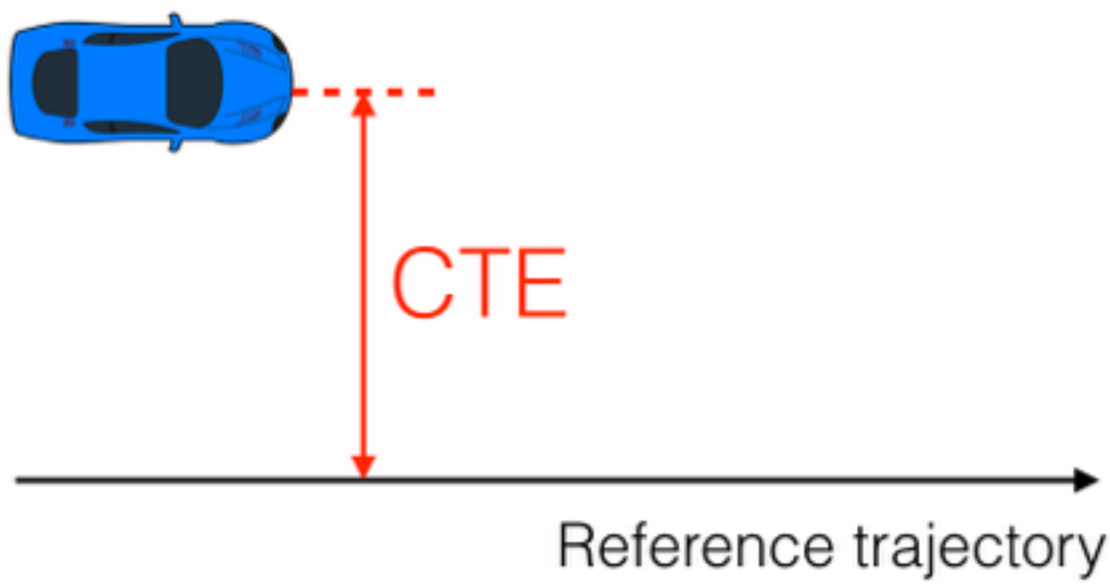
Describe the effect each of the P, I, D components had in your implementation:

PID (proportional, integral, derivative) controller is a control algorithm that allows to manage the behavior of a vehicle with respect to a desired path.
The adjustment of the trajectory of the the vehicle is done in proportion to a Cross Track Error (CTE) which is basically the offset of the vehicle's trajectory in respect with the target trajectory.

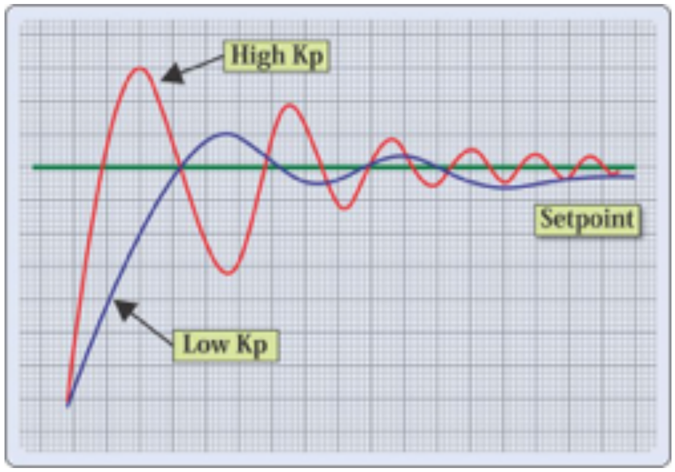


P-Controller: (P-term)

The proportional term produces an output value that is proportional to the error value. Therefore it will steer in proportion to the CTE, multiplying the error by a constant Kp, called the proportional gain constant.

alpha = - Kp*CTE

This forces the car to steer towards the reference trajectory but also never converges to it as, when alpha reaches 0, the car is still turned of a certain angle. We end up in a marginally stable condition with a lot of oscillations around the trajectory. As below, the magnitude of the oscillation is influenced by the value of Kp.



PD-Controller: (P-term, D-term)

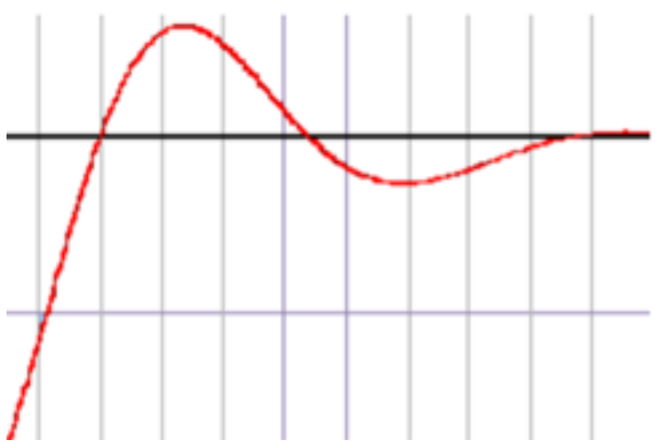
To avoid the escalation introduced just by the P-term we can use the derivative term D.
The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. So:

alpha = - Kd * d(CTE)/dt

where the derivative term d(CTE)/dt is:

d(CTE)/dt = [CTE(t) - CTE(t-1)] / delta_t   (where delta_t = 1)

So, as the error is decreasing over time the derivative term counter-steers and allows a smooth decline, getting the vehicle to converge smoothly towards the reference trajectory.



PID-Controller: (P-term, D-term, I-term)

The integral term of the PID controller takes care of adjusting a systematic bias that might be inherent in the vehicle and it's proportional to both the magnitude of the error and the duration of the error.
It is measured by the integral of the sum of CTE over time.

alpha = - Ki * Sum(CTE)

So, ultimately the complete formula for the PID controller will result as follows:

alpha = - Kp*CTE - Kd * d(CTE)/dt - Ki * Sum(CTE)

Describe how the final hyper-parameters were chosen:

Instead of implementing a twiddle algorithm for choosing the best parameters, I decided to manual tune them based on the behavior of the vehicle in the simulator.
I first set Kd and Ki to zero and just modified the Kp parameter.
The higher the value of Kp, the more the car was oscillating. I found that values around 0.1 were giving a good compromise in terms of the amplitude of the oscillations. I ended up using 0.05 as the oscillations were smaller and more manageable.
At this point the car was proceeding (oscillating) along the track following a trajectory that was positioned on the right side of the road. I therefore figured that there was a bias term in the system which I could offset changing the Ki value.
In this regards, in fact, I started using a value of 1.0 but it was too high and it was causing instability, so I kept decreasing it until the car was mainly following a trajectory at the center of the road. On the other hand, using a very small value, such as 0.001 was not sufficient to overcome the bias term. I ended up using 0.1 which was adjusting the offset nicely.
Finally, I modified the Kd value in order to reduce the oscillations and to allow the car to maintain a pretty stable trajectory.
I started with 1.0 but the oscillations were still considerable. On the other hand, higher values, such as 20.0 were giving an excessive response, more abrupt corrections that were overshooting the reference trajectory.
Ultimately I settled with 5.0 which, after a small disturbance in the beginning, it quickly reached the reference trajectory and maintained the car within the road lane lines.

Here the final values:

Kp = 0.05;   Ki = 0.1 ;   Kd = 5.0