PS3: Path Service Demonstration in STDR

Kristina Collins, kvc2@case.edu

February 16, 2017

1 Introduction

This assignment presents an example of planned pathing in the ROS STDR (Simple Two-Dimensional Robot) Simulator. Like PS1, the assignment uses open-loop control, and serves chiefly as an example of a ROS service between a service node and client node.

2 Demonstration Steps

In a terminal window, type:

- roslaunch stdr_launchers server_with_map_and_gui_ plus_robot.launch
- rosrun kvc2_path_service kvc2_path_service
- rosrun kvc2_path_service kvc2_path_client

This will open the STDR simulator and run the robot around the maze according to its intended path. It's a good idea to use the grid tool in STDR to check the robot's position.

3 Notes and Evaluation

The key addition to the sample code was the get_yaw_and_dist function, which uses the distance formula to calculate the length traveled between two points and takes the arctangent of the X and Y differences to calculate the requisite yaw angle.

```
118 // New function to compute distance and heading from start to goal:
119 void get_yaw_and_dist(geometry_msgs::Pose current_pose, geometry_msgs::Pose goal_pose,double &dist, double &heading) {
120  double x_dist = goal_pose.position.x - current_pose.position.x;
121  double y_dist= goal_pose.position.y - current_pose.position.y;
122  double z_dist= goal_pose.position.z - current_pose.position.z; //just for completeness
123  dist = sqrt(pow(x_dist, 2) + pow(y_dist, 2) + pow(z_dist, 2)); //Good ol' distance formula
124  if (dist < g_dist_tol) { //too small of a motion, so just set the heading from goal heading
125    heading = convertPlanarQuat2Phi(goal_pose.orientation);
126  }
127  else {
128     heading = atan2(y_dist, x_dist); //trig happens! only good in two dimensions, though.
129  }
130 
131 }
132
```

4 Conclusion

As with PS1, the robot doesn't always make it to the top corner, but it's a lot nicer to pick out spots on the map and have the computer calculate yaw than to have to do the math on one's own. Future work (attempts are evidenced by vestigial code in the service node) would include subscribing to the /robot0/odom topic and incorporating the robot's current position into path calculations, rather than accumulating estimates for each path step - in other words, implementing crude open-loop control. However, this method wouldn't be adequate for avoiding obstacles unless it were paired with reactive control, as in PS3. It looks like PS4 will be geared towards accomplishing this.

The soundtrack for this video is Cage the Elephant's "Tiny Little Robots." Yes, this is going to be a thing.