ECE 3 Fall 2020 Lab Section 5 Notes – Feedback Control

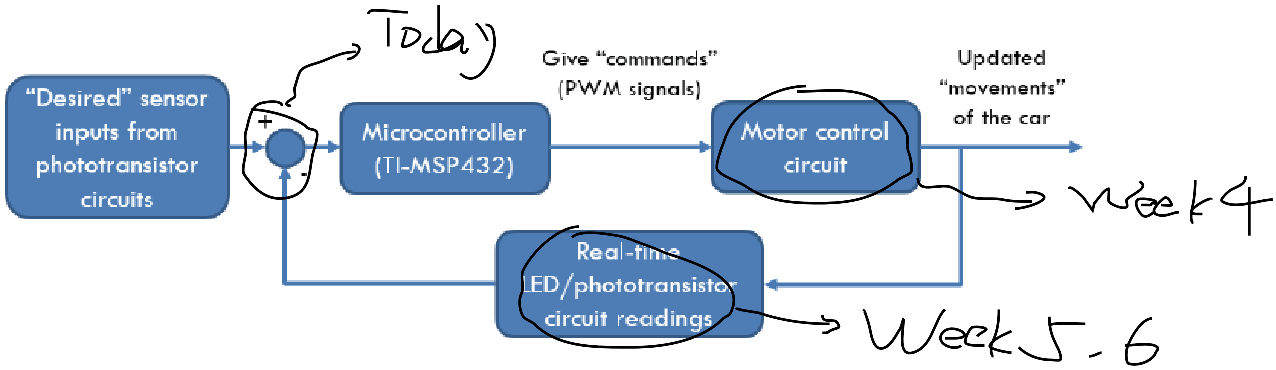


FIGURE 3-1: High level understanding of the ECE3 class project

<u>Sensor Fusion and the Intuition Behind:</u>

Problem of the "guardrail" strategy:

void setup() {

……

}

void loop() {

(1) Read raw sensor data: $[R_0, R_1, \ldots, R_7]$

(2) Pre-process the data (calibration/normalization):

$$\left(\begin{array}{c} (R_0 + Offset_0) \times Scale_0 = S_0 \\ (R_1 + Offset_1) \times Scale_1 = S_1 \\ \ldots \\ (R_7 + Offset_7) \times Scale_7 = S_7 \end{array} \right)$$

(3) Change motor speeds based on sensor inputs with fixed thresholds:

…

if (*some sensors "see" black (>some thresholds) and other sensors "see" white (<some thresholds)*)

conditions is a small finite number

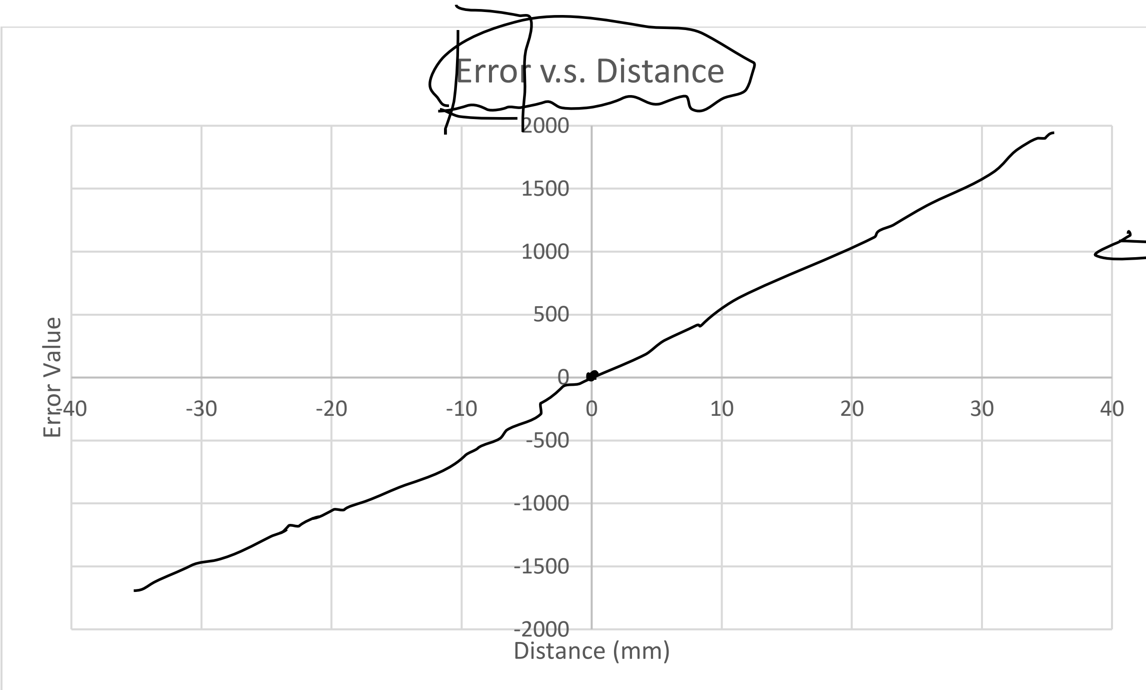
left_pwm = *some number;*
right_pwm = *some number;*

analogWrite (left_pwm_pin, left_pwm);
analogWrite (right_pwm_pin, right_pwm)

else if
…

Intuition of the sensor fusion idea: Can we extract a figure of merit that <u>represents the real time signed distance between the car and the track</u> from the 8 sensor readings? (error value)

Ideal error value v.s. distance plot:



How do we set this "error" value?

Benefit:

Guardrail

…
if (*some sensors "see" black (>some thresholds) and other sensors "see" white (<some thresholds)*)

left_pwm = *some number;*
right_pwm = *some number;*

analogWrite (left_pwm_pin, left_pwm);
analogWrite (right_pwm_pin, right_pwm)

else if
…

Feedback

…

left_pwm = left_base_speed ± k*error
right_pwm = right_base_speed ∓ k*error

Constant

Pay attention to the polarity here.

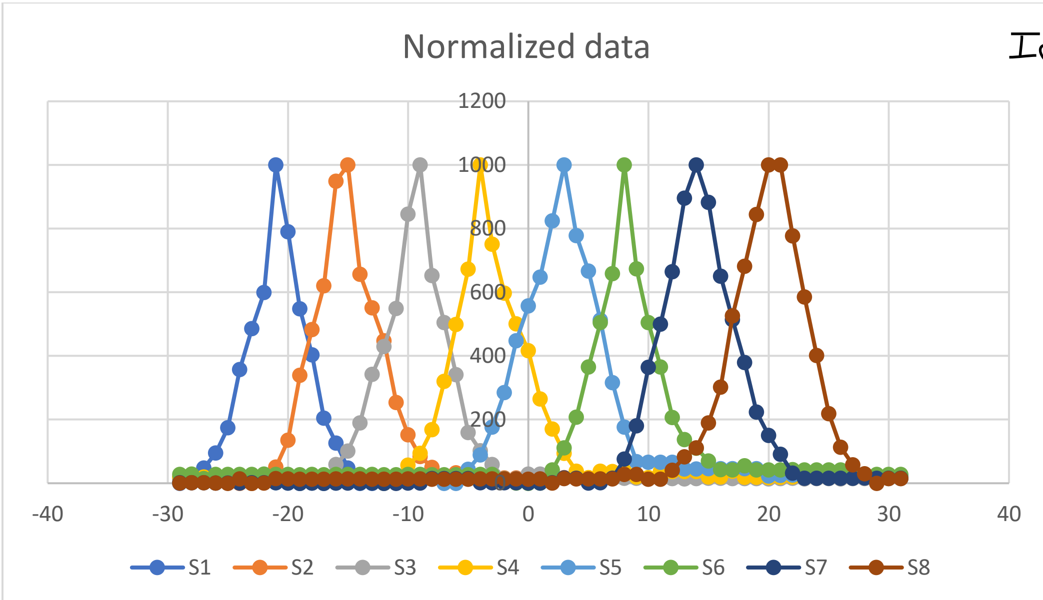
analogWrite (left_pwm_pin, left_pwm);
analogWrite (right_pwm_pin, right_pwm)

…

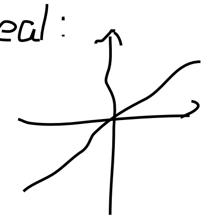
Proportional Control

Sensor Fusion

After calibration

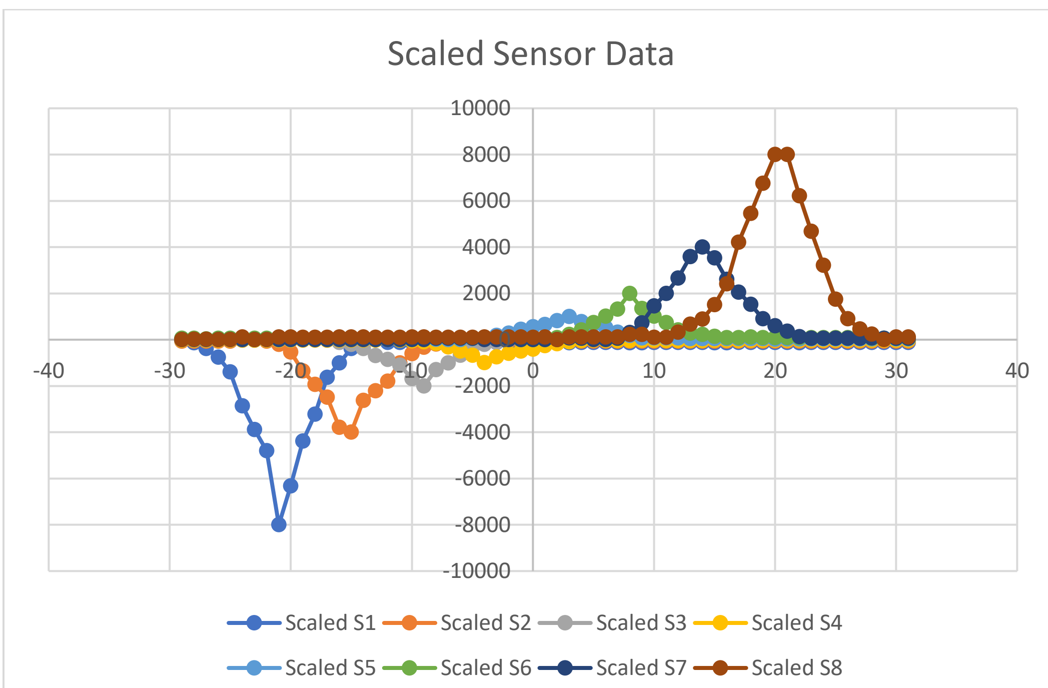


Ideal:



Weight factor 8-4-2-1 (-8S1+-4S2+-2S3+-1S4+1S5+2S6+4S7+8S8)

Sum

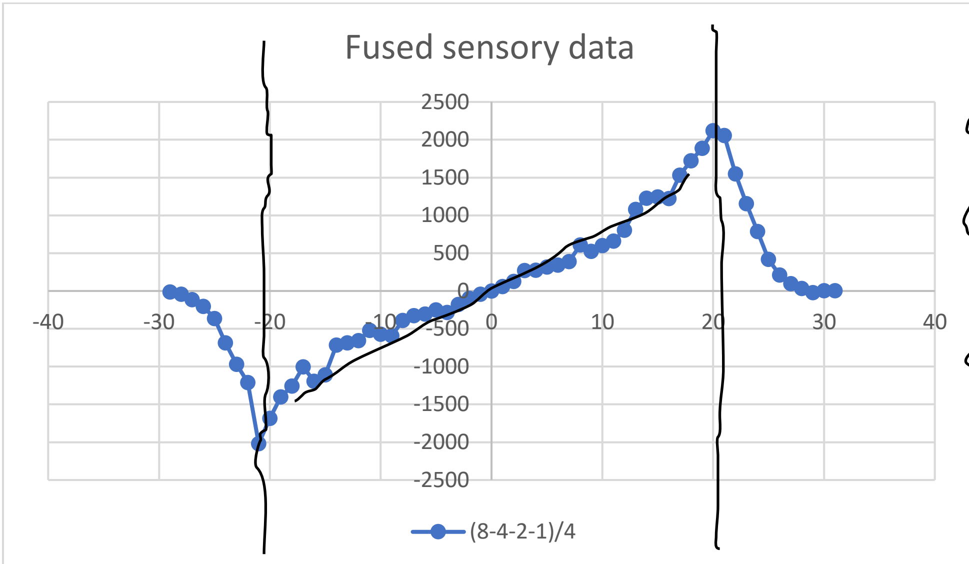


8 sensor readings

⇓

"Error"

Sensor fusion:



Calibration:

① 8 offsets

② 8 scale factor

Sensor Fusion:

8 weights to put on sensors

~~$k_i \cdot \int error\,dt$~~

constant

24 parameters

\+

2 parameters

PD controller:

$$Current\ Speed = Reference\ Speed - \underbrace{k_p \times error}_{\text{proportional control}} - \underbrace{k_d \times \frac{derror}{dt}}_{\text{derivative control}}$$

Constant

How to implement $\frac{derror}{dt}$?

$k_d \cdot \frac{derror}{dt} \Rightarrow \frac{\Delta error}{\Delta t} \cdot k_d$

Constant

$$k_d^* = \frac{k_d}{\Delta t_{loop}} \Rightarrow \frac{prev\_error - cur\_error}{\Delta t_{loop}} \cdot k_d$$

?

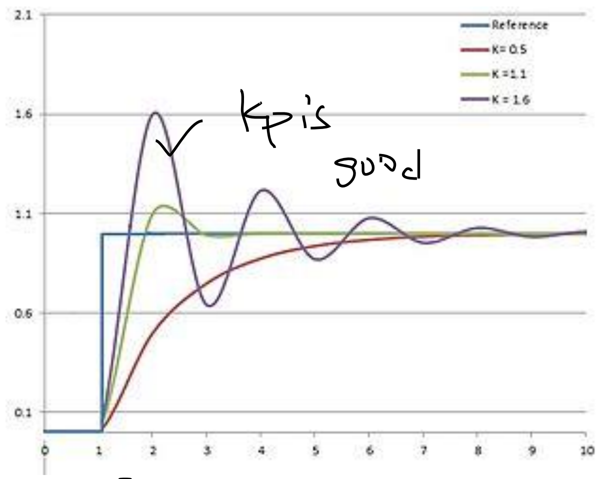
constant

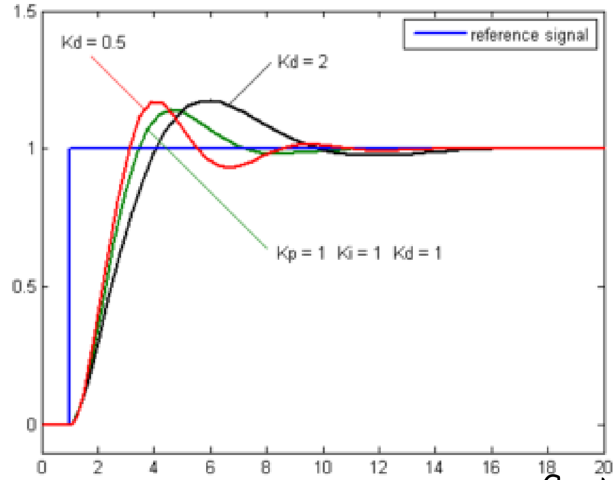
$$\Rightarrow \frac{avg(past\ n\ errors) - cur\_error}{\Delta t_{loop}} \cdot k_d$$

$$\Rightarrow k_d^* \cdot (prev\_error - cur\_error)$$

experiment

When to implement the derivative control?





Generic effect of kp with kd & ki fixed

Generic effect of kd with kp & ki fixed

① Only implement kp, until car wiggling around the track

② Implement kd s.t. smooth out the wiggles

Goal of the project:

car follow the colored tracks

⇓

middle of car $\xrightarrow{\text{align}}$ center of the track

⇓

"Error" value ⟸ eliminate real-time

↳ 24 parameters + kp kd 2 controlling param.

+ 2 base wheel speed = 28 parameters