Pose to Pose Control

1 Problem Statement

Design a controller that drives the robot from a known location $\begin{bmatrix} x_o & y_o & \theta_o \end{bmatrix}^T$ to a given location $\begin{bmatrix} x_g & y_g & \theta_g \end{bmatrix}^T$. The robot's kinematics are given by the following model:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{1}$$

Where, x, y and θ represent the state of the robot. v and ω are the linear and angular velocities of the robot.

2 Software Description

Using ROS 2 and Gazebo Gz as the middleware and simulator, a package "limo_simulation" is given.

Develop a C++ node in "limo_control" package that would publish to "cmd_vel" with the message type "geometry_msgs/Twist" for controlling the robot, and subscribe to "odom" with the message type "nav_msgs/Odometry" for feedback.

Take a look at the README file in the repository for more information.

Note: This is a standard setup, feel free to modify it.

3 Output Requirements

The performance gauge for this task primarily includes fundamentals of robotics, and software proficiency.

- Modular software is expected, including but not limited to clean and efficient code, and scripts that help run the code without too many commands.
- The controller should reduce the euclidean distance error and absolute orientation error below 5 centimeters and 0.1 radians at the very least. Provide plots to defend your code output for this requirement.

- The ability to make changes post development when requested for. For example, a request to add safety limits to the controller output.

Please watch this video for the minimum expected result.

Note: Modular software is a vague definition, but it will help us understand how aware you are of standard practices!