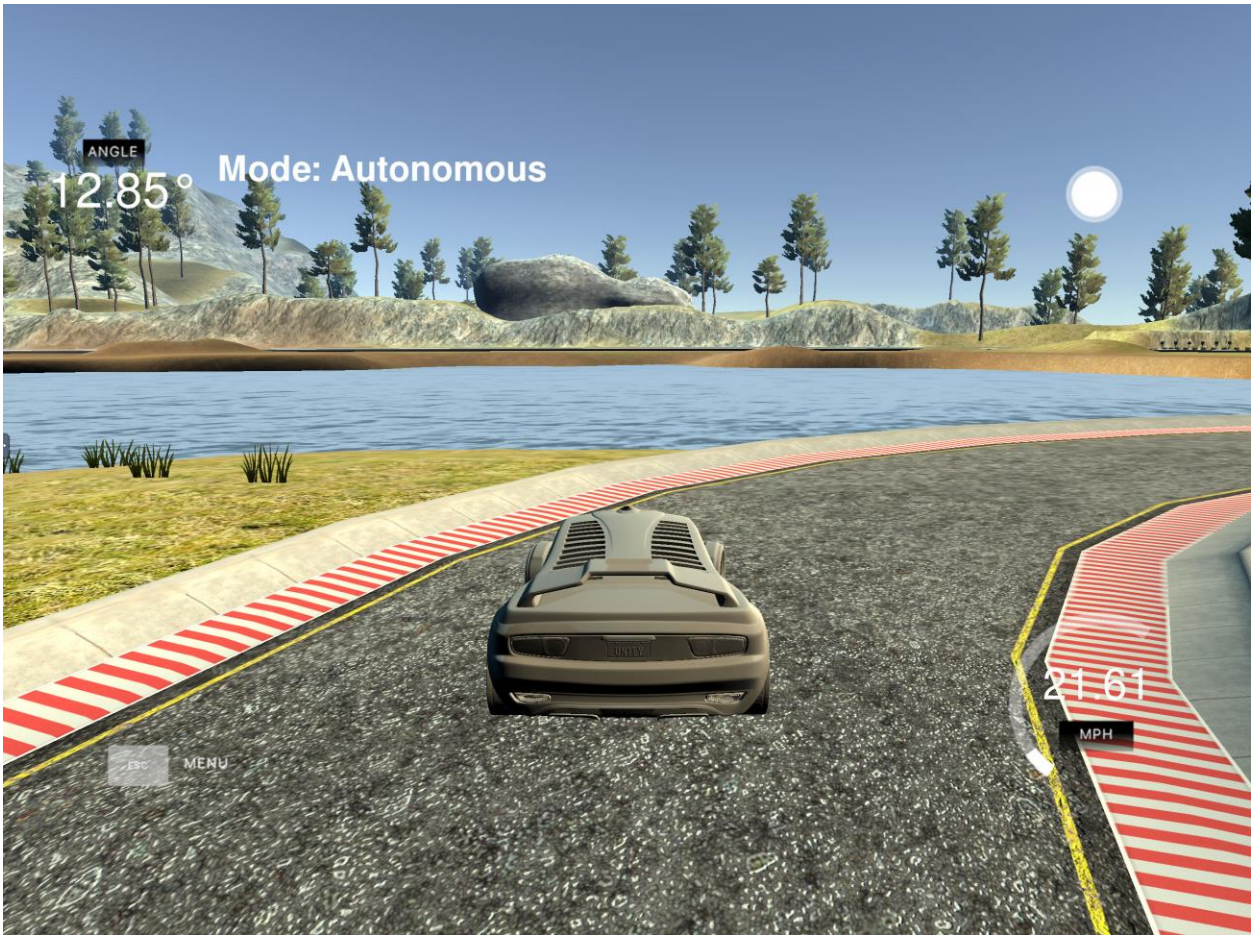
PID Controller Project



Report Prepared by Mwesigwa Musisi-Nkambwe for Udacity Self-Driving Car Nanodegree

Introduction

Control is how we move the throttle, brakes, and steering, hence controlling the car.

> **A PID controller is a feedback mechanism used to a control a system; in our case this will be our vehicle.**

Control Algorithms are normally called controllers, the controller I will use in this project is called a PID (proportional–integral–derivative) controller. A PID controller is a feedback mechanism used to a control a system; in our case this will be our vehicle. The controller is broken down into 3 components.

The 3 components (shown in the equation below) are Proportional, Integral, and Derivative.

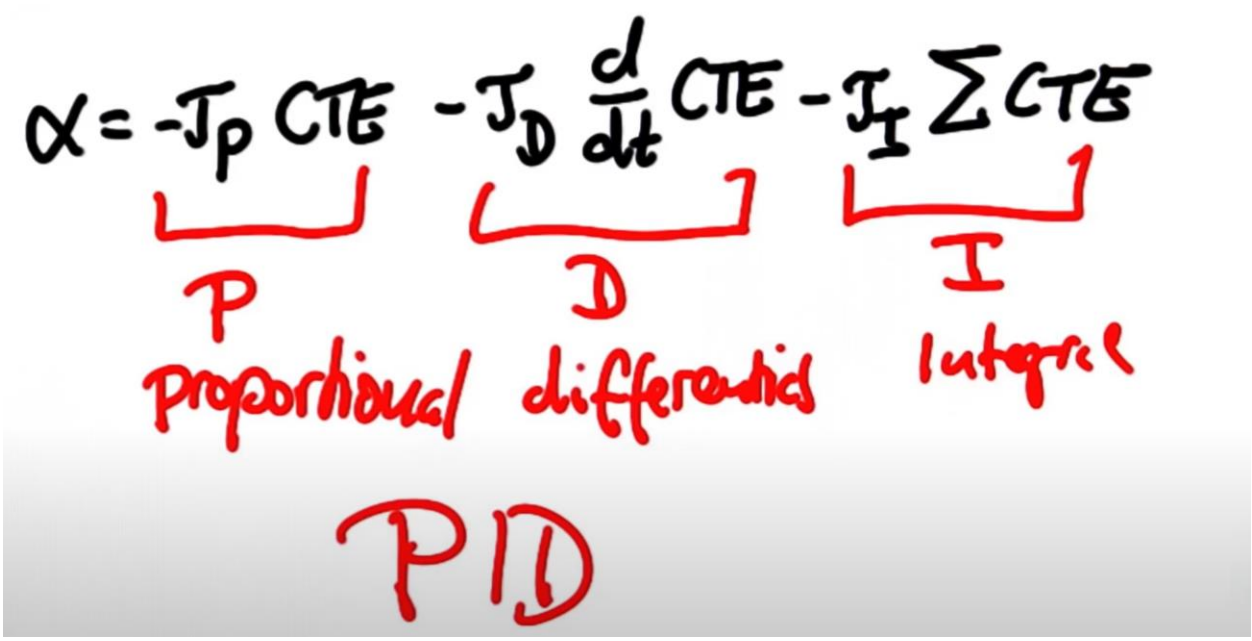


Figure 1 – Equation for PID Controller (Courtesy of Udacity)

In the above equation (in our situation) a cross-track error (CTE), is used use to measure the distance between our vehicle and the center lane (a reference) as shown in the image below. For PID control the actuating signal consists of proportional error signal minus the derivative and integral of the error signal. If the equation was only made up of the P(proportional), the car would continually overshoot/undershoot the center lane (oscillating). The D(differential) is used to counteract the oscillation and the I(integral) is used to bring the car close to the center lane when the car is parallel to the center lane but not close toi it. Using the above equations with τ we steer in proportion to the CTE that is supplied by the simulator. The summation of the total error is used as a correction for steering angle and throttle until it reaches the center of the lane and maximum acceleration, then it continuously corrects as the car moves.

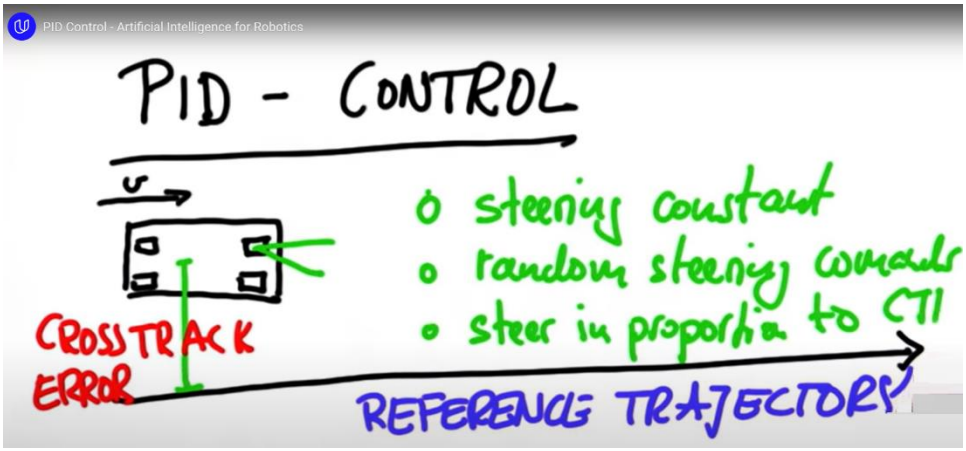


Figure 2 – Image depicting how Cross Track Error is calculated (Courtesy of Udacity).

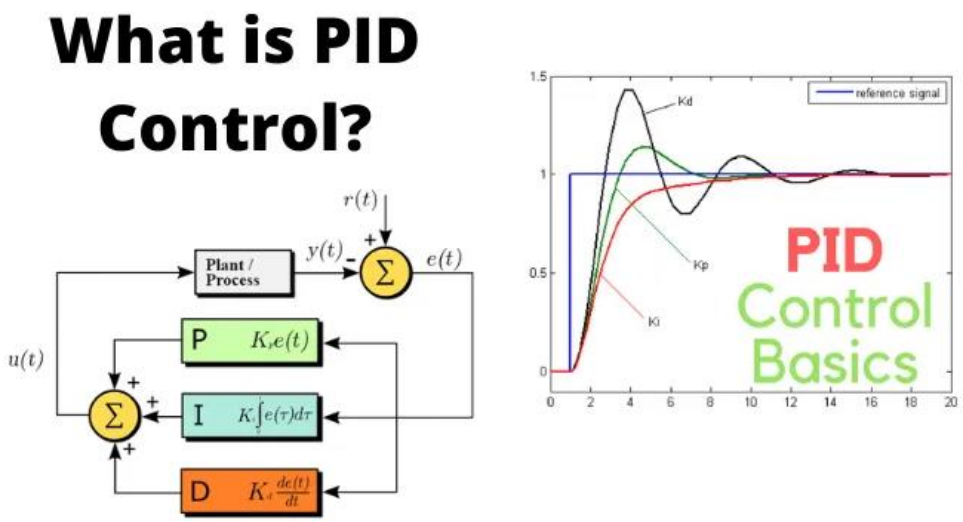


Figure 3 – Components that make up PID Control (Courtesy of www.electrical4u.com)

Twiddle

Twiddle is used to optimize a set of parameters; the effectiveness of the optimization is measured by goodness. In this project's case the goodness value is the average CTE, Twiddle is used to minimize the average CTE.

> In this project's case the goodness value is the average CTE, Twiddle is used to minimize the average CTE.

Twiddle incrementally increase or decrease the hyperparameters until the CTE is minimized. This decision-making process is shown in the code below (main.cpp, line 67-77):

```
pid.UpdateError(cte);
double total_error = pid.TotalError();
std::cout << "Total error " << total_error << std::endl;
if(total_error > 1.0)
{
  total_error = 1.0;
}
if(total_error < -1.0)
{
  total_error = -1.0;
}
```

Additional functions in PID.cpp help here; this is shown in the code below (PID.cpp in its entirety)

```
#include "PID.h"

/**
 * TODO: Complete the PID class. You may add any additional desired functions.
 */

PID::PID() {}

PID::~PID() {}

void PID::Init(double Kp_, double Ki_, double Kd_) {
  /**
   * TODO: Initialize PID coefficients (and errors, if needed)
   */

  p_error = 0.0;
  i_error = 0.0;
  d_error = 0.0;

  Kp = Kp_;
  Ki = Ki_;
  Kd = Kd_;


}

void PID::UpdateError(double cte) {
  /**
   * TODO: Update PID errors based on cte.
   */

  d_error = cte - p_error; // = cte - prev_cte

  p_error = cte;
  i_error += cte;

}

double PID::TotalError() {
  /**
   * TODO: Calculate and return the total error
   */
   return -1.0 * Kp * p_error - Ki * i_error - Kd * d_error;  // TODO: Add your total error calc here!
}
```