Steering Control*

EENG350: Systems Exploration, Engineering, and Design Laboratory

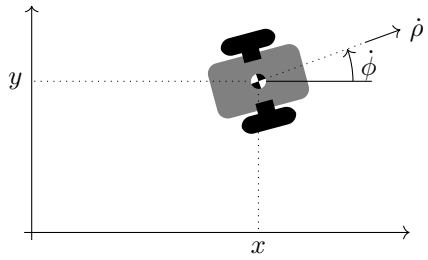
Tyrone Vincent and Darren McSweeny

Department of Electrical Engineering
Colorado School of Mines

Spring 2021

1 Introduction

In this document, we will discuss the problem of steering a vehicle along a desired trajectory. This discussion assumes vehicle with two powered wheels.



There are several ways that you could approach controlling the motion of the vehicle. We will discuss one way of breaking down the problem. Your vehicle has two powered wheels, with the motor voltage as the "raw" actuation variable for each wheel. With two wheels you can both move forward (and reverse) and turn, depending on the relative speed of each wheel. We will continue to use ϕ to define the rotational position of the vehicle relative to the x-axis, so the rotational velocity is $\dot{\phi}$, with positive indicating counter-clockwise rotation when viewing the vehicle from above. The (instantaneous) forward velocity will be denoted $\dot{\rho}$, so that ρ is the distance that the robot travels.

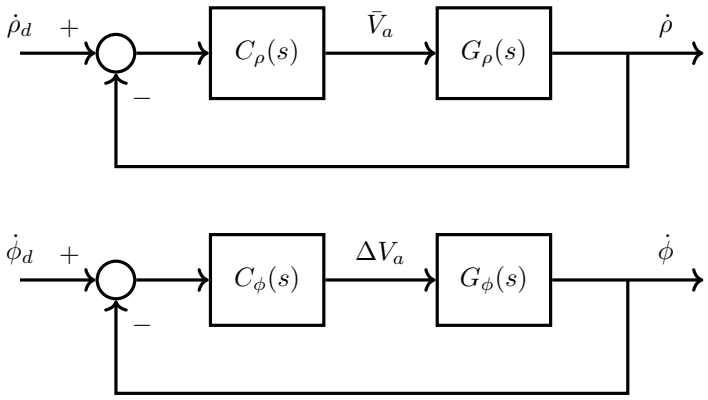
2 Control Strategy

To structure the control design, you can first develop a control system that takes the desired lateral speed and desired turning rate as inputs, and applies the motor voltage in order to achieve these set-points.

As it stands, we need to consider the robot a two-input two-output system. The inputs are the two motor voltages, and the outputs are the forward velocity and rotational velocity. However, we can make a simple transformation which will decouple the effect of the inputs on the outputs[1]. Let $V_{a,1}$ be the voltage applied to motor 1 (assumed to be on the right) and let $V_{a,2}$ be the voltage applied to motor 2 (on the left). (For both motors, we assume the direction pin of the motor driver is utilized so that a positive voltage tends to move the robot forward. Since the motors are mounted differently, the state of the direction pin is probably opposite for each motor.) Set $\bar{V}_a = V_{a,1} + V_{a,2}$ to be the sum of the two voltages, and set $\Delta V_a = V_{a,1} - V_{a,2}$ be the difference. The pseudo-input $\bar{V}_a$ will primarily affect the forward speed of the vehicle and ΔV_a will primarily affect the rotation of the vehicle. We can thus consider two single-input single-output control systems to regulate the forward and rotational speed.

[1] In fact, the decoupling will not be perfect, but any residual effects can be handled as disturbances

This transformation assumes that a positive voltage will tend to rotate the wheels in order to move the vehicle forward. You will want to make sure all your Arduino code is set up to follow this convention. Because the motors are mounted facing opposite directions, the direction pin may need to be inverted for one of the motors, depending on how you wired the motors to the motor driver.



The transfer functions $G_\rho(s)$ and $G_\phi(s)$ define the vehicle dynamics

Forward velocity to sum of motor voltages

$$G_\rho(s) = \frac{\dot{\rho}(s)}{\bar{V}_a(s)}, \quad G_\phi(s) = \frac{\dot{\phi}(s)}{\Delta V_a(s)}.$$

rotational velocity to voltage difference between motors

You will identify these transfer functions using step response experiments.

3 Identification Experiments

To identify the systems $G_\phi(s)$ and $G_\rho(s)$ you can perform step experiments. To perform a unit step input for the $G_\phi(s)$ system, you would choose $\bar{V}_a = 1$, Since we would not want the rotational system to interfere with the experiment, we would also choose $\Delta V_a = 0$. To figure out the actual voltage that we would apply to the motors, we would invert the system of equations

Sum: 255 or 400
Both equal

$$\begin{aligned} \bar{V}_a &= V_{a,1} + V_{a,2}, \\ \Delta V_a &= V_{a,1} - V_{a,2}. \end{aligned}$$

You can check that the answer is $V_{a,1} = \frac{\bar{V}_a + \Delta V_a}{2}$ and $V_{a,2} = \frac{\bar{V}_a - \Delta V_a}{2}$. Thus, for the ρ system step experiment, ($\bar{V}_a = 1$ and $\Delta V_a = 0$) you apply $V_{1,a} = .5$ and $V_{2,a} = .5$ and for the ϕ system step experiment ($\bar{V}_a = 0$ and $\Delta V_a = 1$) you apply $V_{1,a} = .5$ and $V_{2,a} = -.5$. To measure the outputs $\dot{\rho}$ and $\dot{\phi}$, we can use the motor encoders. Assuming no slipping, the velocity of the robot at the left or right wheel is equal to the rotational velocity of the wheel times the radius. Let $\dot{\theta}_1$ be the rotational velocity of wheel 1 in radians per second, and similarly for $\dot{\theta}_2$ (you have calculated these before from the encoder readings to implement your motor controller). Let the radius of the wheels be r. The instantaneous forward velocity, $\dot{\rho}$, will be the average of the velocity of the two sides of the vehicle, so that

$$\dot{\rho} = r\frac{\dot{\theta}_1 + \dot{\theta}_2}{2}.$$

The rotational velocity, $\dot{\phi}$, will be the difference between the velocity of the two sides of the vehicle, divided by the distance between the wheels, which we will call d:

$$\dot{\phi} = r\frac{\dot{\theta}_1 - \dot{\theta}_2}{d}.$$

This transformation assumes that $\dot{\theta}_1$ and $\dot{\theta}_2$ are positive when the wheels spin to move the vehicle forward. You will want to make sure all your Arduino code is set up to follow this convention. Because the motors are mounted facing opposite directions, you may have to change the sign of one of your encoder readings.

Set up your experiment code so that you can set variables representing $\bar{V}_a$ and ΔV_a to desired values, and have the code calculate the corresponding motor voltages and values for analogWrite. Also, have the code calculate and print out $\dot{\rho}$ and $\dot{\phi}$. Use a stopwatch and tape measurer to verify that these calculations are correct. Any errors here will cause major headaches later!

4 Inner Loop Controller Implementation

Depending on your requirements, the controllers $C_\phi(s)$, and $C_\rho(s)$ can be as simple as proportional control. (Hint: there is no need for zero steady state error with respect to the velocity setpoints). Using MATLAB or Simulink, you can test the performance of different proportional gains using the transfer functions identified in the previous step. Once you have designed your controllers $C_\phi(s)$, and $C_\rho(s)$, you should be able to write Arduino code to implement them. You should be able to set a desired forward speed $\dot{\rho}_d$ and desired turning rate $\dot{\theta}_d$, and the robot should move at these rates. The controllers $C_\rho(s)$ and $C_\phi(s)$ have as their output $\bar{V}_a$ and ΔV_a. When implemented in Arduino, you will need to convert back to the raw motor voltages using $V_{a,1} = \frac{\bar{V}_a + \Delta V_a}{2}$, $V_{a,2} = \frac{\bar{V}_a - \Delta V_a}{2}$, and then convert $V_{a,1}$ and $V_{a,2}$ to the required value for `analogWrite` and correct value for the motor voltage sign pin (also called the direction pin.)

Test that these controllers work before going forward. Take step response data of the closed loop system and compare to simulation. A specific test that you can do, which will also be part of Demo 1, is to see if you can have your robot follow a semi-circular path at a specific rate. Suppose you want to have your robot go along a semi-circle of radius r meters in t_0 seconds. Then you would choose forward velocity $\dot{\rho} = \frac{\pi r}{t_0}$ m/s and rotational velocity $\dot{\phi} = \frac{\pi}{t_0}$ rad/s.

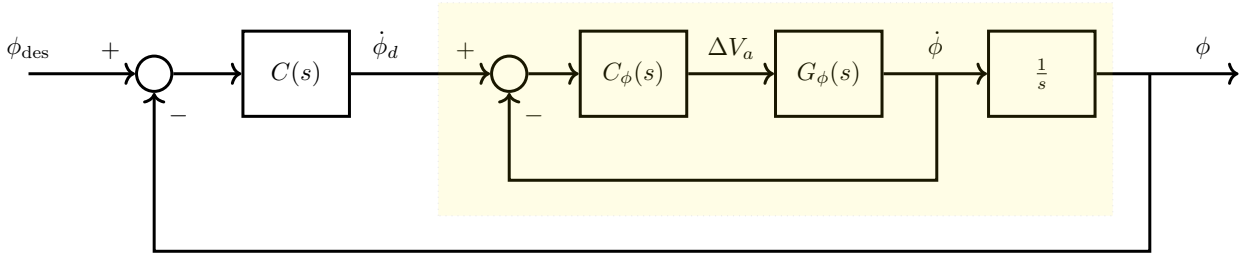
5 Outer Loop Control

An outer control loop will use the error between the estimated and desired value of ϕ (where the desired value ϕ_{des} is set by a command (i.e. "turn 90 degrees left") or by your camera measurement) to choose the desired turning rate $\dot{\phi}_d$. The speed set-point $\dot{\rho}_d$ can be chosen based on other criteria (e.g. in transit, reaching a desired location, etc.)[2].

Note that ϕ is calculated by integrating $\dot{\phi}$, which in turn is estimated from encoder readings. Of course, since we took a derivative of the encoder readings to create $\dot{\phi}$, we could actually just utilize the encoders readings themselves to obtain ϕ. So in your Arduino code, if θ_1 is the reading of encoder 1 **in radians** and θ_2 is the reading of encoder 2 **in radians**, then you can calculate

$$\phi = r\frac{\theta_1 - \theta_2}{d}.$$

A conceptual block diagram of the turning control strategy that you can use for control design would look like the following.



The transfer function $C(s)$ is the outer loop controller, which takes the difference between the desired angle and the measured angle ϕ to set the desired turning rate $\dot{\phi}_d$. A good choice for $C(s)$ is a Proportional-Derivative controller

[2]It would also be possible to design a controller that would use the error on ϕ to set ΔV_a directly. However, as emphasized above, the discussion in this document is only one way to conceptualize the control.

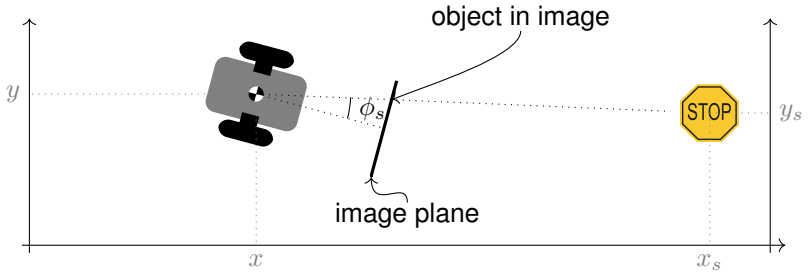
(this is because the system under control in the yellow box is at least second order and is type 1). To design this controller, you will want to find the transfer function of the yellow box,

$$G(s) = \frac{C_\phi(s)G_\phi(s)}{(1 + C_\phi(s)G_\phi(s))s},$$

and use this as the "plant" for the control design process to determine $C(s)$. If this plant is third order or higher, you should use either the root locus or frequency response methods for designing the controller.

6 Testing a control system for moving towards an object

Once you have a controller that can turn to a particular angle and move a certain distance, you can use this controller to move towards an object.



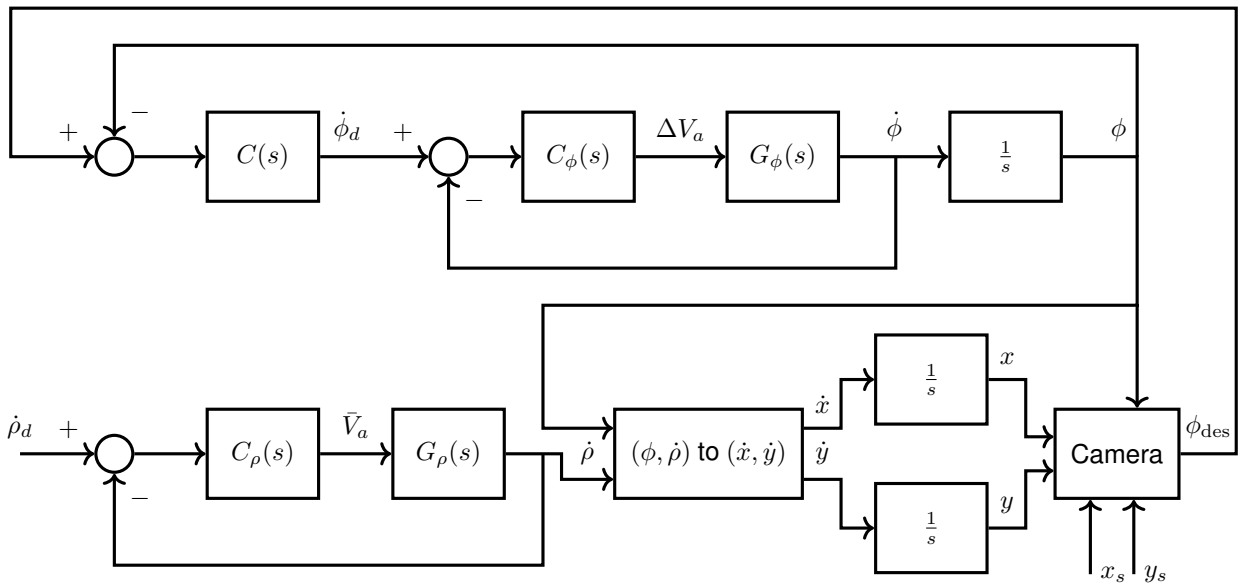
If a camera is mounted on the center of the front of the vehicle, and the object is found in the image, then the difference between the (horizontal) pixel corresponding to the center of the object in the image and the pixel corresponding to the center of the image can be used to determine the angle toward the object, ϕ_s. (You will have to know the field of view of the camera to do the conversion, or you can calibrate your camera by taking a picture of an calibration object, like a ruler, at a known distance.) In this document, ϕ_s is assumed to be positive in the counter-clockwise direction, so that it is positive when the object is in the left part of the image, and negative when the object is in the right part of the image.

If the vehicle is currently at location x, y and the object is at location x_s, y_s, then some trigonometry gives us

$$\phi_s = \text{atan2}\left(y_s - y, x_s - x\right) - \phi \tag{1}$$

where atan2 is a four-quadrant inverse tangent, which separates the x and y axis arguments to get the correct answer in all four quadrants, and ϕ is the angle of the robot with respect to the x axis.

We can use Simulink to model the system and controller acting on the measurement of ϕ_s from the camera. The following block diagram has a more accurate representation of the system motion and camera measurement. It also produces the position of the vehicle (x, y) which you can use to animate the results.



The conversion block $(\phi, \dot{\rho})$ to $(\dot{x}, \dot{y})$ implements

$$\dot{x} = \dot{\rho}\cos(\phi),$$
$$\dot{y} = \dot{\rho}\sin(\phi).$$

The location of the object is set as x_s along the x-axis and y_s along the y-axis. The camera block returns the angle from the center of the camera to the location of the object ϕ_s, calculated as $\phi_s = \text{atan2}\left(y_s - y, x_s - x\right) - \phi$, and sets the desired angle as

$$\phi_{\text{des}} = \phi + \phi_s$$

Where ϕ is the robot rotation at the time the camera measurement was made.