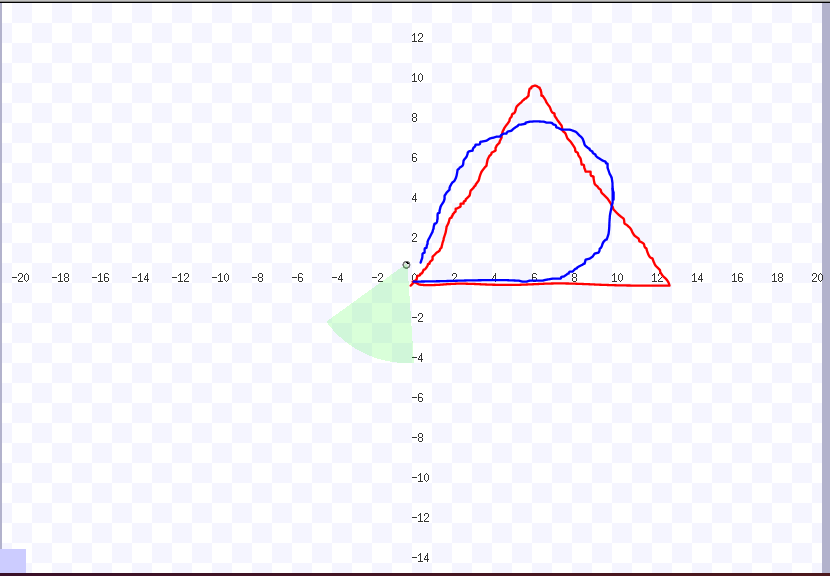
Empty world after running roslaunch command:

*Result: rostopic pub /stagesim/assignment3/goal assignment3/Goal 12*

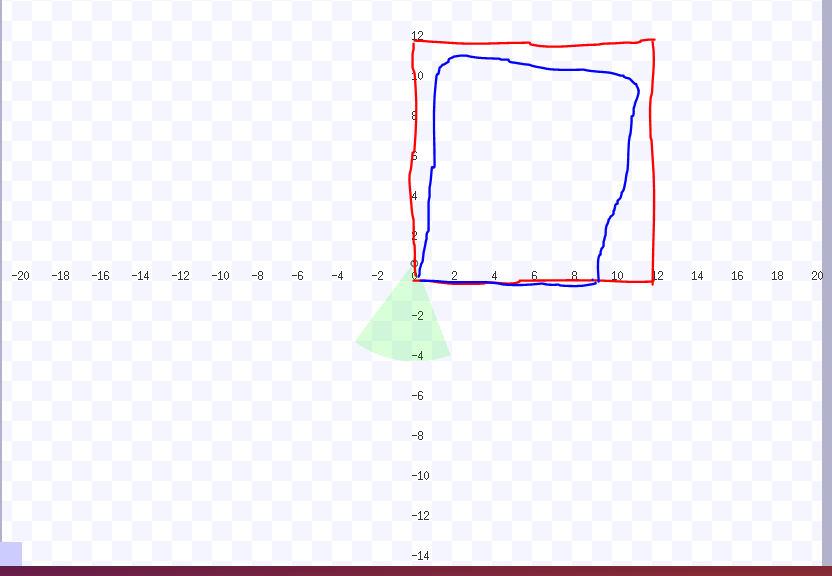
*Lookahead distance was defined at 3.*



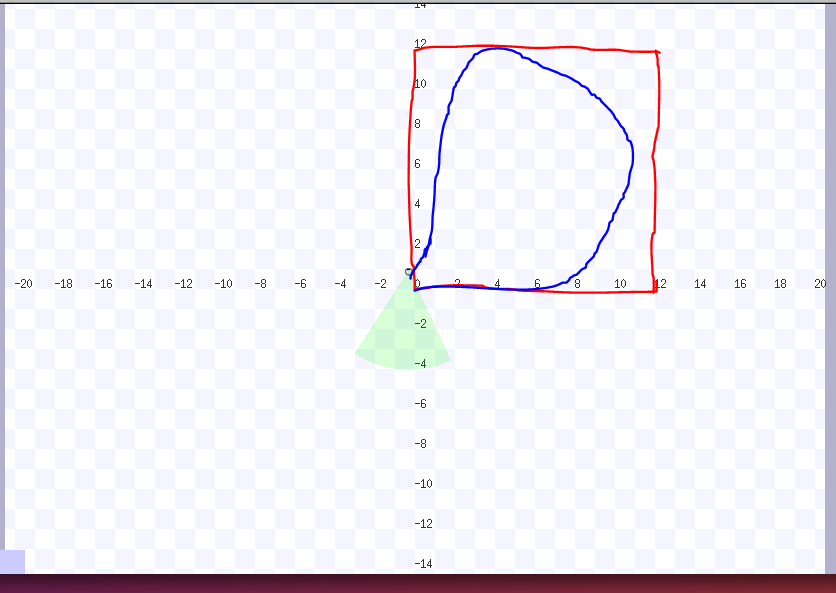
*Change lookahead distance to 5, run same command:*



*Rectangle, size 12, lookahead distance 3:*



*Rectangle, size 12, lookahead distance 5:*



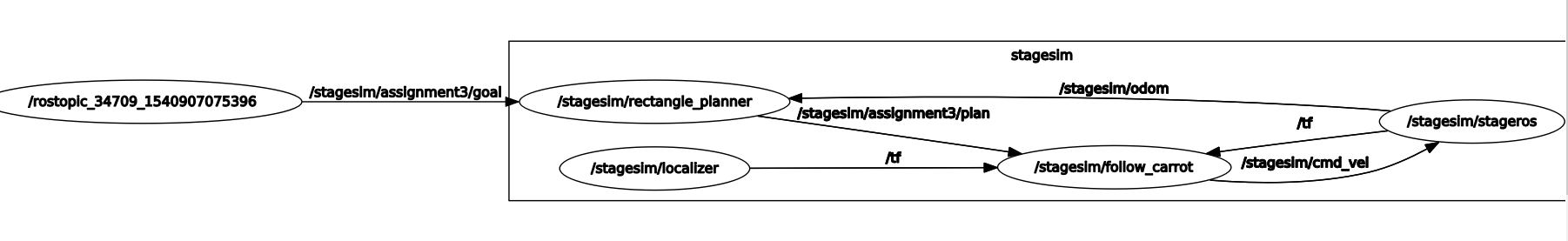
Conclusion:

When you increase the lookahead-distance, the robot will make smoother turns, but stays further away from the corners of both the triangle and rectangle options. This makes sense, as the robot will start turning towards the next Point on its path, once the lookahead distance is equal to the actual distance to the point.

When the lookahead-distance is smaller, the robot will follow the shape of the rectangle/triangle much more.

**How it works**

The RQT graph looks as follow:



From left to right: /rostopic\_34709\_##### is publishing to rectangle\_planner with a message of type assignment3/goal. This is the result of executing ‘ *rostopic pub /stagesim/assignment3/goal assignment3/Goal 12’ .* The rectangle planner listens to this and creates a plan of type assignment3/plan. To create and calculate the points for this plan, /stagesim/odom is used as well. This message is published by the simulator, /stagesim/stageros. When this plan is created, it is published on /stagesim/assignment3/plan.

The follow\_carrot node holds all code to control the movement and position of the simulator robot. Based on /stagesim/stageros’ tf messages, it knows where the robot is and commands it to move further, turn or stop. Commanding the robot is done by publishing on the /stagesim/cmd\_vel topic.

**Choices made:**

We decided that our self-declared message type ‘goal’ should hold just one float to define the size.

**How the code works**

Rectangle\_planner and Triangle\_planner create a Path, which is an array of PoseStamped datatype. Every posestamped has an x, y and orientation. X and Y to define target position, orientation to define the direction that the robot should face after moving to the PoseStamped. This path is published to a topic, and the follow\_carrot node listens to this topic.

Follow\_Carrot node reads this Path and extracts the multiple PoseStampeds in the array. For every PoseStamped, it will send the simulator towards this x,y until the PoseStamped is within lookahead-distance. When the robot is within reach, it will start moving towards the next PoseStamped in the array.

The simulator is controlled in a servoing mode, which means that its current odometry is used to calculate distances to the target Positions, which are the corners of the triangle or rectangle.