International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.5 (Sep-2020)

http://dx.doi.org/10.12785/ijcds/090511

Complex Track Maneuvering using Real-Time MPC Control for Autonomous Driving

Wael Farag [1, 2]

[1]*College of Eng. & Tech., American University of the Middle East, Kuwait.*
[2]*Electrical Eng. Dept., Cairo University, Egypt.*

Received 8 Oct. 2019, Revised 31 Jan. 2020, Accepted 25 Aug. 2020, Published 1 Sep. 2020

Abstract: This work proposes a framework to design, formulate and implement a path tracker for self-driving cars (SDC) based on nonlinear model-predictive-control (NMPC) approach. The presented methodology is developed to be used by designers in the industrial sector, practitioners and academics. Therefore, it is straight forward, flexible, and comprehensive as well. It allows the designer to easily integrate multiple objective terms in the cost function either opposing or correlating. The proposed design of the controller not only targets accurate tracking but also comfortable ride and fast travel time as well by introducing several sub-objective terms in the main cost function to satisfy these goals. These sub-objective terms are weighted according to their contribution to the optimization problem. The SDC-NMPC framework is developed using the high-performance language C++ and utilizes highly optimized math and optimization libraries for best real-time performance. This makes the SDC-MPC well suited for use in both ADAS and self-driving cars. Extensive simulation studies featuring complex tracks with many sharp turns have been carried out to evaluate the performance of the proposed SDC-NMPC at different speeds. The presented analysis shows that the proposed controller with its tuning technique outperforms that of the PID-based controller.

Keywords: MPC Control, Self-Driving Car, Autonomous Driving, Path Planning, Tracking.

1. INTRODUCTION

Increasing safety, reducing road accidents, improving energy efficiency and enhancing comfort and driving experience are the major motivations behind equipping modern cars with Advanced Driving Assistance Systems (ADAS) [1-2]. These motivations represent incremental steps toward a hypothetical future of safe fully autonomous vehicles [3-8].

Recent self-driving cars (SDCs) architectures include a dedicated module "the motion planner" that produces the path/trajectory the car has to follow to reach its designated destination [9]. The motion planner module generates desired trajectories for the next 5-10 seconds (in the future) and updates these trajectories at a 10 Hz rate [9]. The generated trajectory comes in the form of a series of waypoints measured on global-map coordinates. SDCs possess as well a Drive-by-Wire (DBW) system "the path tracker" that strives to follow the generated trajectory as precise and as efficient as possible [9]. "Precise" in terms of minimizing the aggregated Cross-Track Errors (CTEs) between the generated trajectories and the actual SDC driving trajectories. In addition, "Efficient" in terms of improving ride quality, minimizing travel time and reducing fuel consumption.

The motivation of this paper is to facilitate for the SDC path tracker to achieve the previous kind-of-opposing objectives by proposing a sophisticated control methodology that takes into account all these objectives and provide a way to find a delicate balance among them based on the designer preferences. The proposed methodology is based on the Model Predictive Control (MPC) technique due to its flexibility and practicality [10]. The MPC technique is an optimal control method that has the advantage of tailoring its optimization strategy in a way that offers multiple options to reach the desired performance for strongly nonlinear systems with constraints [10], which are difficult to handle using traditional linear control approaches [11].

E-mail: wael.farag@aum.edu.kw, wael.farag@cu.edu.eg

Numerous thought-provoking MPC design methodologies exist in the literature, in [12] Ikeda et al proposed a new design of optimal control that takes only specified discrete values and applies finite-horizon sum-of-absolute-values optimization. The work has been also extended to include infinite-horizon model predictive control with the presence of bounded noise and applied to altitude control of an aircraft showing how the MPC takes only discrete output values, which may not fit our problem as both SDC outputs (steering and speed) are continuous states.

Moreover, Silva et al [13] proposed an iterative MPC that uses an iterative procedure to adjust a time-variant linearization of the nonlinear model of the system at each sampling time, which may allow controlling systems where a direct nonlinear MPC approach is not feasible. However, in the work considered in this paper, the used SDC model is time-invariant and the MPC optimization is solved using a primal-dual interior-point method [14] applied directly to the nonlinear model of the SDC.

Furthermore, Vatankhah et al [15] proposed an adaptive nonlinear MPC approach based on a neural network model [16] that generates offset-free output even under external disturbances and parameters' uncertainties. The adaptive structure for the model was provided by on-line training of the neural network. However, this approach might be considered not practical for the SDC as it required an off-line training of the neural network with data collected from the controlled system while running under PID control. Additionally, online training for a controller is not desired in safety-critical systems like SDCs.

Trajectory tracking is the process of designing a controller [10] that guides a vehicle (either ground [18], aerial [12] or marine [21]) and minimizes (or maximizes) some measure of performance while satisfying a set of constraints.

The novelties of this paper can be enumerated as follows:

1) Proposing a framework for constructing a Nonlinear MPC path tracker (SDC-NMPC) that provides a way of easily integrating several driving objectives through compounding a multi-term cost function. The proposed framework is meant to be easy to comprehend and straight forward to employ by the designers. Moreover, it allows multiple state variables to be combined together in one cost term if they are correlated or to be inserted separately in individual cost terms if their effect is required to be decoupled from other state variables.
2) Emphasizing the development and implementation stages and highlighting the real-time performance in detail to bridge the gap between theory and practice, and to address the industrial audience as well. The SDC-NMPC has been developed using C++ [25] with advanced math libraries to optimize real-time performance. Thus, it is taking into account the throughput (execution time) of affordable CPUs to handle the limited computational resources of embedded automotive hardware, and to prove that it is well suited for use in Advanced Driving Assistance Systems (ADAS) or self-driving cars.
3) Most of the cited work for the design of path trackers using MPC approach use only two terms in the cost function [10], which are the aggregated cross-track errors and the vehicle-orientation errors. However, in this work, several sub-objective terms are included to improve ride quality, fuel efficiency, and safety. Examples of such terms are speed-regulation term, steering-smoothening term, acceleration-management term, speed-steering term, and control-effort minimization term.
4) The MPC's nonlinear convex optimization problem is solved iteratively using the efficient primal-dual interior-point technique [14] with the direct incorporation of the vehicle nonlinear model (without linearization) aiming for highest accuracy [10]. Therefore, both IPOPT optimization [26] and the CppAD differentiation [27] efficient C++ libraries are employed seeking the best execution time.

2. NMPC PROBLEM FORMULATION

The general discrete-time nonlinear system in (1) is considered as the vehicle model

$$x_{k+1} = f(x_f, u_k),\ x_k \in \mathbb{X},\ u_k \in \mathbb{U}, \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ are the vehicle state and control action at time step *k*. Accordingly, a general MPC formulation can be expressed as follows

$$\begin{aligned} &\min \sum_{i=0}^{N-1} l(\hat{x}_i, \hat{u}_i) \\ &s.t. \quad \hat{x}_{i+1} = \bar{f}(\hat{x}_i, \hat{u}_i),\ i = 0, \dots, N-1 \\ &\hat{x}_0 = x_k, \\ &\hat{x}_i \in \mathbb{X}, \hat{u}_i \in \mathbb{U} \end{aligned} \tag{2}$$

where *N* is the horizon length, x_k is the initial condition which is the vehicle state at time step k, $\hat{x}\ and\ \hat{u}$ are the predicted state and control action respectively, $\bar{f}(.,.)$ is the nonlinear model equation representing the vehicle, $l(.,.)$ is the objective function. For tracking NMPC, $l(.,.)$ has usually a quadratic form ($|\hat{x}_i - x_s|^2$) that minimizes the difference between the predicted state and the set-point x_s.

Equation (2) is solved using the Primal-Dual Interior-Point (PDIP) method [15] as a constrained nonlinear optimization problem. To illustrate how the PDIP works, simply consider the all-inequality version of a nonlinear optimization problem

$$\min f(x)\ s.t.\ c_i(x) \geq 0\ for\ i = 1, \dots, m \tag{3}$$

$$x \in \mathbb{R}^n$$

$$where\ f: \mathbb{R}^n \to \mathbb{R}, c_i: \mathbb{R}^n \to \mathbb{R}$$

The logarithmic barrier function [28] associated with (3) is

$$B(x,\mu) = f(x) - \mu \sum_{i=1}^{m} log\big(c_i(x)\big) \quad (4)$$

Here μ is a small positive scalar, called the "barrier parameter". As μ converges to zero, the minimum of $B(x,\mu)$ should converge to a solution of (3). Consequently, the barrier function gradient is required and can be calculated as

$$g_b = g - \mu \sum_{i=1}^{m} \frac{1}{c_i(x)} \nabla c_i(x) \quad (5)$$

where g is the gradient of the main function $f(x)$, and $\nabla c_i(x)$ is the gradient of $c_i(x)$.

3. The Vehicle Model

In this paper, the kinematic bicycle model [29] is used to emulate the behavior of the self-driving car. The nonlinear continuous-time equations that describe the kinematic bicycle model shown in **Error! Reference source not found.** in an inertial frame are:

$$\begin{aligned} \dot{x} &= v * \cos(\psi + \beta) \\ \dot{y} &= v * \sin(\psi + \beta) \\ \dot{\psi} &= \frac{v}{l_r} * \sin(\beta) \\ \dot{v} &= a \\ \beta &= tan^{-1}\left(\frac{l_r}{l_f + l_r} * tan(\delta_f)\right) \\ \dot{\delta_f} &= \omega \end{aligned} \quad (6)$$

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 δ_r. Since in most vehicles the rear wheels cannot be steered, it is assumed that $\delta_r = 0$. ω is the steering angular velocity.

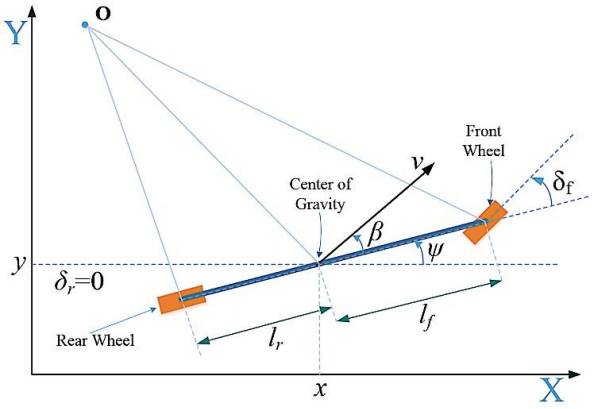

Figure 1. The Kinematic Bicycle Model.

Compared to higher fidelity vehicle models [29], 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.

The MPC employs the vehicle's motion model to plan an optimized and realistic trajectory given a set of constraints. These constraints could be the limits of the vehicle's motion, and a combination of costs that define how the vehicle should move (such as staying close to the best fit and the desired heading or preserving it from the excessive jerk of the steering wheel).

4. The Proposed SDC-NMPC Track Follower

To design and implement the *SDC-NMPC* track follower, several measurements need to be collected periodically from the *SDC* sensors. The following is the list of these measurements:

1) The "p_x" and "p_y" (the vehicle's current *x* and *y* positions) measured in the global "map" coordinates. These values are received from the "*SDC* Localization Module" which used the fusion between Global Position Systems (GPS) [30], Inertial Measurement Units (IMU) [30], LiDAR, and Radar sensors [31] to produce accurate car positioning coordinates on the global map.
2) The *SDC* velocity "V_{SDC}" at the given instance measured in miles/hour (mph) and received from the car speedometer [32].
3) The *SDC* orientation angle (heading) "ψ" (-ve for left and +ve for right) in radians received from the IMU, Radar or the fusion between them [33].
4) The *SDC* orientation angle "Ψ-unity" in radians commonly used in navigation and simulations, it is calculated directly from "ψ" [34].
5) The current steering angle of the *SDC* "δ_f" measured in radians using the mounted vehicle steering angle sensor.

6) The current throttle value mapped to the range [-1, 1] where (-ve for braking and +ve for speeding).

7) The "PTs_x" and "PTs_y" arrays that include the waypoints (reference track) measured in global coordinates supplied by the path planner module of the *SDC* [35].

The *SDC-NMPC* tracker then uses some of the above information to produce a steer (angle) command as well as the throttle command (speed) to the *SDC*. The following are the steps used for the implementation of this controller:

1) The received waypoints ("PTs_x" and "PTs_y") are converted from global coordinates to vehicle coordinates using the following transformation equations:

$$VPTs_{x_i} = \left(PTs_{x_i} - p_x\right)\cos\psi - \left(PTs_{y_i} - p_y\right)\sin\psi \tag{7}$$

$$VPTs_{y_i} = \left(PTs_{y_i} - p_y\right)\cos\psi + \left(PTs_{x_i} - p_x\right)\sin\psi \tag{8}$$

where $VPTs_{x_i}$ and $VPTs_{y_i}$ are the i^{th} waypoint of the reference track in vehicle coordinates, calculated using the received vehicle positions ("p_x" and "p_y") and the vehicle orientation angle "ψ".

2) An n^{th} order polynomial equation is then fitted using the transformed waypoints (in this implementation n is selected to be 3). This polynomial (Eq. (9)) now represents the "desired route/track" that the vehicle should follow precisely to find its way throughout the track.

$$y_{track} = a_o + a_1 x_{track} + a_2 x_{track}^2 \tag{9}$$

3) The Cross Track Error "*CTE*" and the error in the vehicle orientation angle "$e\psi$" are then calculated using the calculated coefficients of the fitted polynomial as follows:

$$CTE = a_o \tag{10}$$

$$e\psi = -\tan^{-1} a_1 \tag{11}$$

Note that the *SDC* position "p_x" and "p_y" are always zeros in the vehicle coordinates.

4) Six states (N_STATES = 6) are then selected to represent the *SDC-MPC* states as follows:

$$\begin{aligned} state_1 &= p_x = 0 \\ state_2 &= p_y = 0 \\ state_3 &= \psi = 0 \\ state_4 &= V_{SDC} \\ state_5 &= CTE \\ state_6 &= e\psi \end{aligned} \tag{12}$$

where p_x, p_y, and ψ are measured in vehicle coordinates. These states are then supplied to the *NMPC* solver (*IPOPT* [26]) to find the predicted sequence of the required "steering value" and "throttle value".

5) Within the *NMPC* solver, the *MPC*-control horizon is defined by the time step (i.e. $\Delta t = 0.075$ sec) and the duration in terms of the number of steps (i.e. N = 30).

6) The number of controller outputs is defined as N-OUTPUT = 2, and they are the steering (angle: δ) command as well as the throttle command (speed/brake).

5. Implementation of The SDC-MPC Algorithm

The following points shed the light on some details and specifics of the implementation of the proposed track follower:

1) The algorithm is implemented using the high-performance language GCC C++ [25] on Ubuntu Linux operating system [36]. This combination is fitting for the required real-time performance.

2) A C++ Algorithmic Differentiation Package [27] is used to numerically solve the differential equations of the *SDC* model given by Eq. (6).

3) The *NMPC* solver is implemented using the *IPOPT* package [26] which is used to solve the optimization problem of minimizing the objective function "$l(.,.)$" as given by the example in Eq. (2) and will be detailed later in the next section.

4) The number of *NMPC* solver variables is determined from the above information to be (n_vars = N-STATES*N + N-OUTPUT*(N-1)).

5) Moreover, the number of *NMPC* solver constraints is then determined from the above information to be (n_constraints = N-STATES * N).

6) The controller variables and constraints are initialized and set to reinforce the boundary conditions for the optimization problem (e.g. the steering command (δ) is constrained between -25° to 25°, and the throttle command is also constrained between -1 to 1).

7) Then the output of the controller (i.e. the solution of the optimization problem) is fed to the simulator or the actuators.

6. The NMPC Objective Function

The cost (objective) function of the *SDC-NMPC* is composed of several terms. Each term has its own sub-objective within the core optimization problem. The main goal is to find a solution that can satisfy the purposes of these terms according to their weights (contribution) in the overall objective function. The final weights of all terms

that determine their contribution are given in TABLE I. The description of the terms and their sub-objectives are listed as follows:

1) CTE term: The CTE (the cross-track error) represents the misalignment of the vehicle with respect to the center of the track at a given instance. The sub-objective of this term is to minimize the aggregation of the Cross Track Errors (*CTE*) for all prediction points (N=30) as given by Equations (13) and (14) below:

$$CTE = desired\ position - vehicle\ position \quad (13)$$

$$Sub\_Objective\ Term = \text{minimize}\{MSE\} = min\frac{1}{N}\sum_{i=0}^{i=N} CTE_i^2 \quad (14)$$

2) eΨ term: The sub-objective of this term is to minimize the aggregation of the errors ($e\Psi$) in the vehicle orientation angle, at a given instance, for all prediction points (N=30) as given by Equations (15) and (16) below:

$$e\Psi = desired\ angle - actual\ vehicle\ angle \quad (15)$$

$$Sub\_Objective\ Term = \text{minimize}\{MSE\} = min\frac{1}{N}\sum_{i=0}^{i=N} e\Psi_i^2 \quad (16)$$

3) V error term: The sub-objective of this term is to minimize the aggregation of the speed (V_{SDC}) errors with respect with the reference speed received from the *SDC* path planner (e.g. V_{ref} = 100 *mph*) for all prediction points (N=30) as given by Equations (17) and (18) below:

$$eV = reference\ speed - vehicle\ speed \quad (17)$$

$$Sub\_Objective\ Term = \text{minimize}\{MSE\} = min\frac{1}{N}\sum_{i=0}^{i=N} \mathrm{V}_i^2 \quad (18)$$

4) Speed regulation term: The objective of this term, given by Equation (19), is to help the controller manage the speed throughout the track. The main purpose of this term is to speed up when the road is straight (increase V_{SDC} while Ψ is small) and to slow down when there is a turn ahead (reduce V_{SDC} when Ψ is relatively large). The amount of slowing down is proportional to how sharp is the turn ahead.

$$Sub\_Objective\ Term = min\frac{1}{N}\sum_{i=0}^{i=N} \mathrm{V}_{SDC_i}^2\,\Psi^2 \quad (19)$$

5) Steer control term: The objective of this term, given by Equation (20), is to help the controller to optimize the control effort by not taking unnecessary sharp steering commands.

$$Sub\_Objective\ Term = min\frac{1}{N}\sum_{i=0}^{i=N} \delta_i^2 \quad (20)$$

6) Acceleration control term: The objective of this term, given by Equation (16), is to help the controller to optimize the control effort by not taking unnecessary accelerating/braking (*acc*) commands.

$$Sub\_Objective\ Term = min\frac{1}{N}\sum_{i=0}^{i=N} acc_i^2 \quad (21)$$

7) Speed-steering term: The objective of this term, given by Equation (22), is to correlate between the steering command (δ) and the actual speed (V_{SDC}) of the *SDC*. The idea is to allow the controller, when issues a relatively big steering command, to reduce the speed and vice versa.

$$Sub\_Objective\ Term = min\frac{1}{N}\sum_{i=0}^{i=N} \mathrm{V}_{SDC_i}^2\,\delta_i^2 \quad (22)$$

8) Change of steering command term: The objective of this term, given by Equation (23), is to minimize the value gap between sequential steering actuation. In other words, reduce the sudden change in the subsequent steering commands.

$$Sub\_Objective\ Term = \text{minimize}\{MSE\} = min\frac{1}{N}\sum_{i=0}^{i=N-1} (\delta_{i+1} - \delta_i)^2 \quad (23)$$

9) Change of acceleration command term: The objective of this term, given by Equation (24), is to minimize the value gap between sequential acceleration actuations. In other words, reduce the sudden change in the subsequent acceleration commands.

$$Sub\_Objective\ Term = \text{minimize}\{MSE\} = min\frac{1}{N}\sum_{i=0}^{i=N-1} (acc_{i+1} - acc_i)^2 \quad (24)$$

TABLE I. THE CONTRIBUTION OF OBJECTIVE FUNCTION TERMS.

SDC-MPC Objective Function		
Term Type	*Weight Value*	*Comment*
CTE (w_cte)	15.0	Follow path accurately
eΨ (w_espsi)	2.75	Follow path accurately
V-error (W_v_err)	0.65	Try to reach max speed
Speed regulation (W_psi_des_v)	0.50	Avoid high speed in turns
Steer control (W_delta)	50000.0	Avoid sudden steering
Acceleration control (W_acc)	10.0	Avoid aggreesive acceleration
Speed-steering (W_delta_v)	50.0	Avoid steering at high speed
Change of steering command (W_d_delta)	150.0	Avoid aggressive steering
Change of acceleration command (W_d_acc)	0.0	Avoid aggressive acceleration

Moreover, the constraints in the *NMPC* objective function are set as follows:

1) The initial points (point '0') of each controller state are initialized by the incoming state values from the SDC hardware or the simulator at this specific instance.

2) The rest of the points (from 1 => (N-1)) are constrained by the vehicle model which is given by Equation (25) that is used to update the six states (X_{k+1}, Y_{k+1}, Ψ_{k+1}, V_{k+1}, CTE_{k+1} & $e\Psi_{k+1}$):

$$\begin{aligned} &X_{k+1} - (X_k + V_k cos(\Psi_k)\Delta t) = 0 \\ &Y_{k+1} - (Y_k + V_k sin(\Psi_k)\Delta t) = 0 \\ &\Psi_{k+1} - \left(\Psi_k - \left(\frac{V_k}{L_f}\right)\delta_k \Delta t\right) = 0 \\ &V_{k+1} - (V_k + acc_k \Delta t) = 0 \\ &CTE_{k+1} - \left((f(x_o) - Y_k) + V_k sin(\mathrm{e}\Psi_k)\Delta t\right) = 0 \\ &\mathrm{e}\Psi_{k+1} - \left(\left(\Psi_k - f'(x_o)\right) - \left(\frac{V_k}{L_f}\right)\delta_k \Delta t\right) = 0 \end{aligned} \quad (25)$$

where $f(x_o)$ and $f'(x_o)$ are the waypoints polynomial and its slope values at the point '0'.

7. THE SDC-NMPC DESIGN HIGHLIGHTS

The following points have to be highlighted throughout the design process of the NMPC:

1) Waypoints Polynomial: The calculated waypoints polynomial coefficients are not used directly in the NMPC state equations, but instead they go through a pre-processing step by taking the weighted averaged of their current and previous values as shown in Equation (26). This step helps to smooth out the transition between frames and makes the waypoints polynomial more stable. The value of *K* is set to be "0.9" after several trials.

$$C_i = K * new\_C_i + (1 - K) * prev\_C_i \quad (26)$$

2) Speed and the Objective function: The reference speed (V_{ref}) is set at 100 mph which makes tuning the controller more challenging and the need for speed regulation around corners and at sharp turns is highly desirable. Therefore, two objective function terms are incorporated into the overall NMPC objective function: the "Speed regulation term" (#4) and the "Speed-steering term" (#7). The "Speed regulation term" specifically proved to be very effective as it makes the speed inversely proportional to the desired steering angle (which is large at sharp turns and makes the car slower).

3) Actuator Latency: The actuator latency is estimated to be at least 100 *msec*. This has been compensated for by simply using future actuator commands instead of the first one calculated. For example, the *SDC-NMPC* solver produces ((N-1)=29) predicted steering angle commands ($\delta_{k+1} \rightarrow \delta_{k+29}$). Instead of using δ_k as usual, for example, δ_{k+3} is used as 3*Δt = 225 *msec* > 100 *msec*. δ_{k+1}, δ_{k+2}, and δ_{k+3} have been tried and the latter proved to be more effective.

4) N and Δt selection: several trials and errors are used to determine the most appropriate values for both parameters. Values from 0.05→0.2 are tried for Δt, while values from 5→30 are tried for N. A conclusion is reached, that a long sight for the controller is a good feature that improves its performance, therefore, N = 30 is selected. In other words, if good predicted points in the near future (i.e. k = 1, 2 … 5) are required, the controller must be allowed to solve for a long stride (i.e. N > 20). Moreover, Δt needs to be small enough in order to allow, for the NMPC, not to lose track of changes, and big enough to allow for longer prediction strides. Finally, Δt=0.075 sec is selected as it is smaller than the actuator latency but big enough to have a good prediction horizon.

8. TESTING AND EVALUATION RESULTS

Extensive trials-and-errors attempts are used to tune the many hyper-parameters of the *SDC-NMPC*. However, to be more consistent and accurate, a numerical Key Performance Indicator (*KPI)* need to be constructed and coded as in Equation (27) to evaluate the performance of the controller under the given set of hyper-parameters:

$$KPI = \sum_{i=0}^{N\_Cycle} CTE_i^2 + 100\, e\Psi_i^2 \quad (27)$$

The indicator is calculated by aggregating and averaging the CTE^2 and the $e\Psi^2$ over a period of *N_Cycle* samples (e.g. *N_Cycle* = 3000, enough to let the car drive for at least one lap around the track shown in Figure 1). The $e\Psi^2$ term is multiplied by 100, to have a comparable weight with the CTE^2 term. This method helped a lot to have a more deterministic comparison between the different trials.

Several test tracks have been used to evaluate the performance of the *SDC-NMPC* under different sets of hyper-parameters in an iterative tuning process. Examples of these test tracks are shown in Figure 1 and Figure 2. Figure 1 shows "the 1st track" that represents a kind of rural road of 3.7 miles long, two-lane, 8.2 m wide, slightly hilly, max 4.5% incline, designed for max. 80 mph speed. The test track contains several straight and curved segments, as well as sharp turns. The sharpest turn has a radius of curvature of 50 meters. Likewise, Figure 2 shows "the 2nd track" of 1.122 miles long mostly curved with 4 sharp turns. The simulation test results of the 2nd track are shown in Figure 3 to

Figure **9** for convenience. The figures depict the profile of *CTE*, *eΨ*, *Speed*, *Steering* command, *Throttle* command, acceleration and jerk during a single revolution of *SDC* on the 2nd Track. These profiles show that the *SDC* was able to reach 80 mph while preserving an acceleration range of [0.84, -1.70] m/sec^2 as well as jerk range of [1.86, -1.88] m/sec^3, which are safe enough by automotive industry standards to provide comfortable ride [37].

The hyper-parameters have to be tuned manually using a dedicated simulation tool that incorporates the vehicle dynamic model explained in Section 3. This tool is developed specifically for this purpose using Unity [38] with an optimized object-oriented structured code [39] and interfaced with the *SDC-NMPC* (*C++* code) using *μWebSocket* messaging [40]. The values of the used vehicle model parameters are listed in TABLE II.

TABLE II. THE VEHICLE MODEL PARAMETERS.

Parameter	Value
l_f	2.67 meter
l_r	2.10 meter

The process of manual tuning of the parameters includes fixing the whole parameters and only changes one while measuring the KPI results after adequate simulation runs as shown in Figure 10. It is necessary to complete at least 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. Figure 10 shows an example of one of this tuning method, V_{ref} = 100 mph is selected as the design parameter value for the *SDC-NMPC*. Moreover, Figure 11 gives another example of how the weighting coefficient of the *CTE* term in the *NMPC* cost function is determined.

The performance of the *SDC-MPC* is compared to that of the carefully tuned PID controller that has been developed in [19] and tested on the "2nd track". TABLE III. summaries these comparisons as well as

Figure 12 depicts the resultant speed profiles. The results show a wonderful improvement in the peak reached speed in the track (131% higher) and average speed (53.6%). It is clear from the profiles in

Figure 12 that the NMPC is able to manage and maximize the speed throughout the track much better than the PID which only acts on lowering the speed to avoid large *CTEs* even on road segments that can accommodate higher speeds **Error! Reference source not found.**. Moreover, the NMPC shows more precision in tracking the path planner waypoints as clearly shown from the highest and the lowest KPI values.

TABLE III. NMPC/PID PERFORMANCE COMPARISON.

KPI	SDC-NMPC	SDC-PID	% Change
Highest Speed	79.5 mph	34.4 mph	+131%
Average Speed	50.1 mph	32.6 mph	+53.6%
Lowest MSE	0.6425	0.8288	-28.99%
Highest MSE	1.0411	1.4575	-39.99%

Moreover, for the purpose of better evaluation of the performance of the designed *SDC-NMPC*, the framework presented in the paper is used to design a conventional-NMPC (cNMPC) track follower. "Conventional" in the sense of only using the tracking errors (CTE and eΨ) in its cost function and eliminating the other terms (e.g. only using the 1st and 2nd terms in Section 6). Then, this cNMPC is tested on the 2nd track using the same reference speed (100 mph). However, the cNMPC failed to complete a single lap without getting out of the road boundaries. Then, the cNMPC is redesigned by adding the 3rd term (the speed-error term), however the controller still fails to complete a single lap without getting out of the boundaries of the road. The process gets iterated and each time adding a new term, till reaching the 7th term where the controller was able to complete the laps without breaching the boundaries. The resultant cNMPC controller contains all the proposed *SDC-NMPC* cost terms except the 8th term, which is "the change of the steering command term". The performance of the resultant cNMPC is then compared to that of the full *SDC-NMPC*, the results are listed in TABLE IV. and shown in

Figure 13,

Figure 14, and

Figure 15.

TABLE IV. NMPC/CNMPC PERFORMANCE COMPARISON.

KPI	SDC-NMPC	cNMPC	% Change
Highest Speed	79.5 mph	78.1 mph	+1.8%
Average Speed	50.1 mph	50.9 mph	-1.5%
Lowest MSE	0.6425	1.4842	-56.71%
Highest MSE	1.0411	1.9797	-47.41%

The above results show that there is no major difference in speed metrics between the two controllers as shown in

Figure 15 as well, or in other words, the travel time almost did not change. However, the cNMPC ride quality is much worse than that of the originally proposed *SDC-*

NMPC. This has been revealed by the reported MSE scores in TABLE IV.

Figure 13 and

Figure 14.

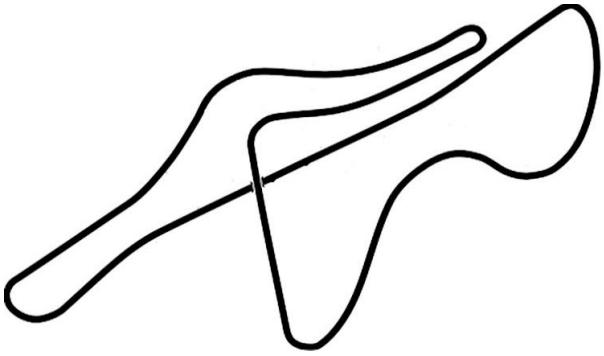

Figure 1. The 1st Test Track.

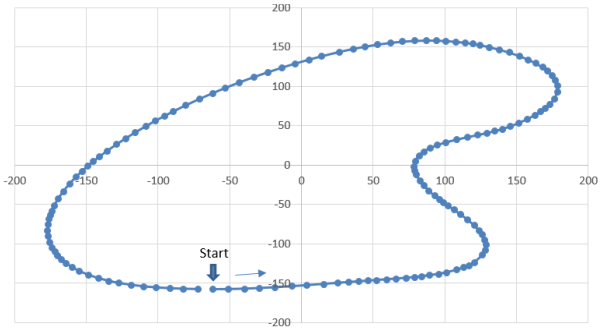

Figure 2. The 2nd Test Track.

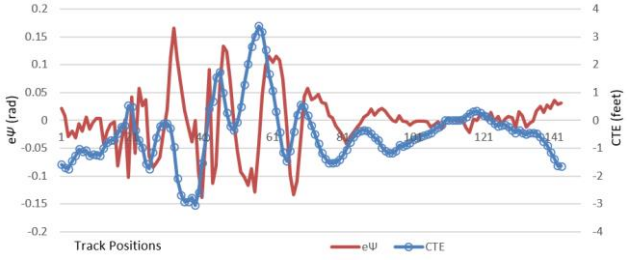

Figure 3. Both CTE and eΨ for one revolution in the 2nd Track.

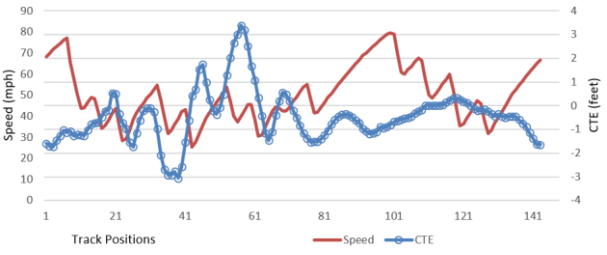

Figure 4. CTE and Speed for one revolution in the 2nd Track.

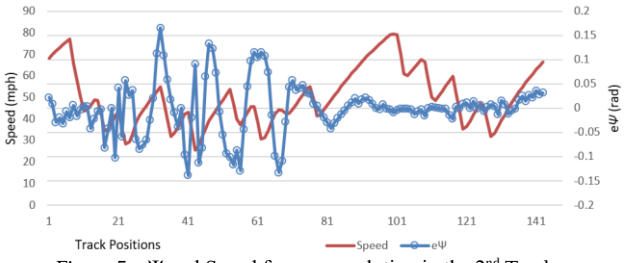

Figure 5. eΨ and Speed for one revolution in the 2nd Track.

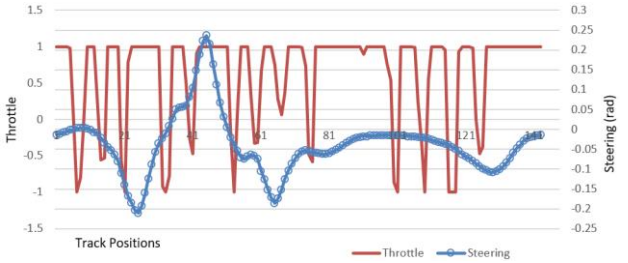

Figure 6. Throttle and Steering commands for one revolution in the 2nd Track.

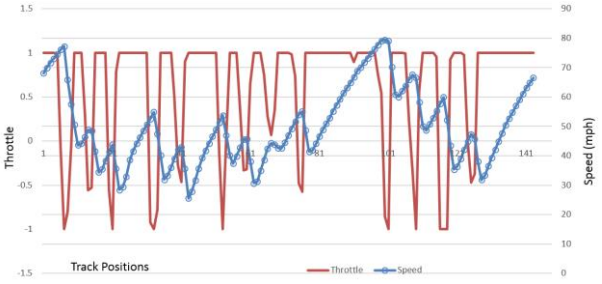

Figure 7. Throttle and Speed commands for one revolution in the 2nd Track.

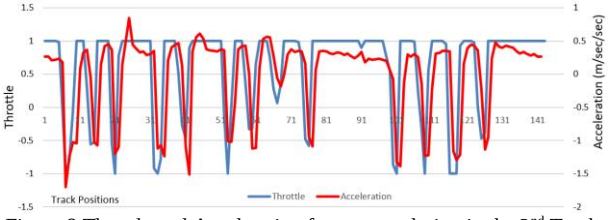

Figure 8 Throttle and Acceleration for one revolution in the 2nd Track.

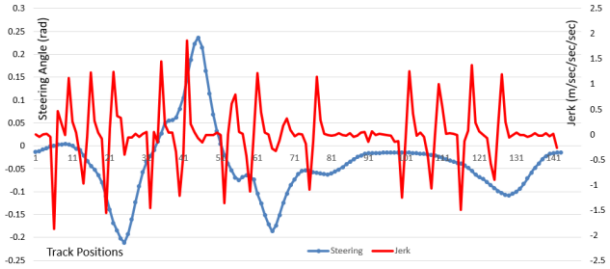

Figure 9. Steering and Jerk for one revolution in the 2nd Track.

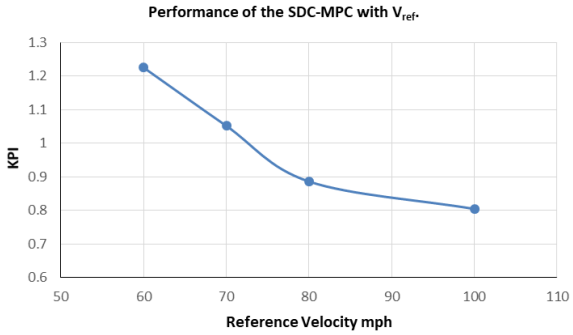

Figure 10. Performance of the SDC-MPC with respect to the speed reference.

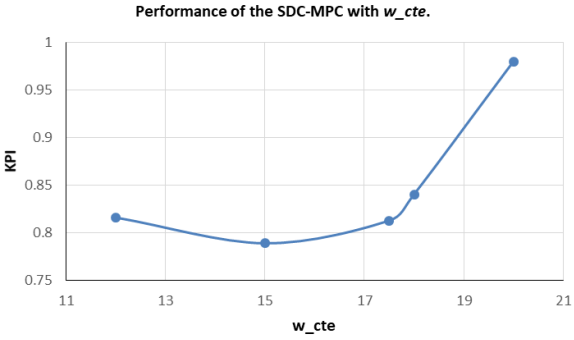

Figure 11. Performance of the SDC-MPC with respect to CTE error weight.

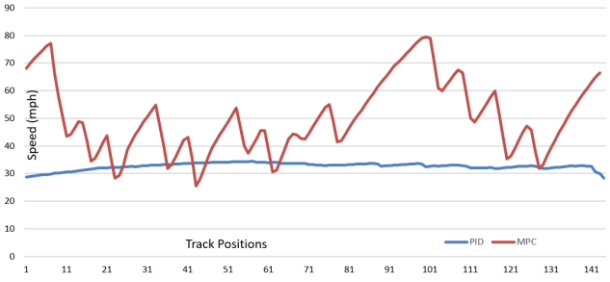

Figure 12. *PID* and *NMPC* Speed Profiles for one revolution in the 2nd Track.

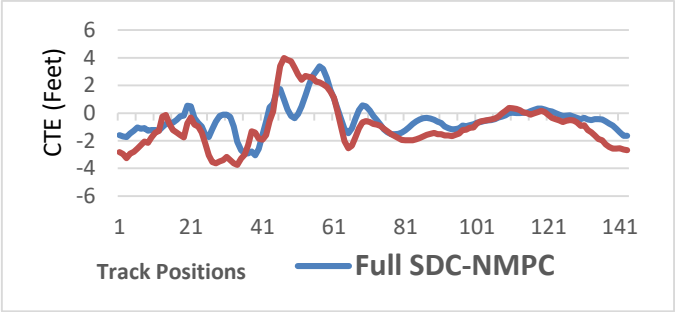

Figure 13. *SDC-NMPC* and *cNMPC CTE* scores for one revolution in the 2nd Track.

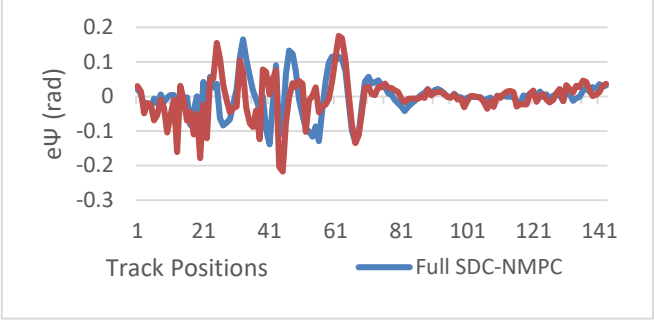

Figure 14. *SDC-NMPC* and *cNMPC eΨ* scores for one revolution in the 2nd Track.

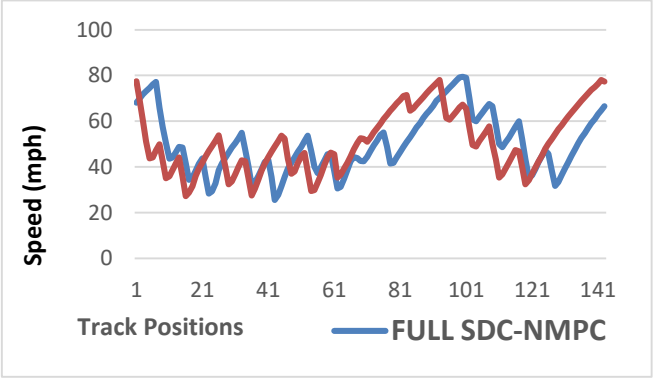

Figure 15. *SDC-NMPC* and *cNMPC* speed Profiles for one revolution in the 2nd Track.

9. DISCUSSION

The paper presents a method to satisfy the requirements of an efficient and comfortable ride for self-driving car while driving through complex tracks. The idea is that for an efficient ride the requirements are:

1) To follow the required trajectory (usually generated by the path planner) as precise as possible (with the lowest deviation) in other words, the lowest aggregated cross-track errors (CTEs).

2) To reach the destination as fast as possible (more efficient).

3) This should be done without violating the comfort requirements of the ride, which can be evaluated by monitoring the acceleration and jerk measurements. The acceleration should be maintained within (±0.2g m/sec^2) and the jerk within (±10 m/sec^3) [37].

4) This should be done as well while respecting the vehicle dynamics (constraints).

The paper formulates the above requirements by proposing a framework for constructing a Nonlinear MPC path tracker (*SDC-NMPC*) that provides a way of easily integrating several driving objectives through compounding a multi-term cost function as presented in Section 6 and TABLE I. Each term contributes to the overall objective according to its corresponding weight. Optimizing this objective function using the primal-dual interior-point algorithm, allows the controller to produce steering, throttle (gas/brake) commands that satisfy the above requirements and constraints. Another contribution of the paper is the detailed description of the real-time implementation of the NMPC and its objective function in C++.

The following conclusive points shed the light on some technical aspects that have been tried or implemented in the described approach:

1) The NMPC controller has a much more complex structure compared to that of the PID. However, it is more effective, especially at higher speeds. It can handle issues like actuator latency and external disturbances much better.

2) One of the main issues with the NMPC is its tuning. There is no theory or criteria that prove that the optimal value for the many used hyper-parameters has been selected. Most methods of tuning are mainly based on extensive search augmented with intuition and experience.

3) From the author's point of view, using transparent methods based on extensive "trial-and-error" endeavors guided by numerical performance indicators is the most convenient approach. This approach allows one to understand the problem at hand much deeper. Furthermore, it allows the incorporation of one's intuition and experience, which reduces a lot of the search space; and consequently, shows more effectiveness at the end.

4) The most powerful aspect of the design of the NMPC is the ability to tailor the cost function [41]. It gives great flexibility to the designer to balance between conflicting requirements and make a better-educated trade-off.

Future research and endeavors are encouraged to enhance this work, and the following are some of the suggested improvements:

1) Experimenting with other terms that may be added to the *SDC-NMPC* cost function such as "*maintain* δ_f**acc => reasonable value*". The idea at big steering angles the car is going to a sharp turn and needs to avoid large acceleration and vice versa.

2) One set of hyper-parameters may not be enough for all the range of speed. Therefore, a kind of adaptive MPC where it can have several sets of parameters for each speed range (similar to gain scheduling in PID) is desirable in this case.

3) Need to add or invent several other performance indicators, like a one to track the time in which the car is able to complete one lap. Another suggested one is to track overshoots and undershoots from the road center, etc.

10. CONCLUSION

In this paper, a framework for designing a non-linear model-predictive-control path follower for autonomous vehicles is proposed and described in detail. The framework uses the primal-dual interior-point technique to iteratively solve the NMPC optimization problem. Consequently, the whole framework is developed using C++ in addition to advanced math libraries to optimize real-time performance. Furthermore, the framework is used to design and implement an *SDC-NMPC* with tailored cost function that handles several objectives simultaneously (either opposing or correlating) to emphasize precision, comfort, and efficiency. The designed *SDC-NMPC* receives waypoints of the reference trajectory from the path planner, the actual location of the SDC in global coordinates from the localization module, the instantaneous speed and steering angle measurements from the SDC associated sensors. The output is then the steering and the gas/brake (throttle) commands. The performance of the *SDC-MPC* track follower is evaluated through extensive simulations in complex tracks with sharp turns. The performance is also compared to that of the classical PID controller showing superior performance.

ACKNOWLEDGMENT

This work used the High-Performance Computing (HPC) facilities of the American University of the Middle East, Kuwait.

REFERENCES

[1] Wael Farag, "Traffic signs classification by deep learning for advanced driving assistance systems", *Intelligent Decision Technologies*, IOS Press, vol. 13, no. 3, pp. 215-231, (2019).

[2] Wael Farag, Zakaria Saleh, "Road Lane-Lines Detection in Real-Time for Advanced Driving Assistance Systems", *Intern. Conf. on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT'18)*, Bahrain, 18-20 Nov., (2018).

[3] Wael Farag, Zakaria Saleh, "Behavior Cloning for Autonomous Driving using Convolutional Neural Networks", *Intern. Conf. on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT'18)*, Bahrain, 18-20 Nov., (2018).

[4] 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).

[5] Wael Farag, Zakaria Saleh, "Tuning of PID Track Followers for Autonomous Driving", *Intern. Conf. on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT'18)*, Bahrain, 18-20 Nov., (2018).

[6] Wael Farag, "Safe-driving cloning by deep learning for autonomous cars", *International Journal of Advanced Mechatronic Systems*, Inderscience Publishers, vol. 7, no. 6, pp. 390-397, (2019).

[7] Wael Farag, "Cloning Safe Driving Behavior for Self-Driving Cars using Convolutional Neural Networks", *Recent Patents on Computer Science*, Bentham Science Publishers, The Netherlands, Vol. 12, No. 2, pp. 120-127(8), (2019**)**.

[8] Wael Farag and Zakaria Saleh, "An Advanced Vehicle Detection and Tracking Scheme for Self-Driving Cars", *2nd Smart Cities Symposium (SCS'19),* IET Digital Library, Bahrain, 24-26 March, (2019).

[9] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar and B. Litkouhi, "Towards a Viable Autonomous Driving Research Platform", *IEEE Intelligent Vehicles Symposium (IV)*, Australia, 23-26 June, (2013).

[10] G. Bai, Y. Meng, L. Liu, W. Luo, Q. Gu, and L. Liu, "Review and Comparison of Path Tracking Based on Model Predictive Control", *electronics*, MDPI, vol. 8, Sept., (2019).

[11] Wael Farag, "Track Maneuvering using PID Control for Self-Driving Cars", *Recent Advances in Electrical & Electronic Engineering*, 13 (1), pp. 91-100, 2020.

[12] T. Ikeda and M. Nagahara, "Discrete-Valued Model Predictive Control Using Sum-of-Absolute-Values Optimization", *Asian Journal of Control*, vol. 20, no. 2, pp. 1-11, March, (2018).

[13] N. F. Silva, C. E. T. Dórea and A. L. Maitelli, "An iterative model predictive control algorithm for constrained nonlinear systems", *Asian Journal of Control*, vol. 21, no. 5, pp. 1–15, Sept., (2019).

[14] A. Wächter and L.T. Biegler, "On the Implementation of a Primal-Dual Interior-Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming", *Mathematical Programming*, vol. 106, pp. 25-57, (2006).

[15] B. Vatankhah and M. Farrokhi, "Nonlinear Adaptive Model Predictive Control of Constrained Systems with Offset-Free Tracking Behavior", *Asian Journal of Control*, vol. 20, no. 1, pp. 1–13, January, (2018).

[16] Wael Farag, "Synthesis of intelligent hybrid systems for modeling and control", Ph.D. Thesis, *Universty of Waterloo*, Canada, 1998.

[17] Wael Farag, Ahmed Tawfik, "On fuzzy model identification and the gas furnace data", *Proceedings of the IASTED International Conference Intelligent Systems and Control*, Honolulu, Hawaii, USA, August 14-16, (2000).

[18] W. Farag, Z. Saleh, "MPC Track Follower for Self-Driving Cars", *2nd Smart Cities Symposium (SCS'19)*, IET Digital Library, Bahrain, 24-26 March, (2019).

[19] W. Farag, "Complex Trajectory Tracking Using PID Control for Autonomous Driving", *International Journal of Intelligent Transportation Systems Research*, Springer, Sept., (2019).

[20] W. Farag, "Complex-Track Following in Real-Time Using Model-Based Predictive Control", *International Journal of Intelligent Transportation Systems Research*, Springer, (2020).

[21] N. Wang, S.-F. Su, X. Pan, X. Yu, and G. Xie, "Yaw-Guided Trajectory Tracking Control of an Asymmetric Underactuated Surface Vehicle", *IEEE Trans. Industrial Informatics*, Vol. 15, No. 6, June, (2019).

[22] N. Wang, G. Xie, X. Pan, X. Yu, and S.-F. Su, "Full-State Regulation Control of Asymmetric Underactuated Surface Vehicles", *IEEE Trans. Industrial Electronics*, Vol. 66, No. 11, Nov., (2019).

[23] N. Wang, H.R. Karimi, H. Li, and S.-F. Su, "Accurate Trajectory Tracking of Disturbed Surface Vehicles: A Finite-Time Control Approach", *IEEE/ASME Trans. Mechatronics*, Vol. 24, No. 3, June, (2019).

[24] N. Wang, H.R. Karimi, "Successive Waypoints Tracking of an Underactuated Surface Vehicle", *IEEE Trans. Industrial Informatics*, June, (2019).

[25] GCC C++, https://gcc.gnu.org/, accessed on 11th Feb, (2019).

[26] IPOPT, https://en.wikipedia.org/wiki/IPOPT, accessed on 11th Feb, (2019).

[27] A C++ Algorithmic Differentiation Package (CppAD), https://coin-or.github.io/CppAD/doc/cppad.htm, accessed on 11th Feb, (2019).

[28] J. Robert Vanderbei, "Linear Programming: Foundations and Extensions", *Kluwer*, pp. 277–279, (2001).

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

[30] W. Rahiman, and Z. Zainal, "An overview of development GPS navigation for autonomous car", *8th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Melbourne, Australia, 19-21 June, (2013).

[31] D. Gohring, M. Wang, M. Schnurmacher, T. Ganjineh, "Radar/Lidar Sensor Fusion for Car-Following on Highways", *5th Intern. Conf. on Automation, Robotics, and Applications, ICARA 2011*, Wellington, New Zealand, 6-8 Dec., (2011).

[32] "Speedometer", https://en.wikipedia.org/wiki/Speedometer, *Wikipedia*, accessed on 9th Feb, (2019).

[33] M. Kok, J. D. Hol, and T. B. Schon, "Using Inertial Sensors for Position and Orientation Estimation", *Foundations and Trends in Signal Processing*, vol. 11, no. 1-2, pp 1-153, (2017).

[34] "Rotation and Orientation in Unity", https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html, accessed on 9th Feb, (2019).

[35] S.G. Anavatti, S. LX Francis, M. Garratt, "Path-planning modules for Autonomous Vehicles: Current status and challenges", *Inter. Conf. on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, Surabaya, Indonesia, 15-17 Oct., (2015).

[36] Ubuntu Linux, https://www.ubuntu.com/, accessed on 11th Feb, (2019).

[37] L.L. Hoberock, "A survey of longitudinal acceleration comfort studies in ground transportation vehicles", Dept. of Transportation, Washington DC, USA, (1976).

[38] Unity, https://unity.com/solutions/automotive-transportation?_ga=2.238996096.1822638216.1551163213-250725045.1549710749, accessed on 26th Feb, (2019).

[39] M. Nagiub and W. Farag, "Automatic selection of compiler options using genetic techniques for embedded software design", *IEEE 14th Inter. Symposium on Comp. Intelligence and Informatics (CINTI)*, Budapest, Hungary, Nov. 19, (2013).

[40] µWebSocket, https://github.com/uNetworking/uWebSockets, accessed on 26th Feb, (2019).

[41] Wael A Farag, VH Quintana, G Lambert-Torres, "Genetic algorithms and back-propagation: a comparative study", *IEEE Canadian Conf. on Elec. and Comp. Eng.*, vol. 1, pp. 93-96, Waterloo, Ontario, Canada, (1998).

Wael Farag earned his Ph.D. from the University of Waterloo, Canada in 1998; M.Sc. from the University of Saskatchewan, Canada in 1994; and B.Sc. from Cairo University, Egypt in 1990. His research, teaching and industrial experience focus on embedded systems, mechatronics, autonomous vehicles, renewable energy, and control systems. He has combined 17 years of industrial and senior management experience in Automotive (Valeo), Oil & Gas (Schneider) and Construction Machines (CNH) positioned in several countries including Canada, USA & Egypt. Moreover, he has 10 Years of academic experience at Wilfrid Laurier University, Cairo University, and the American University of the Middle East. Spanning several topics of electrical and computer engineering. He is the holder of 2 US patents; ISO9000 Lead Auditor Certified and Scrum Master Certified.