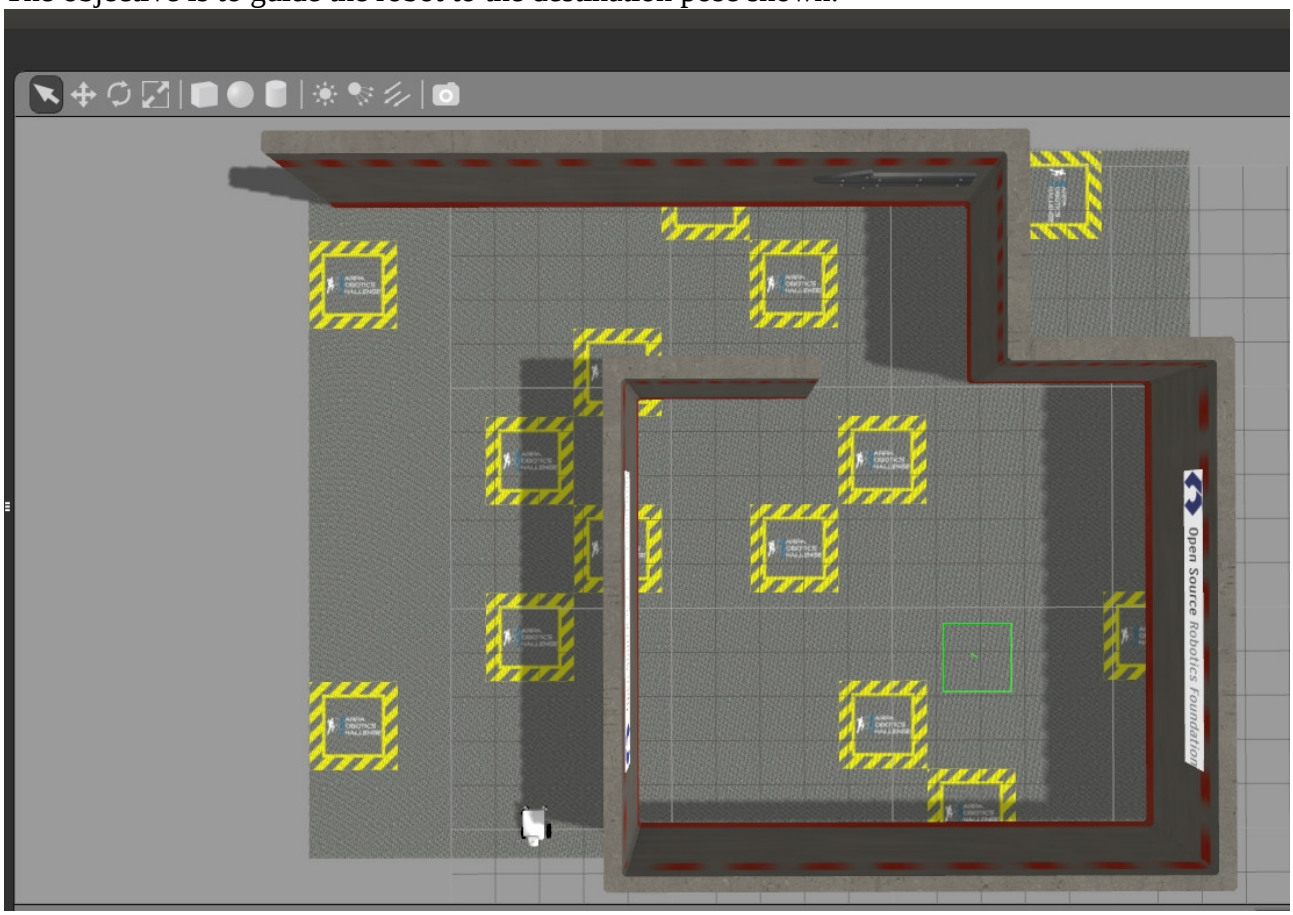
EECS 375/475 PS8:
Simulation of open-loop vs linear steering feedback control

In this problem set, you will compare open-loop control to a linear steering feedback controller. For the simulation, launch the "mobot" in Gazebo with:

roslaunch mobot_urdf mobot_in_pen.launch

The objective is to guide the robot to the destination pose shown:



Start with open-loop control as follows.

In package mobot_pub_des_state, edit the file pub_des_state_main.cpp and comment out lines 11 and 12:

```
//desStatePublisher.append_path_queue(5.0,0.0,0.0);
//desStatePublisher.append_path_queue(0.0,0.0,0.0);
```

Recompile, then start this node:

rosrun mobot_pub_des_state mobot_pub_des_state

The robot should not move, but the desired-state publlisher is ready to accept commands.

Modify a command file. From package mobot_pub_des_state, you can edit the file "pub_des_state_path_client.cpp" to enter via points that provide a pat for the robot to exit the starting pen and end up at the goal location.

Try running your plan open loop. To do so, start up an open-loop controller with:

rosrun mobot_pub_des_state open_loop_controller

This simple node merely copies the speed/spin values of the desired-state publication, and sends them verbatim to the robot as an equivalent cmd_vel publication.

Finally, run your modified client node to command your new path:

rosrun mobot_pub_des_state pub_des_state_path_client

Describe the results of this effort. Are you able to get the robot to converge on the desired goal location?

Next, compare this to motion with steering feedback. Instead of running open-loop control, try running linear control, as follows.

First, restart the simulation:
roslaunch mobot_urdf mobot_in_pen.launch

Start the desired-state publisher:
rosrun mobot_pub_des_state mobot_pub_des_state

Now, instead of running the open-loop controller, start a linear steering controller:
rosrun lin_steering lin_steering_wrt_odom

Finally, send out a path command with your modified client:
rosrun mobot_pub_des_state pub_des_state_path_client

Are you able to get the robot to converge to the desired location?

In the steering-algorithm header file, “steering_algorithm.h”, there are multiple “magic” numbers:
const double K_PHI= 10.0; // control gains for steering
const double K_DISP = 3.0;
const double K_TRIP_DIST = 1.0;
// dynamic limitations: these apply to the steering controller; they may be larger than the limits on des state generation
const double MAX_SPEED = 1.0; // m/sec; adjust this
const double MAX_OMEGA = 1.0; //1.0; // rad/sec; adjust this

Try varying these values and see if you can make the robot get to its goal faster and/or follow its trajectory more precisely.

Document your results (e.g. using rqt_plot for trajectory following performance).