PS4 Action Server for Motion Control

Zhiang Chen

1. action message

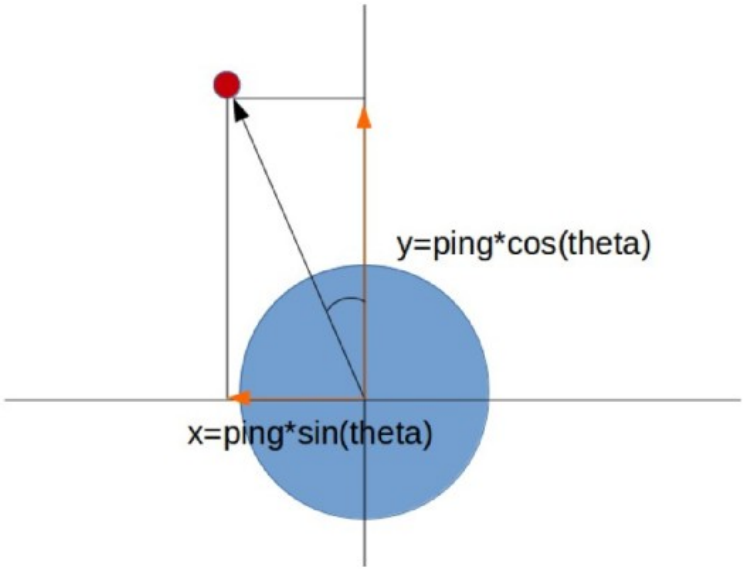
The topic of the action service is “path_action”, and the action structure is defined in path_cmd.action:

```
nav_msgs/Path nav_path
---
bool rslt
---
int32 point
```

The feedback contains the information that segment path the robot is executing.

2. ps4_lidar_alarm

It subscribes to the topic, “scan”, from gazebo. And calculate whether there is any obstacles at front. It also preserves a safe distance for making a turn. When it detects some obstacles, it would publish the alarm to “lidar_alarm”.



When there is no obstacle, it would calculate the shortest distance that the mobot can reach ahead.

```
double theta=abs(-3.14159/angle+angle_increment_*i);
if(laser_ranges[ping_index_min+i]*sin(theta)<radius)
{
dist=(laser_ranges[ping_index_min+i]-radius)*cos(-3.14159/angle+angle_increment_*i)+radius;
distances.push_back(dist);
}
```

theta is the absolute angle between the obstacle and x axis with respect to robot frame. Then if x in the above figure is less than the radius of the robot, it would block the robot's way ahead. And its y value is preserved in the vector distances. Then the smallest one would be gotten from this vector as the shortest safety distance.

3. ps4_action_server

This node is adapted from ps3_path_server. There are two main differences. First, ps4_action_server responses to the action client instead of service server. And its moving mission can be canceled when

there are some obstacles at front. Second, it moves in a relative reference frame instead of absolute reference frame.

When ps4_lidar_alarm detecting the obstacles, it preserves a pace for mobot to make a turn. Thus, when there are some obstacles at front, it only requires the moving movements canceled, while the spinning movement is not affected.

4. ps4_action_client

The action client set a goal for a square path. After sending out the goal, it is checking the whether there is any alarm from “lidar_alarm”. When it receives the alarm signal, it immediately cancel the goal to halt the mobot. And then make a turn until there is no obstacle at front. Then it requests the goal.

Since there is a noise in the motion, I add a calibration for the spinning:

```
const double cabli = 20;
```

Another function is set to design an equilateral geometry for the path.

```
void set_geometry(ps4_action_server::path_cmdGoal &goal, double length, int num)
{
   geometry_msgs::Quaternion quat;
   geometry_msgs::PoseStamped pose_stamped;
   geometry_msgs::Pose pose;

   double angle= (num-2.0)*180/num+cabli;
   goal.nav_path.poses.clear();

   for(int i=0; i<num; i++)
   {
      pose=initPose();
      setPosition(length,0.0,0.0,pose);
      setOrientation(angle,pose);
      pose_stamped.pose = pose;
      goal.nav_path.poses.push_back(pose_stamped);
   }
}
```