EECS 476 Mobile Robotics
PS 3

Peng Xu
pxx37@case.edu

1. Main idea

To drive the robot to the left top corner, basically we need to build a path by constitute a sequence of state information including headings and travel distances. By adding in a series of values as following codes, the path is built into vector variables and stored into a geometry_msgs::Pose, pose.

```
 //create some path points...this should be done by some intelligent algorithm, but we'll hard-code it here
geometry_msgs::PoseStamped pose_stamped;
geometry_msgs::Pose pose;

// initialization
pose.position.x = 0.0; // say desired x-coord is 1
pose.position.y = 0.0;
pose.position.z = 0.0; // let's hope so!
pose.orientation.x = 0.0; //always, for motion in horizontal plane
pose.orientation.y = 0.0; // ditto
pose.orientation.z = 0.0; // implies oriented at yaw=0, i.e. along x axis
pose.orientation.w = 0.0; //sum of squares of all components of unit quaternion is 1
pose_stamped.pose = pose;

// subgoal 1
quat = convertPlanarPhi2Quaternion(PI/2); // get a quaterion corresponding to this heading
pose_stamped.pose.orientation = quat;
pose_stamped.pose.position.y = 3.0; // say desired y-coord is 1.0
path_srv.request.nav_path.poses.push_back(pose_stamped);

// subgoal 2
quat = convertPlanarPhi2Quaternion(0); // get a quaterion corresponding to this heading
pose_stamped.pose.orientation = quat;
pose_stamped.pose.position.y = 6.5; // say desired y-coord is 1.0
path_srv.request.nav_path.poses.push_back(pose_stamped);

// subgoal 3
quat = convertPlanarPhi2Quaternion(PI/2); // get a quaterion corresponding to this heading
pose_stamped.pose.orientation = quat;
pose_stamped.pose.position.y = 6; // say desired y-coord is 1.0
path_srv.request.nav_path.poses.push_back(pose_stamped);

// subgoal 4
quat = convertPlanarPhi2Quaternion(PI); // get a quaterion corresponding to this heading
pose_stamped.pose.orientation = quat;
pose_stamped.pose.position.y = 3.2; // say desired y-coord is 1.0
path_srv.request.nav_path.poses.push_back(pose_stamped);

// subgoal 5
quat = convertPlanarPhi2Quaternion(PI/2); // get a quaterion corresponding to this heading
pose_stamped.pose.orientation = quat;
pose_stamped.pose.position.y = 3.2; // say desired y-coord is 1.0
path_srv.request.nav_path.poses.push_back(pose_stamped);

// subgoal 6
quat = convertPlanarPhi2Quaternion(PI); // get a quaterion corresponding to this heading
pose_stamped.pose.orientation = quat;
pose_stamped.pose.position.y = 3; // say desired y-coord is 1.0
path_srv.request.nav_path.poses.push_back(pose_stamped);
```

Once the path is built in the client end, we use the following command to send the message to the service end.

```
client.call(path_srv);
```

Then with this server/client scheme, the robot would receive the path information, which is like being guided by sensor information in real situations.

2. Example use

   To start the simulator with

   ```
   $ roslaunch stdr_launchers server_with_map_and_gui_plus_robot.launch
   ```

   Run the alarm function:

   ```
   $ rosrun my_ros_service my_path_service2
   ```

   Run a simple, open-loop command sequence with:

   ```
   $ rosrun my_ros_service my_path_client
   ```

Feb. 9 2016