Tuning of PID Track Followers for Autonomous Driving

Conference Paper · November 2018

DOI: 10.1109/3ICT.2018.8855773

CITATIONS
14

READS
244

2 authors, including:

Wael A. Farag
Cairo University
55 PUBLICATIONS **731** CITATIONS

SEE PROFILE

Tuning of PID Track Followers for Autonomous Driving

Wael Farag[1,2,4], Zakaria Saleh[3,5]

[1]American University of the Middle East (AUM), Kuwait.
[2]Electrical Eng., Cairo University, Egypt
[3]University of Bahrain, Bahrain.
[4]wael.farag@aum.edu.kw, [5]zsaleh@ubo.edu.bh

***Abstract*—In this paper, a Proportional-Integral-Differential (PID) controller that facilitates track maneuvering for self-driving cars is proposed. Three different design approaches are used to find and tune the controller hyper-parameters, one of them is specifically proposed in this paper for this specific application. The proposed controller uses only the Cross-Track-Error (CTE) as an input to the controller, whereas the output is the steering command. Extensive simulation studies in complex tracks with many sharp turns, have been carried out to evaluate the performance of the proposed controller at different speeds. The analysis shows that the proposed technique outperforms the other ones. The usefulness and the shortcomings of the proposed tuning mechanism are also discussed in details.**

Keywords—PID Control, Self-Driving Car, Autonomous Driving, PID Tuning.

I. INTRODUCTION

The biggest concern in the automotive market is perhaps the safety of the passengers, pedestrians and other objects on the road. Cars incorporate more and more intelligent systems [1-4] to improve safety, comfort and energy efficiency. Most of them, like ABS (Anti-lock Brake System) and ESP (Electronic Stability Program) are designed to maintain the control over the vehicle in extreme circumstances, being very effective at high speeds. However, for distances and speeds typical in urban traffic in which the vast majority of accidents happen, there is still a lack of safety systems. Fortunately, this trend is being reversed in recent times with the emergence of new intelligent systems, such as pedestrian detection, lane change alarm, speed limit alarm or emergency braking system.

All these systems are small steps towards a hypothetical future of fully autonomous and safe vehicles [5-6]. The vehicle control can be separated into lateral and longitudinal controls. Here the focus will be on the lateral control to follow a given trajectory with a minimized tracking error. In previous studies, various theories and methods have been investigated. These include the PID control methods [7-8], the predictive control method [9], the fuzzy control methods [10-11], the model reference adaptive method [12-13], the neural-network control method [14-15], the SVR (support vector regression) method [16], the fractional-order control method [17], etc. Recently much attention has been attracted by the use of the PID control method. PID control has such advantages as a simple structure, good control effect, robust and easy implementation [18]. However, this method needs supportive algorithms to find and fine-tune its hyper-parameters. It is challenging to find good optimized values for these hyper-parameters that well fit the environment caused by the complexity of vehicle dynamics, uncertainty of the external environments and the non-holonomic constraint of the vehicle.

Here in this paper, a further step towards the prospects of autonomous driving is presented. The main objective is to attain a car capable by itself of following a pre-defined trajectory or route generated by the prospected path planner of the autonomous car, in both highway and urban traffic with the highest comfort level.

The proposed control system uses sensors to provide the actual speed of the controlled car and the relative position of the car with respect to the supplied trajectory. The controller then calculates the optimized actions on the steering wheel. For controlling the direction by the steering wheel, a Proportional-Integral-Derivative (PID) controller is employed.

II. OVERVIEW OF PID CONTROL

The Proportional Integral Derivative controller [19] is a control mechanism widely used in several industrial applications. One of its many uses in robotics is to smoothen the robot's movement inputs in a way that it allows the robot to reach its objective state with as little noise as possible.

The output of a PID depends on three hyperparameters, one for each term of our controller (P-I-D). These hyperparameters are specific to each application, which in our case is driving a car smoothly around a test track.

The Proportional Term: The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain constant. The proportional term is given by:

$$P_{out} = K_p * Error \quad (1)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change. The chart in Fig. 1 shows how different values of K_p would impact the system response.

978-1-5386-9207-3/18/$31.00 ©2018 IEEE

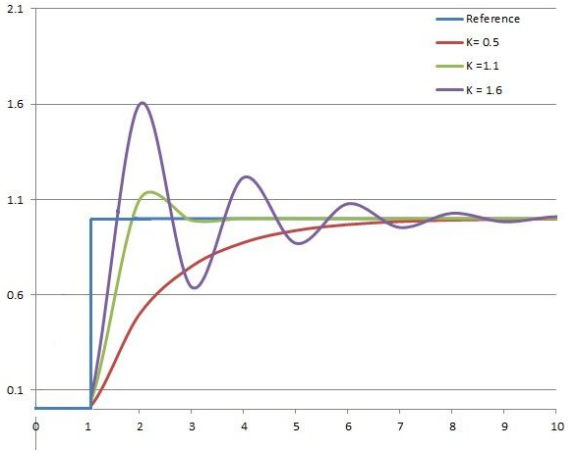

Fig. 1. The K_P effect in the system response.

The Integral Term: The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral term contribution is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (K_I) and added to the controller output.

$$I_{out} = K_I * \int Error\, dt \tag{2}$$

The chart in Fig. 2 shows how different values of K_I would impact the system response.

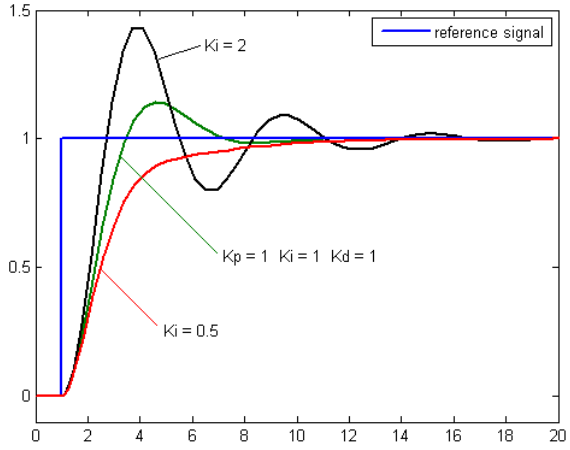

Fig. 2. The K_I effect in the system response.

The Differential Term: 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 K_D. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_D. The derivative term is given by:

$$D_{out} = K_D * \frac{d}{dt}(Error) \tag{3}$$

The chart in Fig. 3 shows how different values of K_D would impact the system response.

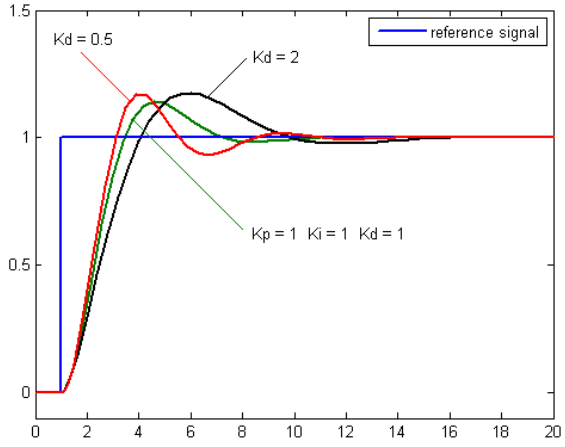

Fig. 3. The K_D effect in the system response.

The overall control action is shown in Fig. 4 can be given as

$$PID_{out} = P_{out} + I_{out} + D_{out} \tag{4}$$

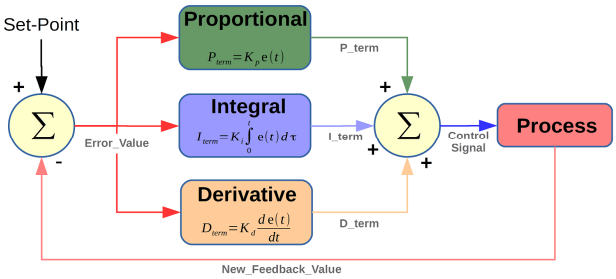

Fig. 4. Stucture of the PID controller.

III. The Vehicle Model

In this paper, the Kinematic Bicycle Model [5] is used to simulate the behavior of the self-driving car. The nonlinear continuous time equations that describe a kinematic bicycle model shown in Fig. 5 in an inertial frame are:

$$\dot{x} = v * \cos(\psi + \beta) \tag{5a}$$

$$\dot{y} = v * \sin(\psi + \beta) \tag{5b}$$

$$\dot{\psi} = \frac{v}{l_r} * \sin(\beta) \tag{5c}$$

$$\dot{v} = a \tag{5d}$$

$$\beta = tan^{-1}\left(\frac{l_r}{l_f + l_r} * tan(\delta_f)\right) \tag{5e}$$

where x and y are the coordinates of the center of mass in an inertial frame (X, Y). ψ is the inertial heading and v is the speed of the vehicle. l_f and l_r represent the distance from the center of the mass of the vehicle to the front and rear axles, respectively. β is the angle of the current velocity of the center of mass with respect to the longitudinal axis of the car. a is the acceleration of the center of mass in the same direction as the velocity. The control inputs are the front and rear steering angles δ_f, and a. Since in most vehicles the rear wheels cannot be steered, we assume $\delta_r = 0$.

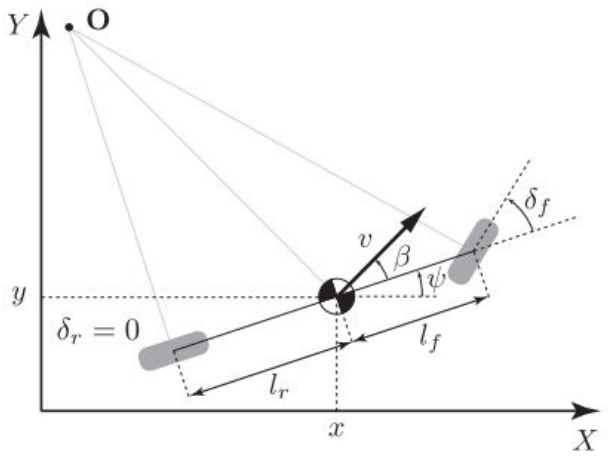

Fig. 5. The Kinematic Bicycle Model.

Compared to higher fidelity vehicle models [19], the system identification on the kinematic bicycle model is easier because there are only two parameters to identify, l_f and l_r. This makes it simpler to port the same controller or path planner to other vehicles with differently sized wheelbases.

IV. The Proposed PID Tuning Method

In order to design and implement the proposed Proportional-Integral-Differential (PID) controller for an autonomous vehicle to successfully maneuver around a complex track which has lots sharp turns, the following measurements are received by the controller:

1) The CTE (the cross track error) which represents the misalignment of the vehicle with respect to the center of the track at a given instance and can be given by:

$$CTE = desired\ position - vehicle\ position \quad (6)$$

2) The vehicle speed at the given instance.

3) The instantaneous vehicle orientation angle (-ve for left and +ve for right).

The PID controller then uses some of the above information to produce a steer (angle) command to the vehicle in addition to a throttle command (speed) if required.

The steer command is produced after applying proportional, integral and differential control to it in terms of K_p, K_I *and* K_d coefficients respectively. Therefore, the main design effort is to carefully tune these three coefficients to get the best possible performance. The performance can be simply defined as to let the vehicle to follow the track center line as closely as possible with the lowest aggregated CTE throughout the entire trip. Therefore, the main goal of the controller is to minimize the aggregated CTE (the objective function) as given by equation (7):

$$Objective\ Function = \text{mininmize}\{MSE\} = min\frac{1}{N}\sum_{i=0}^{i=N} CTE_i^2 \quad (7)$$

The PID coefficients are tuned using a proposed tuning method given the name "*WAF-Tune*". The following procedural steps illustrates this method:

1) The tuning will be done through the ad hoc technique "trial and error" which makes it a transparent procedure, unlike other automating algorithms like "Twiddle" [20] (will be described latter) that is considered a kind of opaque approach (it is not known exactly, how each change in a hyper parameter affects the performance). This manual transparent approach incorporates the intuition as well as the experience in the tuning process.

2) The time span at which the objective function is evaluated is selected to be considerably big (N=10,000). Ten thousand samples (individual CTE measurements) are used to evaluate the performance indicator (given in equation (4)). The number of samples are big enough to get an accurate enough evaluation. It represents the measurements from the vehicle while going around the designated track more than once (several times in case of the high enough speeds).

3) At first, the throttle value is kept constant throughout the performance measurement.

4) The tuning process starts by setting the throttle value at a low value (= 0.1 in our case) and the controller coefficients as follows (K_p = 0.1, K_I = 0.0 and K_D = 0.0).

5) The simulation for several iterations (at least two) in which each iteration produces an individual measurement of the objective function has been run. Then, the average of these measurements is taken to produce the designated performance indicator (average Mean Squared Error "MSE").

6) In the next set of iterations, the K_p has been then incremented small amounts based on intuition while keeping the other coefficients constant. Afterwards, the new performance indicator is then calculated. If the performance gets better, K_p is kept incremented, otherwise; return back to the previous value and move on to the next coefficient K_I by incrementing it a small amount (based on intuition as well).

7) After the performance stops improving with the increment of K_I, then K_D will be picked and getting step by step incremented till the performance indicator stops improving.

8) After tuning the three coefficients for the throttle value = 0.1, their reached values will be the starting point of the next throttle increment iteration.

9) The next throttle value iteration: the throttle value got incremented to say (=0.15), and steps from 5 → 8, got repeated while holding the throttle value constant at 0.15.

10) Keep repeating steps 5 → 9 while incrementing the throttle value (=> 0.2, 0.25, 0.3, 0.35, 0.4 ⋯ etc.).

11) The tuning stops when PID coefficients fails to control the vehicle (the vehicle jumps out of the track) at an upper throttle value and the performance indicator shows a large unacceptable value of MSE.

12) The final set of coefficients {K_p, K_I and K_D} that keep the vehicle within the track and an utmost reached value of throttle is considered the accepted tuned design of the PID.

13) These values are then used at all levels of throttle up to the utmost reached value.

V. The Twiddle Algorithm

The *Twiddle* algorithm [20] is a search algorithm that tries to find a kind of optimized selection of the hyper-parameters

values of the PID controller based on the returned track error. The pseudo code for the implementation of the *Twiddle* algorithm is as follows:

```
# Choose an initialization for the parameter vector ζ = [KP, KI, KD]
ζ = [0, 0, 0]
# Define potential changes
Δζ = [1, 1, 1]
# Define Drive_CTE(ζ) as the function that returns the ∑CTE for driving
   for the track with parameter vector ζ
# Calculate the error
best_err = Drive_CTE(ζ)

threshold = 0.001

while sum(Δζ) > threshold:
   for i in range(length(ζ)):
      ζi = ζi + Δζi
      err = Drive_CTE(ζ)

      if err < best_err:  # There was some improvement
         best_err = err
         Δζi = Δζi * 1.1
      else:  # There was no improvement
         ζi = ζi - 2*Δζi  # Go into the other direction
         err = Drive_CTE(ζ)

         if err < best_err:  # There was an improvement
            best_err = err
            Δζi = Δζi * 1.05
         else  # There was no improvement
            ζi = ζi + Δζi
            # As there was no improvement, the step size in either
            # direction, the step size might simply be too big.
            Δζi = Δζi * 0.95
```

VI. THE ZIEGLER–NICHOLS ALGORITHM

The Ziegler-Nichols tuning method [21] is a heuristic method of tuning a PID controller. This method requires to set K_D and K_I to 0 and gradually increase K_P until it reaches the ultimate gain K_u before the vehicle runs with stable and consistent oscillations as shown in Fig. 6. K_P and the oscillation period T_u are used to set the K_P, K_I, and K_D gains based on the type of controller used as shown in TABLE I.

TABLE I. FINE TUNING THE PID USING "ZIEGLER–NICHOLS".

Ziegler Nichols Method			
Control Type	K_P	$T_i = K_P / K_I$	$T_d = K_D / K_P$
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
Classic PID	$0.6K_u$	$T_u/2$	$T_u/8$

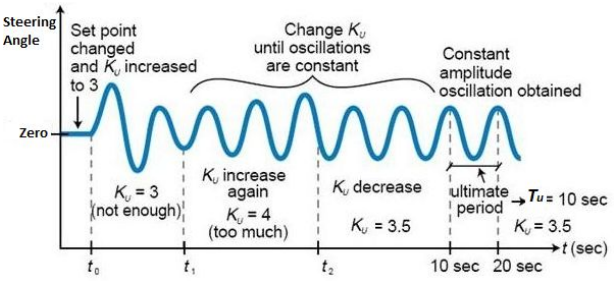

Fig. 6. Fine tuning the PID using "Ziegler–Nichols".

VII. TUNING AND TESTING RESULTS

The tuning of the PID controller is carried out using the upper three algorithms by driving the car around the test track shown in Fig. 7. TABLE II. shows samples of the endeavors of the "*WAF-Tune*" (*Section IV*) to reach fine-tuned values for the controller coefficients $\{K_P, K_I \text{ and } K_D\}$.

The final tuning of PID controller is given in trial #39 where (K_p = 0.35, K_I = 0.0005 and K_D = 6.5). Using these coefficients the controller is tested under different throttle values and the performance results are showing in TABLE III. below.

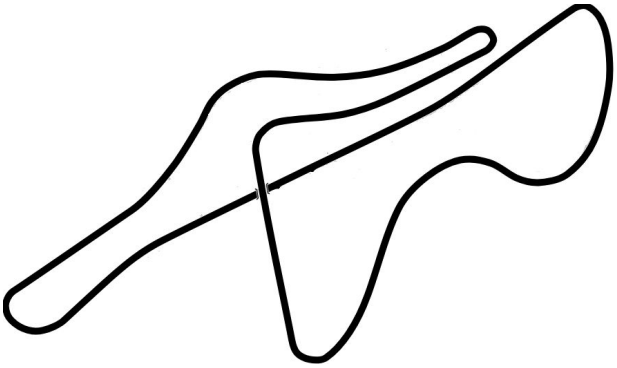

Fig. 7. The Test Track.

To tune the PID using *Twiddle (Section V)*, the hyper-parameters have to be tuned manually at first. This was necessary because the narrow track left little room for error, and when attempting to automate parameter optimization it was very common for the car to leave the track, thus invalidating the optimization. Once the initial preliminary parameters were found that were able to get the car around the track reliably, *Twiddle* is then implemented. It was necessary to complete a full lap with each change in parameter because it was the only way to get a decent "score" (total error) for the parameter set. For this reason, the changes in each parameter are allowed to "settle in" for 100 steps and are then evaluated for the next 2000 steps. In all, *Twiddle* is allowed to continuously run for over 1 million steps (or roughly 500 trips around the track) to fine tune the parameters to their final values $\{K_P$: 0.134611, K_I: 0.000270736, K_D: 3.05349$\}$ with an MSE of 0.1823.

Furthermore, to tune the PID using *Ziegler-Nichols* method, the PID is initialized with hyper parameters values that just allowed the car to drive around the track even with a lot of wobble. Then K_p is getting increased till the car makes full approximately full swings between the boarders (lane lines) of the track at which the value of K_p is captured and set to given the name K_u, while the period of the oscillations at

this point is called T_u. The experimentations with different values of K_u & T_u is shown in TABLE IV. below to arrive at the final optimal values of {K_P: 0.09, K_I: 0.00144, K_D: 1.40625}. Unfortunately, the PID controller with resulted parameters was able to drive car around the track but with a still lot of wobbling. That is why, parameters should further tuned manually by a trial-and-error process. The same process is applied for different speeds (throttle values), so different PID parameters were found for different speeds. However, the overall performance is still inferior to that of the "*WAF-Tune*" and the "*Twiddle"* algorithms.

TABLE II. FINE TUNING THE PID CONTROLLER USING "WAF-TUNE".

Trial #	Throttle	K_p	K_I	K_D	Average MSE	Comments
1	0.1	0.18	0.00005	0.36	0.246889	
2	0.1	0.18	0.00005	0.5	0.243526	
3	0.15	0.18	0.00005	0.5	0.280253	MSE jumps higher due to the increment in throttle value.
4	0.15	0.18	0.00005	0.6	0.270583	
5	0.15	0.18	0.00005	0.7	0.277959	
6	0.15	0.18	0.00005	0.9	0.246081	
7	0.2	0.2	0.00005	1.0	0.293409	
8	0.2	0.2	0.0001	1.0	0.253787	
9	0.25	0.2	0.0001	1.0	0.531351	MSE jumps higher due to the increment in throttle value.
10	0.25	0.2	0.0000	1.0	0.509472	
11	0.25	0.2	0.0000	1.25	0.335846	
12	0.25	0.2	0.0000	1.5	0.300634	
13	0.25	0.2	0.0000	1.75	0.280235	
14	0.25	0.2	0.0000	2.0	0.269807	
15	0.25	0.3	0.0000	2.0	0.209112	
19	0.25	0.4	0.0000	2.0	0.489602	An increment in K_P that caused big loss in performance.
20	0.25	0.4	0.0000	2.5	0.314796	
21	0.25	0.4	0.0000	3.0	0.227320	
22	0.25	0.4	0.0000	3.5	0.192307	
23	0.25	0.4	0.0000	4.0	0.159206	
25	0.25	0.4	0.0000	4.5	0.146784	
26	0.25	0.4	0.0000	5.0	0.129811	
27	0.25	0.4	0.0000	5.5	0.119762	
28	0.25	0.4	0.0000	6.0	0.123671	Major improvements of performance after several increments of K_D
29	0.3	0.4	0.0000	6.0	0.161787	MSE jumps higher due to the increment in throttle value.
30	0.3	0.4	0.0000	6.5	0.166268	
31	0.3	0.45	0.0000	6.5	0.165845	
32	0.3	0.35	0.0000	6.5	0.147639	
33	0.3	0.30	0.0000	6.5	0.156575	
34	0.3	0.4	0.0000	6.0	0.150308	
35	0.3	0.4	0.0000	7.0	0.154149	
36	0.3	0.4	0.0000	6.0	0.166513	
37	0.3	0.35	0.0001	6.5	0.116793	Adding a bit of K_I at this stage helped improve the performance significantly.
38	0.3	0.35	0.0002	6.5	0.115610	
39	0.3	0.35	0.0005	6.5	0.116543	This set of coefficients are considered the final tuning.

TABLE III. TESTING RESULTS FOR THE PID CONTROLLER @ DIFFERENT SPEEDS.

Throttle	K_p	K_I	K_D	Average MSE	Comments
0.1	0.35	0.0005	6.5	0.062864	~ 12 miles/hour
0.15	0.35	0.0005	6.5	0.072709	~ 17 miles/hour
0.2	0.35	0.0005	6.5	0.080815	~ 23 miles/hour
0.25	0.35	0.0005	6.5	0.098465	~ 28 miles/hour
0.3	0.35	0.0005	6.5	0.1165430	~ 34 miles/hour
0.35	0.35	0.0005	6.5	0.148176	~ 39 miles/hour
0.4	0.35	0.0005	6.5	0.160981	~ 44 miles/hour
0.45	0.35	0.0005	6.5	0.207031	~ 49 miles/hour
0.5	0.35	0.0005	6.5	0.331020	~ 55 miles/hour

TABLE IV. EXPERIMENTATION WITH K_U & T_U TO FIND OPTIMIZED K_P, K_I & K_D.

K_u	T_u	Comments
2.0	100	Wheels turns very fast, very fast response
1.0	100	Complete a lap but response still very fast
0.5	100	Better than previous but still fast response
0.5	75	Better than previous but still fast response
0.25	75	Better than previous but still can be improved

0.25	60	Bigger oscillations but completes a lap
0.15	75	Completes a lap but needs to reduce oscillations
0.15	80	Much better but 0.25/0.75 still the best so far
0.25	80	Can still reduce oscillations
0.25	80	Increased speed (60 mph) with same previous settings. High oscillations but still goes through a complete lap.
0.25	90	Several Laps without crashing but still needs to reduce oscillations.
0.25	100	Several Laps without crashing but still needs to reduce oscillations.
0.20	100	Better than previous, but can damp oscillation further.
0.19	110	Better than previous, but can damp oscillation further.
0.17	110	Better than previous, but can damp oscillation further.
0.15	110	Better than previous, but can damp oscillation further.
0.14	110	Better than previous, but can damp oscillation further.
0.14	115	Still need to try and get inside lines with turn 3
0.13	115	Goes out of lane
0.14	120	Completes laps. Try and get to drive inside lane on turn 3
0.14	125	Completes laps. Try and get to drive inside lane on turn 3
0.145	125	Completes laps without a problem but can damp oscillations further
0.15	125	Final solution

VIII. DISCUSSION

The following are some conclusive remarks on the proposed controller and the work done:

1) The PID controller has a very simple structure however, it very effective in many control problems. Therefore, it is the most widely used approach by far in industry (>90%).
2) The main problem with the PID, is its tuning. There is no theory or criteria that proves that you have reached to the optimal value for the coefficients $\{K_p, K_I \text{ and } K_D\}$. All the methods of tuning are mainly based on extensive search with the incorporation of intuition and experience.
3) From our point of view, using transparent methods based on extensive "trial and error" endeavors guided by a numerical performance indicators is the best approach; as it lets you understand the problem at hand much deeper. Furthermore, it allows you to incorporate your intuition and experience which reduces a lot of the search space; and consequently allows you at the end to be more effective.
4) The problem @ hand is a tracking problem in which the set-point (the track center position) keeps changing continuously with time. In such kind of problems, the differential controller (K_D) proved to be very effective and it is designed in principal to track changes. In our case, it played the dominant role. The differential component counteracts the tendency for oscillation or overshoot around the track center line. By properly tuning K_D, it will cause the vehicle to approach the center line smoothly with much lower oscillations.
5) The proportional controller (K_p) is necessary to feedback the error to the controller with corrective action but it is not playing the principal role in improving the performance in this problem. The proportional component had the most directly observable effect on the vehicle's behavior. It causes the vehicle to steer back to the track trying to reduce the CTE.
6) However, the integral controller (K_I) shows to be ineffective is our case (due to the continuous change of the setting point) and in many scenarios it has a detrimental effect. The integral component counteracts a bias in the CTE and speeds up the approach to the center line, however, in our case, the center line keep moving with respect to the vehicle which in many cases may cause the integral controller to produce oscillations.

IX. SUGGESTED IMPROVEMENTS

The following list summaries the suggested improvements:

1) Adding another PID controller to control the throttle in such a way to reduce the speed in sharp curves and to allow full thrust in straight portions of the track. This controller should take the vehicle angle as an input and produces the "throttle value".
2) Use of Adaptive PID control approaches to accommodate the wide range of vehicle speed. One of the popular approaches is the Gain Scheduling (GS) adaptive control, where we use several gain sets (3 or 4 sets), each will be used @ a specific speed range.
3) Applying "*Twiddle*" algorithm to the results of the proposed "*WAF-Tune*" to further optimize the hyper-parameters. However, caution should be taken in in order not to overfit the results to specific test tracks.

X. CONCLUSION

In this paper, a PID controller is designed to efficiently steer an autonomous car following a pre-calculated track from a path planner. Three different methods are presented as design approaches, one of them is newly proposed in this paper. The proposed controller uses on the cross-track-error as an input and outputs the steering commends. The method used to tune and minimize the suggested objective function is described in details. The performance of testing the PID controllers at different vehicle speeds (throttle setting) is also shown in details. A comprehensive discussion and analysis regarding the design of the PID as well as suggestions for improvements and future work is presented.

REFERENCES

[1] Karim Mansour, Wael Farag, "AiroDiag: A Sophisticated Tool that Diagnoses and Updates Vehicles Software Over Air", in the *2012 IEEE Intern. Electric Vehicle Conference (IEVC)*, TD Convention Center Greenville, SC, USA, March 4, 2012, ISBN: 978-1-4673-1562-3.

[2] Wael Farag, Zakaria Saleh, "Traffic Signs Identification by Deep Learning for Autonomous Driving", *Smart Cities Conferemce* (SCS'18), Bahrain, 22-23 April, 2018.

[3] Wael Farag, "Recognition of traffic signs by convolutional neural nets for self-driving vehicles", *International Journal of Knowledge-based and Intelligent Engineering Systems*, IOS Press, Vol: 22, No: 3, pp. 205 – 214, 2018.

[4] Wael Farag, "CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms", *7th Inter. Conf. on Modeling, Simulation, and Applied Optimization (ICMSAO)*, UAE, March 2017.

[5] L. Alonso, J. Pérez-Oria, B. M. Al-Hadithi, A. Jiménez, "Self-tuning PID controller for autonomous car tracking in urban traffic," in *17th Inter. Conf. on Sys. Theory, Control and Computing (ICSTCC)*, Sinaia, Romania, 11-13 Oct. 2013 .

[6] P. Zhao, J. Chen, Y. Song, X. Tao, T. Xu and T. Mei, "Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID," *International Journal of Advanced Robotic Systems,* vol. 9, no. 44, 2012.

[7] T. Le-Anh, M.B. De Koster, "A review of design and control of automated guided vehicle systems", *Erasmus Research Institute of Management (ERIM)*, report series no. 2004 - 03 - LIS, 2004.

[8] F. A. A. Cheein, C. Cruz, T. F. Bastos and R. Carelli, "SLAM - based Cross-a-Door Solution Approach for a Robotic Wheelchair". *International Journal of Advanced Robotic Systems*, Vol 7, No. 2, pp. 155 - 164, 2010.

[9] R. Lenain, B. Thuilot, C. Cariou and P. Martinet, "Model Predictive Control for vehicle guidance in presence of sliding: application to farm vehicles' path tracking", *IEEE Conf. on robotics and automation*, pp. 885 – 890, 2005.

[10] T. Hessburg, "Fuzzy logic control for lateral vehicle guidance", *IEEE Control System Magazine*, Vol 14, pp. 55-63, 1994.

[11] R. Choomuang, N. Afzulpurkar, "Hybrid Kalman Filter/Fuzzy Logic based Position Control of Autonomous Mobile Robot", *Intern. Journal of Advanced Robotic Systems*, Vol 2, No 3, pp. 197 –208, 2005.

[12] W. Wang, N. Kenzo, and O. Yuta, "Model Reference Sliding Mode Control of Small Helicopter X.R.B based on Vision", *Intern. Journal of Advanced Robotic Systems*, Vol 5, No 3, pp 233 – 242, 2006.

[13] R.H. Byrne, "Design of a Model Reference Adaptive Controller for Vehicle Road Following", *Mathematical and Computer Modelling*, Vol 22, pp 343–354, 1995.

[14] Li Z., W. Chen, H. Liu, "Robust Control of Wheeled Mobile Manipulators Using Hybrid Joints", *International Journal of Advanced Robotic Systems*, Vol 5, No 1, pp 83-90, 2008.

[15] B. Shumeet, "Evolution of an artificial neural network based autonomous land vehicle controller", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol 26(3), pp 450–463, 1996.

[16] G. Lacey, Z. Ji, "Computing the solution path for the regularized support vector regression", in *Advances in Neural Information Processing Systems* 18 (NIPS 2005).

[17] D. Zhuang, "The Vehicle Directional Control Based on Fractional Order PDμ Controller", *Journal of Shanghai Jiaotong*, Vol 41(2), pp 278-283, 2007.

[18] K.K. Tan, W. Qing-Guo; H. C. Chieh, "Advances in PID Control", London, UK: Springer-Verlag, 1999, ISBN 1-85233-138-0.

[19] J. Kong, M. Pfeiffer, G. Schildbach and F. Borrelli, "Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design", *IEEE Intelligent Vehicles Symposium (IV)*, Seoul, South Korea, 28 June 2015.

[20] S. Thrun, "CS373: Artificial Intelligence For Robotics," Udacity, San Francisco, California, 2018.

[21] J. ZIEGLER1, N. B. NICHOLS and N. Y. ROCHESTER, "Optimum Settings for Automatic Controllers," *TRANSACTIONS OF THE A.S.M.E.,* pp. 759-765, November, 1942.