Steering Control of Self-Driving Car Using Super-Twisting Sliding Mode Control

Mohammad Dika, Michel Owayjan, Roger Achkar
Departments of Mechatronics Enginering and Computer and Communication Engineering
American University of Science and Technology
Ashrafiyeh, Lebanon
mdika@aust.edu.lb , mowayjan@aust.edu.lb , rachkar@aust.edu.lb

***Abstract*—Super-Twisting is an algorithm implemented on the typical Sliding Mode Control (SMC) that offers a different method of controlling complex systems, and commonly found in advanced robotic systems such as industrial robotic manipulators and mobile robotics. This paper presents a review of the steering control of a self-driving car using Super-Twisting algorithm. The algorithm offers optimal performance and speed when compared to other common controls such as the PID control due to its chattering resistant nature and speed. The algorithm is tested using a real-world servo motor applied on a real car body on the 2021 NXP Racetrack ©, using a set amount of time expected to complete the track at a certain speed. The Super-Twisting algorithm is applied on the motor with the required steering angles for the racetrack with respect to time as the reference input, and it is compared to an implemented PID Control. The results show the superior performance of the Super-Twisting SMC in this application compared to the PID control, offering the best option of control resulting in better line following and hence better overall time on the circuit.**

Keywords—Super-Twisting SMC, Ackermann Steering, PID Control, Curvature, Steering Angle

I. Introduction

The automotive industry has been developing self-driving vehicles since the 1920's [1], with the first truly autonomous car developed in the 1980's by Carnegie Mellon University's Navlab [2] being the first car to do so, and one of the most important aspects of a self-driving vehicle is its ability to remain on its track, and to follow a specific line and trajectory of motion. To remain on the trajectory, the main aspects of the vehicle would be its speed and its steering angle. There have been multiple ways to control the steering of a car relative to its trajectory and speed, whether it is an open loop or closed loop system. Control systems have been implemented to control the steering using servo motors, whether in real cars or autonomous-car racing tournaments, with the servo motor being controlled via PID control, Sliding Mode Control (SMC) and many more.

In the coming sections, the concept of PID and Super-Twisting SMC control will be introduced. The kinematic model of the vehicle according to the Ackermann steering will be explained with details regarding the motion and mathematical equations of the car. Then the mathematical model of the servo motor for steering as well as the implementation of PID and SMC on the motor. The methods will be explained with comparisons of both implementations. The methodology and results are explained.

II. Literature Review

A. Ackermann Steering

Ackermann steering is the default steering of vehicles nowadays. Developed by German carriage builder Georg Lankensperger in 1817 and patented by his agent Rudolph Ackermann in 1818. The Ackerman Mechanism, as shown in figure 1, connects the front wheels of the vehicle to one axis, with the wheels off center, allowing the two wheels' perpendicular axis to meet at one center of rotation [3]. The Ackermann steering design can be found in all vehicles nowadays and in autonomous-car racing competitions such as the NXP Cup cars. The Ackermann design offers as well a mathematical model for path planning and following due to the knowledge of the center of rotation, as well as relating all the parameters needed such as (x,y) position to the speed of the vehicle and the steering angle of the car [4,5].

B. Line Following

Line following mobile robots operate by following a certain line drawn for its trajectory, whether on the ground or on the ceiling. This application of mobile robots is one of the first implementations of the Automated Guided Vehicles, which starting in the 1940's to 1950's. Nowadays this technology works via computer vision, where using a camera sensor, the robot can localize its position relative to a certain line it needs to follow and adjust its movement accordingly [6].

An adjustment to line following is what is called lane tracking, where rather than a robot following a central line, the mobile robot rather detects two lines on its sides and aims to stay between these two lines, hence sticking to a certain lane [7]. Lane tracking is becoming more and more common in cars with their advancement in self-driving, popularized by the Tesla cars.

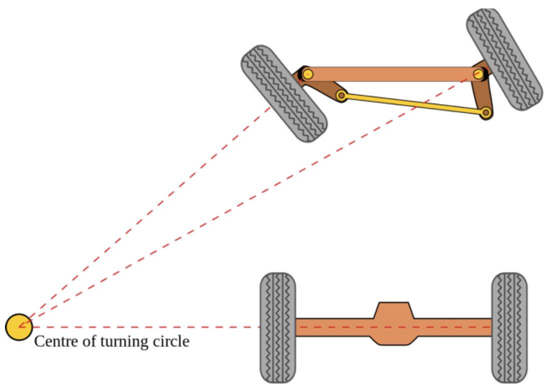


Figure 1: Ackermann Steering

C. Steering Control

With the application of Ackermann steering in autonomous car competitions, and the emergence of electric vehicles, the steering of cars has become electric, with the design of a servo motor acting as the actuator of the front wheels' axis. This has resulted in studies in how to optimize the servo control to offer optimal balance between comfort and speed, with the most common applications being using PID control [8,9].

D. Sliding Mode Control

Sliding Mode is a nonlinear control method. It alters the dynamics of a nonlinear system by applying a discontinuous signal that forces the system to "slide" along a cross-section of the system's normal behavior. The sliding mode control is also known as a variable structure control, where it can switch from one continuous structure to another based on the current position in the state space.

SMC features remarkable properties of accuracy, robustness and easy tuning as shown in multiple studies [1,2,10,11,12]. SMC is split into two phases, sliding surface design and control input design.

The aim of the sliding surface design is to track a desired output, y_{des} by minimizing the error $e=y-y_{des}$. Often, the sliding surface depends on the tracking error e_y with a certain number of its derivatives.

$$\sigma = \sigma(e, \dot{e}, \ldots, e^k) \qquad (1)$$

The function σ should be selected in such a way that its vanishing, $\sigma = 0$, gives rise to a stable differential equation any solution $e_y(t)$ will tend to zero eventually.

The successive phase (PHASE 2) is finding a control action that steers the system trajectories onto the sliding manifold, that is, in other words, the control is able to steer the σ variable to zero in finite time. There are several methods, the focus here will be the super-twisting algorithm which is able to minimize the trial-and-error aspect of the controller design to one variable [13].

III. Methodology of The Work

As mentioned in section II, SMC is split into two phases, sliding surface design and control input design. The Control input will be based on Super-Twisting Algorithm, which offers a continuous output as opposed to the typical SMC signal. The servo motor controlling the steering of the car will be the plant of the system, with the QUBE-Servo 2 being the servo motor of choice for the mathematical model [14,15], the car is then modeled according to the DFRobot GPX ROB0157 chassis. The car dimensions are taking into consideration when operating the kinematic model of the car. Then using the NXP racetrack of 2021, reference steering angles were designed according to a set speed and expected time duration for the whole track, and the reference steering angles were inputted to two systems, PID control and SMC control, of which they were compared and analyzed for the optimal control system in this design.

A. QUBE-Servo 2 Motor

The Servo motor chosen in this study is the QUBE-Servo2 from Quaser, due to its functionality of having high precision as well as a direct relation between voltage applied and the position, as shown in (2).

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{K}{s(\tau s + 1)} \qquad (2)$$

Where K is the model steady-state gain, and τ the model time constant. Θ_m representing the disk position, and finally V_m the applied motor voltage. Figure 2 exhibits the Quebo-Servo 2 Motor's equivalent electric circuit. With table 1 showing the specifications of the servo motor.

From the motor datasheet [15]. We can find K=22.4 and τ = 0.15 s.

Another possible approach to the servo modeling is having the armature of a DC motor separately powered by a separate power source, hence the design of the mathematical model would be that of a normal DC motor, but the voltage coming from a difference power source as seen in (3).

$$P(s) = \frac{\theta(s)}{V(s)} = \frac{K}{s((Js + b)(Ls + R) + K^2)} \qquad (3)$$

Where J, b, L and R are the moment of inertia of the rotor, damping ratio of the mechanical system, internal electric inductance, and internal electric resistance of the motor respectively. However, due to the impracticality of having the armature separately powered, as well as internal differences between a normal speed DC motor and a DC Servo motor, (2) was used for the mathematical model of the servo. Replacing K and τ with their respective values, we reach (4).

Table 1: QUBE-Servo 2 system parameters

Symbol	Description	Value
DC Motor		
R_m	Terminal resistance	8.4 Ω
k_t	Torque Constant	0.042 N.m/A
k_m	Motor back-emf constant	0.042 V/(rad/s)
J_m	Rotor inertia	4.0 x 10^{-6} kg.m^2
L_m	Rotor inductance	1.16 mH
m_h	Load hub mass	0.0106 kg
r_h	Load hub radius	0.0111 m
J_h	Load hub inertia	0.6 x 10^{-6} kg.m^2
Load Disk		
m_d	Mass of disk load	0.053 kg
r_d	Radius of disk load	0.0248 m

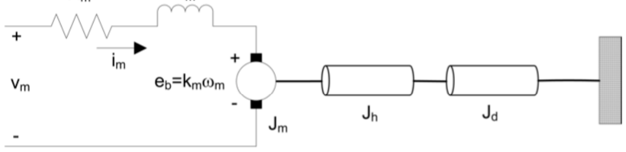


Figure 2: QUBE-Servo 2 Equivalent Circuit

$$P(s) = \frac{\theta_m(s)}{V_m(s)} = \frac{22.4}{s(0.15s+1)} \tag{4}$$

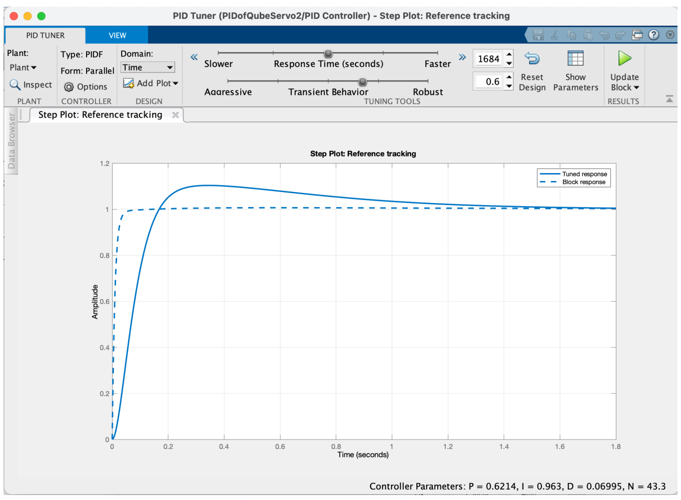
All values and equations of the QUBE-Servo 2 can eb found in [14,15].

B. PID Control of Servo Motor

Two methods of control were applied to the servo motor, PID Control and SMC Control. PID Control works by continuously measuring the error value e(t). the difference between the desired output and the actual output, then applies corrections based on proportional, integral, and derivative terms. PID is controlled by the three gains K_p, K_I and K_D representing proportional gain, integral gain and derivative gain. Typically to determine these values, we start by zeroing all three gains and start by incrementing K_p until we reached desired speed, then start to increase K_D to illuminate the high oscillations and finally increment K_I to reach the better steady state error. In our implementation, the model was implemented in SIMULINK ©, where the three gain values were determined using the PID Autotuner found in SIMULINK ©, as shown in figure 3. With the optimal values for the gains, required the best balance for accuracy and speed can be found in table 2. These values will be used in the coming chapters when testing and comparing with Super-Twisting SMC. Figure 4 showcases the whole block diagram with the servo motor plant.

Table 2: PID Gain Values

Symbol	Description	Value
K_p	Proportional Gain	3.98977233045272
K_I	Integral Gain	3.04294722446996
K_D	Derivative Gain	0.69283743850694


PID Tuner (PIDofQubeServo2/PID Controller) - Step Plot: Reference tracking

Plant: Plant; Type: PIDF; Form: Parallel; Domain: Time

Response Time (seconds): 1684

Transient Behavior: 0.6

Controller Parameters: P = 0.6214, I = 0.963, D = 0.06995, N = 43.3

Figure 3: PID Tuning

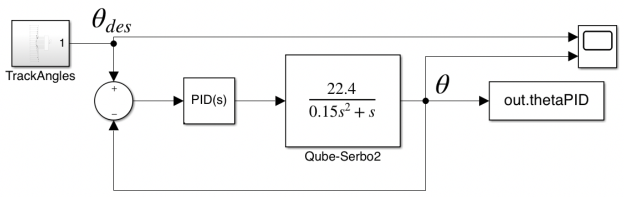


Figure 4: PID Control Block Diagram

C. Super-Twisting SMC

In section II the SMC was introduced with the introduction of σ. However, a more accurate representation of σ depends on the order of the system, denoted by k, as seen in (5).

$$\sigma = \left(\frac{d}{dt} + p\right)^k e \tag{5}$$

Where k=r-1, with being the relative degree between the input and the output, from (4) we can see that our difference r=2, hence k=1, so (5) now becomes (6). The term *p* is chosen arbitrarily, using trial and error *p*=48.

$$\sigma = \dot{e} + p.e \tag{6}$$

To control σ, the super twisting algorithm is implemented. Super-twisting is best implemented when the order of the system is of the second order or above, in our case it is indeed 2nd Order. Super-Twisting is chosen in favor of a normal SMC implementation due to the discontinuous nature of SMC which generates chattering, which is undesirable. Using Super-Twisting allows stability due to its continuous nature, hence we can control the chattering problem.

To control σ, we introduce correction terms to control the voltage V_m of the servo motor to control the position.

$$V_m = V_1 + V_2 \tag{7}$$

$$V_1 = -\sqrt{F^*|\sigma|}sgn(\sigma) \tag{8}$$

$$V_2 = -1.1F^*sgn(\sigma) \tag{9}$$

Where V_1 and V_2 are the correcting terms, with *K* selected arbitrarily. In our testing F^*=50. Equations (8) and (9) can be replaced in (7) to achieve the total equation of the input to the servo motor, found in (10).

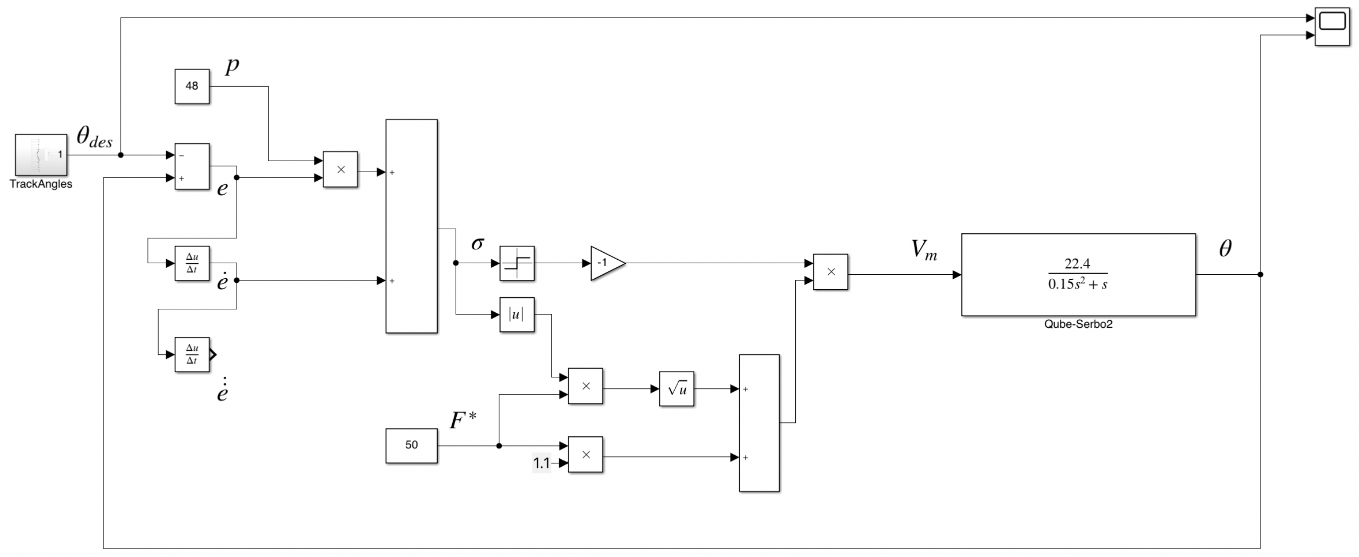


Figure 5: Super-Twisting SMC Block Diagram

$$V_m = -sgn(\sigma)(1.1F^* + \sqrt{F^*|\sigma|}) \tag{10}$$

In figure 5 can be found the block diagram for the Super-Twisting SMC design.

D. Reference Track

To test the designed control systems, a reference track was designed to measure the control design performance. The reference track is a replica of the track used by NXP © for the NXP CUP © 2021.

The track, found in figure 6, offers optimal testing due to its design of multiple multi-directional curves, zigzags, and straight lines.

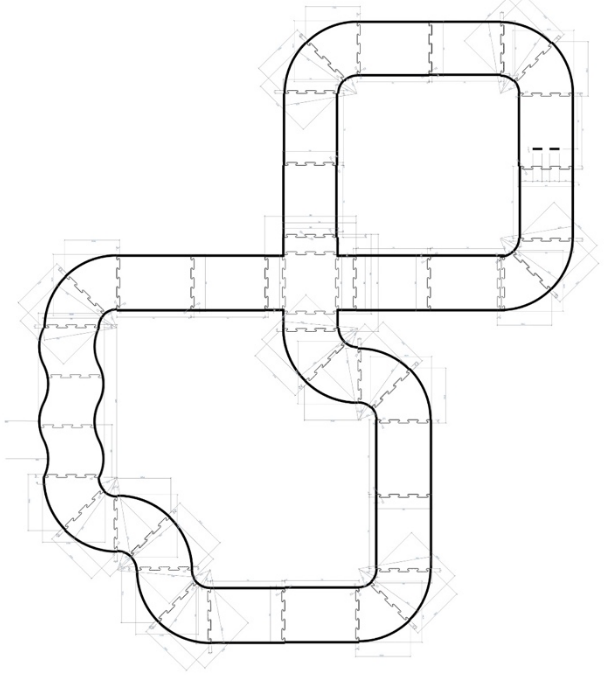


Figure 6: Reference Track

The track was used to find the reference angles θ_{des} for our systems to match. The track's starting point is at the two black dashes on the top right, moving counterclockwise. The track was used in the NXP CUP © 2021 competition, with the winning team completing the track in around 10 seconds, hence that will be our time constraint for the track, finishing it all in 10 seconds. This time duration T will assist us with finding the required steering angles depending on time for our reference systems to handle.

To determine the required steering angles the speed of the car is also needed. To find the speed we need to find the total length of the track. Since the self-driving car does not know the track previously, it is assumed that it will follow the middle of the track. All the turns found in the track have the same turning radius and arc length. The curves can be seen in a close-up in figure 7.

The inner radius of the curve being 170 mm and outer curve 720 mm, we can find the midpoint of the track being 445 mm due to the off-center of the inner radius.

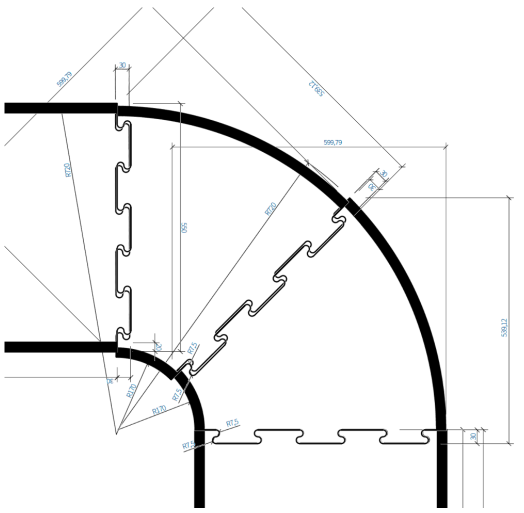


Figure 7: Curvature of the Track

With all curves having the same turning radius, and all straight lines having known lengths of 1440 mm, we can find the total length of the track to be *L*=20.44 m. Hence the speed needed, assuming the car is moving at a constant speed throughout the track. $V = \frac{x}{t} = \frac{20.44}{10} = 2.044\, m/s.$

E. Ackermann Kinematics

Now that the speed, curvature, and length of the track are known, we need to find the steering angles required. The car model is of the Ackermann steering design, found in figure 8. Where ϕ is the steering angle, θ is the orientation of the center of gravity and x and y coordinate of the robot in the frame. Equations (11), (12) and (13) representing the relationships between the variables.

$$x = u_1 \cos(\theta) \tag{11}$$

$$y = u_1 \sin(\theta) \tag{12}$$

$$\dot{\theta} = \frac{u_1}{l} \tan(\phi) \tag{13}$$

Where u_1 is the forward velocity of the car and *l* is the distance between the front and back axis of the wheels. In the ROB0157 car model, *l*=17.5 cm.

To find the necessary steering angles for the track and curves, we resort to figure 9.

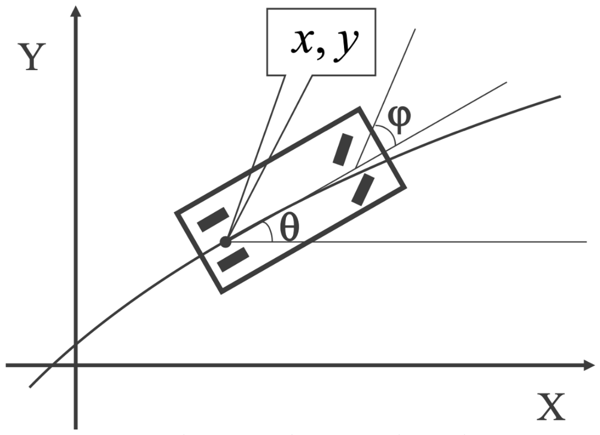


Figure 8: Ackermann Kinematics

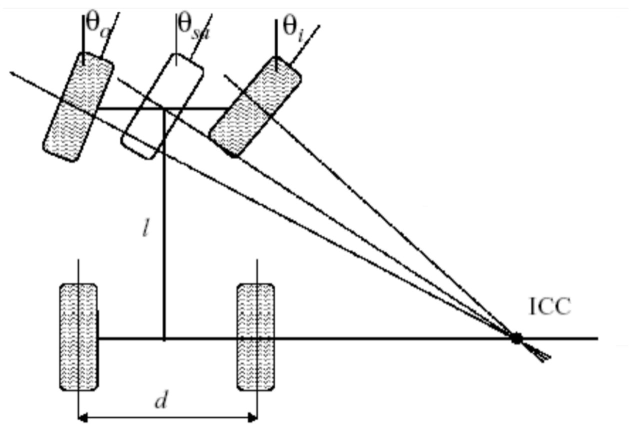


Figure 9: Ackermann Curve Steering

Since the two front wheels are connected to the same axis and hence the same servo motor, they can be modeled and replaced by a singular wheel in the middle of the axis with a steering angle θ_{SA}. This is the same steering angle as the QUBE-SERVO 2 motor. Variable *d* is the distance between the two fixed back wheels of the car.

Using trigonometric rules, we can find θ_{SA} using (14).

$$\theta_{sa} = \cot^{-1}\left(\frac{R}{l}\right) \tag{14}$$

Where R is the rotation radius, found from to be R=44.5 cm and *l*=17.5 cm, we can find the necessary steering angle at the curvature of the track to be:

$$\theta_{sa} = \cot^{-1}\left(\frac{44.5}{17.5}\right) = 21.5\,° \tag{15}$$

Now that θ_{SA}, the steering angle is known for all turns of the circuit, and since the speed is known to be V=2 m/s, we can find the reference steering angles θ_{ref} to cover the whole track.

Since we know the lengths of the straight lines and the distances between curves, we can measure how long θ needs to be zero in a straight line, and since we know the length of the arcs of the curves, we can know how long $\theta=\theta_{SA}$ on the curves to match the rotations.

Plotting the steering angles with respect to time, we reach the figure 10, showcasing θ_{ref}, the reference dataset to be entered to our control systems.

Note that θ is negative when the robot is rotating clockwise, and positive when rotating counterclockwise. This dataset will be the input to our control systems.

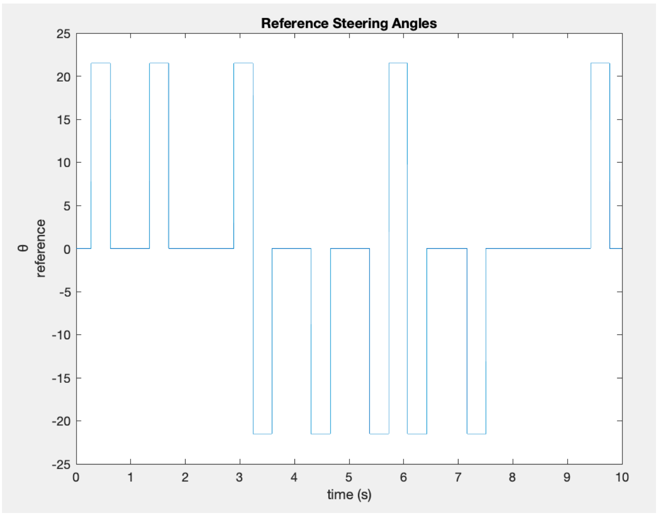


Figure 10: Desired Steering Angles Vs. Time

IV. Results And Conclusion

A. Results

Taking the dataset found in figure 10 and inserting it to the PID Control system found in figure 4, yields the following output θ_{real}, found in figure 11. Comparing the PID results against the desired steering angles in figure 12, we can see that the PID control matches it well.

However, closer inspection of the results shows in figure 13 that the PID output does not match the desired angles.

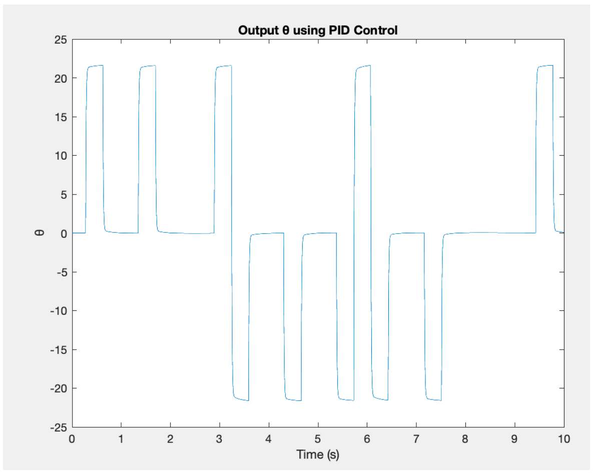


Figure 11: PID Control Results

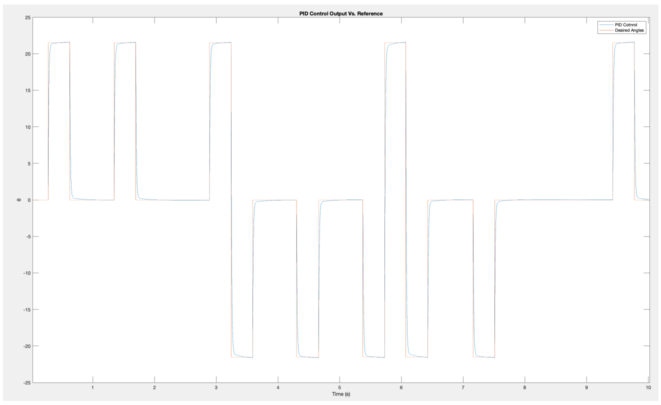


Figure 12: PID Output Vs. Reference

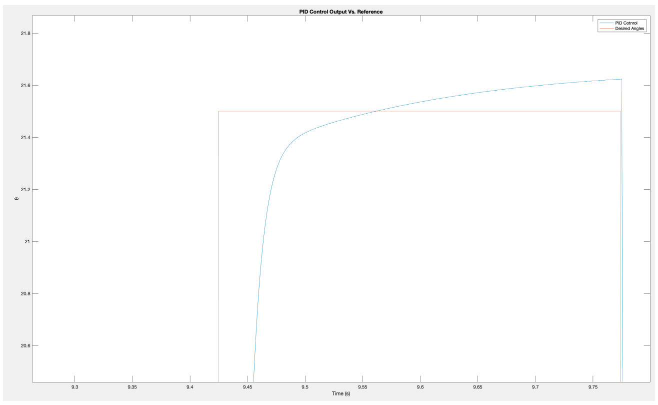


Figure 13: PID Output Vs. Reference Closeup

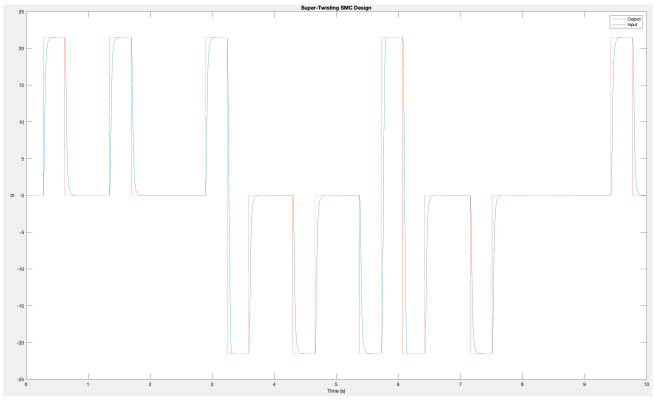


Figure 14: SMC Output Vs. Reference

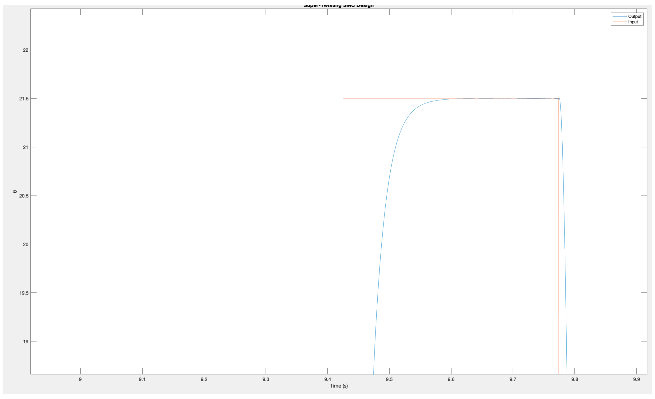


Figure 15: SMC Output Vs. Reference Closeup

Although the difference may seem unsubstantial, but this overshoot will cause the self-driving car to oscillate around the line it is attempting to follow, resulting in unstable movement and poor performance especially in competitions. Adding the same reference angles to the Super-Twisting SMC design yields figure 14 and a close-up in 15.

We can analyze from figure 15 that not only does the SMC design match the speed of the PID Control, but it also offers a much more stable response, resulting in better line tracking without any chattering or oscillations, and hence better overall performance for the self-driving vehicle.

B. Conclusion and Future Work

In this paper, the Super-Twisting Sliding Mode Controller design was implemented on the steering of a self-driving car and compared to the performance of the same application using PID Control. The Super-Twisting SMC could match the speed of the PID Control while eliminating the overshooting and any possible case of chattering that may arise from the normal SMC application, hence offering optimal performance when it comes to line following and especially competitive self-driving car competitions. Such gains from the SMC design result in huge advantages in races and offers smoother drives and application on real world vehicles. The Super-Twisting algorithm can be implemented not just on self-driving vehicles, but to be implemented on any trajectory following from spaceships, airplanes, ships, and any other trajectory following machine.

References

[1] The Milwaukee Sentinel, "'Phantom Auto' will tour city". 8 December 1926. Retrieved 23 July 2013.

[2] "Carnegie Mellon". *Navlab: The Carnegie Mellon University Navigation Laboratory. The Robotics Institute*. Retrieved 2014-12-20

[3] J.Zhao, X. Liu, Z. Feng and J. Dai, "Design of an Ackermann-type steering mechanism," in Jounral of Mechanical Engineering Science, 2013.

[4] T. Rizano, D. Fontanelli, L. Palopoli, L. Pallottino and P. Salaris. "Global Path Planning for Competitive Robotic Cars," in *52nd IEEE Conference on Decision and Control*, 2013.

[5] T. Rizano, D. Fontanelli, L. Palopoli, L. Pallottino and P. Salaris. "Local Motion Planning for Competitive Robotic Cars," in *52nd IEEE Conference on Decision and Control*, 2013.

[6] A. Kondár, Z. Törcsvsári, Á. Nagy and I. Vajk, " A Line Tracking Algorithm Based on Image Processing," in *IEEE 12th International Symposium on Applied Computational Intelligence and Informatics*, May 17, 2019.

[7] P. Amaradi, N. Sriramoju, L. Dang, G. Tewolde and J. Kwon, " Lane Following and Obstacle Detection Techniques in Autonomous Driving Vehicles," IEEE 976-1-4673-9985-2, 2016.

[8] X. Wang, W. Wang, L. Li, J. Shi, B. Xie, "Adaptive Control of DC Motor Servo System with Application to Vehicle Active Steering," in *IEEE/ASME Transactions on Mechatronics*, 2018.

[9] P. Kumar, V. Babu.(2014, November) " Position Control of Servo Systems using PID Controller Tuning with Soft Computing Optimization Techniques," in *IJERT*, Vol. 3 Issue 11.

[10] Modern Sliding Mode Control Theory. New Perspectives and Applications. G. Bartolini, L. Fridman, A. Pisano, E. Usai (Eds.), Springer Lecture Notes in Control and Information Sciences, Vol. 375.

[11] V.I. Utkin, "Variable Structure systems with Sliding Modes." *IEEE Transaction on Automatic Control*, 22, 2, 212-222, 1977.

[12] H. Maghfiroh, C. Apribowo. (2020, April). "Basic Tutorial on Sliding Mode Control in Speed Control of DC-motor." *JEEICT*, Vol. 2, No. 1. Available: https://www.researchgate.net/publication/341127740_Basic_Tutorial_on_Sliding_Mode_Control_in_Speed_Control_of_DC-motor

[13] A. Chalanga, S. Kamal, L. Fridman, B. Bandyopadhyay, J. Moreno. "Implementation of Super-Twisting Control: Super-Twisting and Higher Order Sliding Mode Observer Based Approaches," in *IEEE Transactions on Industrial Electronics*, 2016.

[14] Quanser, "Direct Discrete Control Design," Servo 2 Datasheet, 2020

[15] Quanser, "Qube-Servo 2," Servo 2 Datasheet, 2020