ASSIGNMENT 2 - ROS Kinematics of a DDMWR

Nathan Ramos & Quinn Frady

Description of the problem

The goal of this assignment is to develop a ros package to simulate the forward and inverse kinematics of a differential drive mobile robot. We were allowed to choose our own robot parameters for two tasks that are incrementally added to a developed ros package.

Task 1

The goal of task 1 was to develop and test a ros package to simulate the forward kinematics of differential drive mobile robot. Task 1 wanted the following ros topics:
/cmd_vel - type Twist
/pose2D - type Pose2D

To execute this assignment, we created our package da2_1 in the workspace da2 that references std_msgs, roscpp, and geometry_msgs for its dependencies. Afterward we created our file for the first node, pub_cmd_vel.cpp which was designed to periodically publish random velocities in the linear.x and the angular.z direction. The published message was of type geometry_msgs::Twist to the aforementioned topic /cmd_vel. Then, we created the controller node in the file controller_cmd_vel_and_pose.cpp which was designed to subscribe to /cmd_vel and then publish the current pose of type geometry_msgs::Pose2D to the topic /pose2D. The current pose was calculated by following forward kinematic math to calculate the change in time, distance (x and y), and theta to determine the distance traveled and add that to the pose values.

To execute this assignment you must run the following steps

1. Catkin_make
2. roscore
3. roslaunch da2_1 step1.launch
4. View the data in plotjuggler

Task 1 Source Code Directory

DesignAssignments/da2/src/da2_1/src/

Task 1 Video

https://www.youtube.com/watch?v=q0yKOK77_AE&list=PLZTXnWnnMe9eMQkYrS3KXLsXxmp48Bh74&index=2

Task 2

The goal of task 2 was to develop and test a ros package to simulate the inverse kinematics of a differential drive mobile robot. Task 2 wanted the following ros topics:

/goal_pose - type Pose2D
/cmd_vel - type Twist
/current_pose - type Pose2D

The input of the robot is /goal_pose, which sends x,y, and angle goals for the robot to achieve. We developed pub_pose.cpp to publish a random /goal_pose every time it is run. From there, ctrl_pose.cpp activates multiple nodes: a subscriber to the /goal_pose topic, and a publisher of /cmd_vel and /current_pose. Within ctrl_pose.cpp, the program waits for the first callback from /goal_pose, and stores the goal x, y, and theta values for /current_pose to achieve.

It then activates phase 1 of the robot, which calculates the theta needed for the robot to point towards the coordinates of the goal pose from its (0,0) starting position. The robot then turns on its angular velocity Z and calculates the theta with respect to time until it matches the target angle of approach, turning off angular velocity Z.

From there, it enters phase 2, which calculates the distance the robot needs to travel to reach the goal coordinates. The linear velocity X is then turned on, allowing the robot to move forward and simultaneously maps its distance traveled to X and Y components until it reaches the desired distance. The linear velocity X is then turned off and at this point the robot's coordinates should match the goal pose's coordinates.

It then enters phase 3, which reactivates the angular velocity Z until its calculated theta matches the orientation of the goal pose. Afterwards, angular velocity Z is turned off and the robot comes to rest. While each phase is in motion, the robot publishes its velocities and poses to /cmd_vel and /current_pose at a frequency of around 500 Hz.

To execute this assignment you must run the following steps:

1. catkin_make
2. roscore
3. rosrun da2_2 pub_pose
4. rosrun da2_2 ctrl_pose
5. view the data in plotjuggler

Task 2 Source Code Directory

DesignAssignments/da2/src/da2_2/src/

Task 2 Video

https://youtu.be/IY1kgO5NeV0?si=6Cst2u7jpjFgDc3V