CS 81 - Principles of Robot Design & Programming

Carter Kruse

Homework 3 - PID Controller

Purpose

In this assignment, you will put in practice the concepts of the PID controller.

Assumptions

Assume that you have the Husarion ROSbot 2.0 robot with a 360 degree field of view LIDAR (the measurement is stored in the ROS message LaserScan), modeled as a differential drive robot. Assume that the robot can be controlled via a ROS Twist message. The linear velocity is set to a constant 0.5 m/s, while you can control the angular velocity.

General Instructions

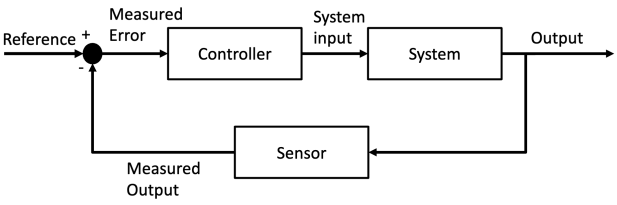
Answer to the following questions, showing all the relevant drawings and intermediate math work by uploading a PDF file containing the work. You can include code written by yourself if used for doing some of the repeated calculations, but write down the formulas used on paper. Feel free to add comments in the text box.

Question 1

Consider the wall following problem. The robot must keep following the wall on the right at a given distance (assume that the robot is in an infinitely long corridor, and the robot will not start facing the wall).

Write the model for a general closed-loop controller. In particular, the following elements should be written down explicitly: the state, the actuation command, and the error.

Closed loop control or *feedback control* is a type of controller that computes the system input using a sensor to measure the error that is taken into account.



The goal of a feedback controller is to achieve and maintain a desired state (*set point*) by using the information coming from the sensor(s).

Error is the difference between the current and desired state. *Sampling rate* is the frequency at which the sensors read new information that can be used to compute the error.

Notation

- $X\,(t)$ = State Space
- $x\,(t)$ = State
- $U\,(t)$ = Input/Action/Control Space
- $u\,(t)$ = Input/Action/Control
- $e\,(t)$ = Error

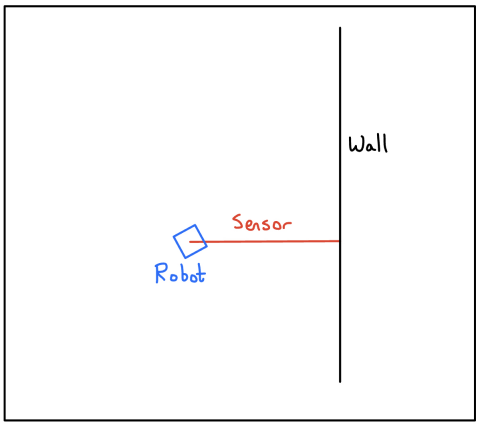
In the wall following problem, where the robot must keep following the wall on the right at a given distance, the following may be specified.

State: The pose (position and orientation) of the robot, particularly the distance from the wall, given as the x-position. This is determined by the 360^{o} LIDAR sensor (which stores measurements in the ROS message LaserScan).

Actuation Command: The changing of the angular velocity (ω) (determined by the error and a given controller), which modifies the pose (position and orientation) of the robot. This is set/controlled via a ROS Twist message, with the linear velocity set to a constant $0.5\,m/s$

Error: The distance from the robot to the set point, which is equivalent to difference between the x-position of the robot and the goal location (x-coordinate).

Draw the robot, the wall, and the sensor readings that are useful to understand your model.



The robot may be oriented in any direction, with any position. The sensor reading is interpreted to be the perpendicular distance from the center of the robot to the wall, which is given from the LIDAR sensor on the robot.

Specify how you will use the sensor to measure the state.

As indicated the 360^{o} LIDAR sensor will be used to meausre the state of the robot, which is it's perpendicular distance from/to the wall.

As a result of using the perpendicular distance, the angle of the LIDAR measurement (in the robot reference frame) will change as the robot is oriented differently (at a different angle), so adjustments must be made as the robot's pose is modified.

In the case where the distance is less than or greater than a given value (the desired state), the robot will respond accordingly.

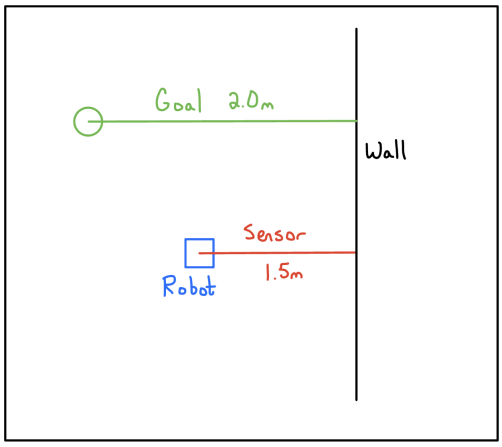
Question 2

A discrete P controller is used to control the robot with a proportional gain of 1. The robot starting pose is exactly parallel to the wall and the laser sensor returns a measurement of $1.5\,m$ to the right of the robot.

Given this discrete P controller, write how long it will take to get the robot to the target distance of $2m$ from the wall, using the model you defined in Question 1, showing your calculations and the error over time, at a sampling rate of 0.2 seconds. Assume that the robot follows the forward kinematics model for the differential drive robot seen in class.

Given a discrete P controller that is used to control the robot with a proportional gain of 1 (i.e. $K_p = 1$), we may model the process in which the robot reaches the target distance of $2\,m$ from the wall. As stated, the robot starting pose is exactly parallel to the wall, and the laser sensor returns a measurement of $1.5\,m$ to the right of the robot.

This is demonstrated by the following image:



To calculate how long it will take for the robot to reach the target distance, let us consider a forward kinematics model for a differential drive robot, with the following:

Time (s): The time for each "iteration" is determined by the sampling rate, which is $(0.2\,s)$.

X-Location, Y-Location (m): The x-location and y-location (in m) give the position of the robot in a global reference frame. The origin is located at the point on the wall closest to the robot when it starts and the axes are aligned with the North and East compass directions, following the right-hand rule. Thus, the (x-location, y-location) starts at $(-1.5\,m, 0\,m)$.

Goal (m): The goal gives the x-location of the target distance, which is $(2\,m)$ from the wall, which is $(-2\,m)$ in the global reference frame.

Error (m): The error gives the difference between the x-location of the robot and the goal location.

$$(Error = XLocation - Goal)$$

Angular Velocity (rad/s): The angular velocity is updated at every "iteration" and thus is expressed using the following model $u_k = K_p e_k$ with a proportional gain $K_p = 1$.

$$(Angular\,Velocity = Error)$$

Linear Velocity (m/s): The linear velocity remains constant throughout the motion of the robot, at $(0.5\,m/s)$.

Radius (m): To calculate the radius of the circular arc that the robot traces out at every "iteration", we may use $v = \omega r$, where v represents the linear velocity (in m/s), ω represents the angular velocity (in rad/s), and r represents the radius (in m).

$$(Radius = Linear\,Velocity/Angular\,Velocity)$$

Theta (rad): The angle θ (in rad) gives the rotation (pose) of the robot in a global reference frame, as defined previously. Thus, the angle starts as $\left(\frac{\pi}{2}\,rad\right)$.

ICC (m): The ICC (instantaneous center of curvature) gives the point around which the robot moves in a circular motion. In a differential drive model, the ICC may be determined by $(ICC_x, ICC_y) = (x - R\sin(\theta), y + R\cos(\theta))$.

$$(ICC_x, ICC_y) = (X\,Location - Radius\sin(Theta), Y\,Location + Radius\cos(Theta))$$

New X-Location, New Y-Location (m), New Theta (rad): The new x-location, y-location, and angle (theta) of the robot may be calculated using the forward kinematics model, given the current state (x, y, θ) and the action $(\omega, \Delta t)$. This allows us to determine the pose of the robot at each iteration.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) & 0 \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega\Delta t \end{bmatrix}$$

$$\begin{bmatrix} New\,X Location \\ New\,Y Location \\ New\,Theta \end{bmatrix} = \begin{bmatrix} \cos(Angular\,Velocity * Time) & -\sin(Angular\,Velocity * Time) & 0 \\ \sin(Angular\,Velocity * Time) & \cos(Angular\,Velocity * Time) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X Location - ICC_x \\ Y Location - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ Angular\,Velocity * Time \end{bmatrix}$$

The calculations described above are included in this spreadsheet. Given this discrete P controller, it will take slightly less than 2.4 seconds for the robot to reach the target distance of $2\,m$ from the wall, using the defined model.

The calculations and error over time (with a sampling rate of 0.2 seconds) are provided for the forward kinematics model of the differential drive robot.

Question 3

Considering the previous scenario, write whether the robot is overshooting, motivating your answer. If you believe it will overshoot, write an option to avoid overshooting and write the calculations using that option, simulating the scenario.

Yes, the robot is overshooting, as evidenced by the graph produced for the previous section. While the proportional term accounts for present errors, allowing the state to reach the target reference, the state tends to overshoot the target reference in a P controller.

In other words, the control input is not reduced far enough in advance of the target being reached, as future errors (or lack thereof) are not accounted for.

This is primarily due to the (proportional) adjustment in the angular velocity failing to rectify the situation in which the robot moves quickly to reach the target reference, with no consideration of the error that may be produced after this has been accomplished. Intuitively, the robot is unable to make a sharp turn (in a short time interval) to immediately remain in line, following the wall to the right at a given distance.

Alternatively, we may think about this as follows. The robot has a constant linear velocity, while the angular velocity is determined by the controller. By the time that the control command is re-evaluated (according to the sampling rate), the overshooting will have already happened. Further, even if when re-evaluating the control command the robot is exactly at the reference, the robot exists in the physical world, and thus is subject to inertial forces.

The control function is not responsible for the robot's position, and only adjusts the robot's orientation, which implicitly changes the robot's position (over a period of time). In this particular case, the robot will not be perfectly aligned with the angle of the line it is trying to follow by the time it reaches the line, and thus will move past the line when continuing to move forward.

To address the oscillation problem (overshooting) that develops as a result, we need to reduce the control input well before the target is approached. To do so, we may predict the error in the near future with a derivative term, which works as long as the error does not oscillate at a high frequency.

The introduction of a derivative term accounts for future errors, creating a PD controller as follows:

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t) \qquad u_k = K_p e_k + K_d \frac{e_k - e_{k-1}}{\Delta t} \text{ (Discrete Version)}$$

The only calculation that changes from the initial spreadsheet is as follows:

Angular Velocity (rad/s): The angular velocity is updated at every "iteration" and thus is expressed using the following model $u_k = K_p e_k + K_d \frac{e_k - e_{k-1}}{\Delta t}$ with a proportional gain $K_p = 1$ and a derivative gain $K_d = 2.3$.

$$\left(Angular\,Velocity = Error_k + K_d * \frac{Error_k - Error_{k-1}}{Time} \right)$$

The calculations are included in this spreadsheet. Given this discrete PD controller, it will take slightly less than 7.2 seconds for the robot to reach the target distance of $2\,m$ from the wall, using the defined model.

The calculations and error over time (with a sampling rate of 0.2 seconds) are provided for the forward kinematics model of the differential drive robot.

By introducing the derivative term with a gain of $K_d = 2.3$, the overshooting is mostly eliminated and the error (relatively) quickly drops to zero. As a result, the robot follows the wall to the right at a distance of $2\,m$ consistently, with no oscillation.

Question 4

Considering the previous scenario (Question 2), write what happens if the sampling rate becomes 0.4 seconds, instead of 0.2 seconds. Write the calculations and show the error over time.

If the sampling rate is adjusted to 0.4 seconds, the robot undergoes a similar process, though the "iteration" is slightly different, as the updated angular velocity is modified less often. This results in a different robot pose (position and orientation) compared to a sampling rate of 0.2 seconds.

The calculations from the initial spreadsheet do not change, as only the time (given by the sampling rate) changes.

The calculations are included in this spreadsheet. Given this discrete P controller, it will take slightly less than 2.4 seconds for the robot to reach the target distance of $2\,m$ from the wall, using the defined model.

The calculations and error over time (with a sampling rate of 0.4 seconds) are provided for the forward kinematics model for the differential drive robot.

If we adjust the sampling rate from $0.2\,s$ to $0.4\,s$, the robot reaches the reference target in approximately the same amount of time. However, there are significant drawbacks to modifying the sampling rate.

In particular, the oscillation becomes slightly worse with the less frequent sampling time, as the robot does not update the angular velocity as frequently. This means that for any given iteration, the absolute value of the error is likely to be higher with a less frequent sampling rate. To optimize performance, the sampling rate should be as frequent as possible (under physical constraints).

Further, in extending the simulation to 60 seconds, the oscillation not only continues, but a steady-state error is introduced, leading the robot far from the reference target. This devastating result means that the error over time never reduces to zero, indicating that the robot is unable to follow the wall as expected.

Question 5

Considering the wall following problem above, write if there is any reason to introduce the integral term. Please motivate your answer.

There does not seem to be any reason to introduce the integral term, as there are no systematic errors and the error estimate is not (relatively) noisy.

The integral term in a model is designed to account for past errors, which does not seem to be the case in this simulation. However, if there was known error in the measurements returned from the robot's LIDAR sensor, or in the movement (position/orientation) of the robot, an integral term may be used to correct this issue.

Moreover, in simulation, it seems as though a simple PD controller rectifies the oscillation problems that occur when using a simple P controller. The steady-state error reduces to zero, indicating there is no point to introducing an integral term. By setting the proportional term and derivative terms appropriately, the model functions as expected.

(The proportional term is designed to decrease the rise time, while the derivative term reduces overshoot (oscillations) and settling time.)