Trajectory Following and Adaptive Cruise Control under Simultaneous Obstacle Avoidance

A.Paunikar, M.Amine, and S.Ghiya
anshpkr, mohie, gswapnil @umich.edu

***Motivation*— The autonomous vehicle problem can be subdivided into multiple branches of sub-problems; with increasing levels of complexity that converge towards real world dynamics. This project is a continuation of prior work that solved the Trajectory Following (TF) problem via PID control. The problem is extended to include simultaneous Adaptive Cruise Control (ACC) and Obstacle Avoidance (OA). Under the necessity to predict the vehicle's dynamics in order to impose spatial constraints, the team is motivated to tackle all of the three problems using Model Predictive Control (MPC). This paper walks the reader through the three problems in the order they are tackled, and explains the chronological progress of each problem towards having a functional MPC solution.**

I. Problem Description [MOHIEDDINE]

The project's main objective is to successfully simulate two vehicles driving between the bounds of a track while they maintain a fixed separation, and while they avoid obstacles in the center of the road. The track is an open-source model from the University of Michigan [1], representing the technical section between sector 1 and 2 at the Circuit Of The Americas, in Austin TX. Information of the positions of the track's borders and center-line, as well as the orientation of the center-line is provided. The problem is 2D, where the car is represented by a polygon, and the obstacles are circles of 3m radii. The obstacles are randomly generated and only require a flag that sets their amount on the track.

II. Simulation Environment and Control Settings [MOHIEDDINE]

A. Nomenclature and Variable Definitions

Due to the presence of two vehicles with identical dynamics, it is important to set a clear definition of the variables used in the paper. The paper refers with C1 to all states and parameters that are related to Car1, which is the lead vehicle tackling the trajectory following problem. Consequently, C2 indicates the states and parameters specific to Car2. Obs refers to obstacles, WP refers to waypoints, and Cline refers to the center line, which is essentially a discrete array of center points in a counterclockwise order accross the circuit.

B. Data Streaming

Simulation is executed in Matlab. The data is streamed across six main data structures.

- SIM: Contains simulation environment variables like timing and sim settings.
- TRAJ: Trajectory info - waypoint and nearest cline point.
- DATA_C1: Contains Car 1's current + past states.
- DATA_C2: Contains Car 2's current + past states.
- MPC: Contains our settings for our controller that involve cost matrices and horizon/time-step settings.
- GRAPHICS: Contains information for visualization.

Refer to the the Appendix in Section IX for code displaying the definition and implementation of the data structures.

C. The Dynamics Model

The project uses the simple nonlinear bicycle model of a FWD vechicle to represent the dynamics of the vehicle [2]. Equations (1) to (3) show that the model is of three states which are the Cartesian coordinates of the car $[x, y]$, and the yaw ψ. The input to the model is the velocity of the car u (immediate control over velocity is assumed), and the steering angle δ. The non-linearity of the model necessitates to either linearize it for usage with linear MPC, or to use non-linear MPC.

$$\dot{x} = [-\frac{l_2}{l_1} sin(\delta) sin(\psi) + cos(\delta) cos(\psi)]u \quad (1)$$

$$\dot{y} = [\frac{l_2}{l_1} sin(\delta) cos(\psi) + cos(\delta) sin(\psi)]u \quad (2)$$

$$\dot{\psi} = [\frac{1}{l_1} sin(\delta)]u \quad (3)$$

Table I defines the constants involved in the model and their corresponding numerical values used.

TABLE I
MODEL CONSTANTS

Symbol	Physical Representation	Value
l_1	Wheel Base[m]	1.6
l_2	Rear Wheel Center to COG [m]	0.8

D. Setting [X,Y] References

The spatial nature of the TF and ACC problems necessitates the usage of reference states, as they require the x and y states of the car to follow specified references, instead of stabilizing around the origin. Enforcing a ψ_{ref} is not necessary for the success of the control problem; however, the paper discusses how enforcing it further simplifies each corresponding problem (in their respective sections).

Waypoint Logic for TF

An efficient way to reduce the computational complexity of the trajectory following problem is to split the track into sub-waypoints: Instead of feeding in the final position on the track as a fixed goal - which enforces computation of solutions that respect **all** track bounds - the track is subdivided into several "final positions" that act as waypoints. Waypoints are obtained by iterating over the discrete center-line points of the track. Once the car is within an L2 threshold distance d_{c2wp} from a goal waypoint, the waypoint updater continues its iterations over the center-line array, till it finds the next center point that is at a different L2 threshold distance d_{wp2wp} from the previous waypoint. Algorithm 1 demonstrates the logic in pseudo code.

Algorithm 1 Waypoint Update Pseudo Code

```
1: if $||XY(C1) - XY(Wp)||_2 < d_{c2wp}$ then
2:     $Update\_wp = true$
3:     $Wp_{prev} = Wp$
4: while $Update\_wp$ do
5:     $idx = idx + 1$
6:     $Wp = Cline(idx)$
7:     if $||XY(Wp) - XY(Wp_{pref})||_2 > d_{wp2wp}$ then
8:         $Update\_wp = false$
9:         $[xC1_{ref}, yC1_{ref}] = [X_{Wp}, Y_{Wp}]$
```

Car1 as Ref for Car2

As is intuitively expected, the lead car will explicitly be the reference that the chase car follows. There is no complexity in the positional references being fed to the chase car. At any instance in time, equation (4) shows that the [X,Y] reference of the chase car is itself the [X,Y] position of the lead car.

$$[xC2_{ref}, yC2_{ref}] = [xC1, yC1] \tag{4}$$

E. The Controller Settings

All the MP Controllers share the same settings in regards to time steps and their horizon. They also share the same Q and R weighting that penalizes input and states. Table II lists out all the parameters involved in the MPC and their numerical values.

TABLE II
MPC PARAMS

Symbol	Physical Representation	Value
N	Horizon Steps	20
T	Time Step [s]	0.05
Q	States Cost	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
R	Input Cost	$\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

It is also worthy to note that no terminal cost is imposed, following the logic that the reference continuously updates at each step of time (for the chase car) and after a certain amount of time steps for the lead car.

F. Control Structure and Ode45 Simulation

The control problem and consequently the simulation runs until the lead vehicle reaches the final center point on the track. Each loop updates references, the control inputs, and the states of the cars.

1) Run waypoint update to update ref of car1.
2) Run car1 MPC to obtain control input U1.
3) Run car2 MPC to obtain control input U2.
4) Simulate car1 via ode45 given previous iteration state and U1.
5) Similarly simulate car2.
6) Check if car1 is at the finish line, if so stop, otherwise repeat.

III. TRAJECTORY FOLLOWING - LINEAR MPC [SWAPNIL]

The simplified bicycle model stated above was used to create an augmented model to formulate Linear Quadratic MPC problem. The first step is to linearize the nonlinear model. The model is linearized around the states and input at each time step. Since the prediction horizon is considered small and the vehicle is also bounded to move slowly, the linearization is valid around the states as the equilibrium at each time step.

Linearized Continuous Time Model is shown below

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = [A] \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} + [B] \begin{bmatrix} u \\ \delta \end{bmatrix} \tag{5}$$

Where,

$$A = \begin{bmatrix} 0 & 0 & -u(cos\delta sin\psi + (cos\psi sin\delta)/2) \\ 0 & 0 & u(cos\delta cos\psi - (sin\delta sin\psi)/2) \\ 0 & 0 & 0 \end{bmatrix} \quad (6)$$

$$B = \begin{bmatrix} cos\delta cos\psi - \frac{sin\delta sin\psi}{2} & -u\frac{cos\delta sin\psi}{2} + cos\psi sin\delta \\ cos\delta sin\psi + \frac{cos\psi sin\delta}{2} & u\frac{cos\delta cos\psi}{2} - sin\delta sin\psi \\ \frac{5}{8}sin\delta & \frac{5}{8}ucos\delta \end{bmatrix} \quad (7)$$

The linearized model is converted into discrete time for simulation purposes. The next step is to define a reference tracking problem. The references are obtained as the coordinates of the center line above a certain defined threshold and the orientation data associated with that coordinate. Then an augmented model is developed in accordance with the reference tracking formulation discussed in the class.

$$X = \begin{bmatrix} x & y & \psi & x_{ref} & y_{ref} \end{bmatrix} \quad (8)$$

$$A_{aug} = \begin{bmatrix} A & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$B_{aug} = \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} \quad (10)$$

Once the augmented model is obtained, the nest step is to formulate the cost function as a Quadratic minimization Problem. The idea was to minimize the cost function using quadprog.m function available in Matlab.The Q matrix is modified as following

$$Q_{mpc} = E^T Q E \quad (11)$$

where,

$$E = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix} \quad (12)$$

The cost function is defined as following :

$$J = U^T H U + 2q^T U \quad (13)$$

Minimizing this cost function drives the error between the current state and the reference to zero.The G,W,T,S,H matrices were formed using the formQPMatrices function developed in one of the homeworks.Finally the parameters are passed in the quadprog as equations (14) and (15) show

$$x_{quadprog} = quadprog(H, Lx0, G, W + Tx0) \quad (14)$$

$$U_{qprog} = IMPC * x_{quadprog} \quad (15)$$

The control input sequence thus generated is the optimal control inputs for the predicted horizon. The first control input is selected.Using ODE45 the vehicle motion is simulated using the selected control input and the next state is obtained over which the model is linearized and the next MPC iteration is carried out.The reference points are updated to the next one once the vehicle comes in close proximity of the current reference point .

The simulation is carried out and unusual behavior of the vehicle is observed. The vehicle looses control over the steering and the heading of the vehicle updates to undesirable and unusual values. This motivates to look back into the linearization of the model. On careful examination it is found that the model used is non-holonomic and therefore upon linearizing around certain points, the linearized model obtained is uncontrollable.

Therefore the next approach is to formulate a non linear MPC problem

IV. Trajectory Following - Non-Linear MPC [ANSHUL]

With the linear MPC failing to provide the desired behavior of consistently moving towards waypoints while respecting the bounds of the track, the problem switches to utilize nonlinear optimizers, and consequently undergo nonlinear MPC. This translates to re-formulating the cost function to explicitly include the nonlinear dynamics of the vehicle, and impose the constraints on states and control via nonlinear formulation.

Similar to the linear MPC formulation, the stack of control input over the horizon $U_N = [[u_1, \delta_1]; \ldots; [u_N, \delta_N]]$ is the decision variable to be obtained as the solution of the nonlinear optimization problem. The stack of states over the horizon is predicted via euler integration of the nonlinear dynamics, given the initial state and the stack of control input. Equations (16) through (18) explicitly formulates the euler integration.

$$X_n = F([x_o, y_o, \psi_o], U_N) = \begin{bmatrix} x_1, y_1, \psi_1 \\ \vdots \\ x_k, y_k, \psi_k \\ \vdots \\ x_N, y_N, \psi_N \end{bmatrix} \quad (16)$$

$$F = \begin{bmatrix} f([x_o, y_o, \psi_o], u_o) \\ \vdots \\ f([x_{k-1}, y_{k-1}, \psi_{k-1}], u_{k-1}) \\ \vdots \\ f([x_{N-1}, y_{N-1}, \psi_{N-1}], u_{N-1}) \end{bmatrix} \quad (17)$$

$$f = \begin{bmatrix} x_{k-1} + \dot{x_k} Ts \\ y_{k-1} + \dot{y_k} Ts \\ \psi_{k-1} + \dot{\psi_k} Ts \end{bmatrix}^T \quad (18)$$

Where the state derivates are obtained immediately from plugging in the previous state and control input into the dynamics equations in equations (1) through (3).

The cost function in equation (19) is thus set up exclusively as a function of the control inputs, the initial state, and the references.

$$J(x_o, U_N) = X_{cost}(x_o, U_N) + U_{cost}(U_N) \tag{19}$$

$$X_{cost} = (X_N - X_{ref})^T Q_{stck}(X_N - X_{ref}) \tag{20}$$

$$U_{cost} = (U_N - U_{ref})^T R_{stck}(U_N - U_{ref}) \tag{21}$$

$$Q_{stck} = \begin{bmatrix} Q & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & Q \end{bmatrix}, R_{stck} = \begin{bmatrix} R & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & R \end{bmatrix} \tag{22}$$

Note that the subtraction with X_{ref} representes a subtraction of each state vector obtained from F by a reference state vector.

With the cost function well defined, the problem shifts towards defining the nonlinear bounds. The bounds focus on the control input and the heading. The constraints are exclusively inequality constraints, with no requirements to enforce any equality constraints.

- Bound on the control inputs $|u| < 20 \quad [m/s]$ and $|\delta| < 0.5$
- Bound on the heading error from the center line $|\phi_{C1} - \phi_{cline}| < 0.1 \quad [rad]$

With the controller forced to find solutions that respect the center line's orientation with very little slack, the problem gets majorly simplified. The tactic safeguards for deviation away from the track and it results in 1) fewer constraints to satisfy, and consequently 2) faster computation times for a TF control iteration.

To provide the solver with the bounds in a meaningful fashion, the nonlinear constraint function is formed based on equation (V)

$$h_{ineq} = \begin{bmatrix} |U_N| - U_{bounds} \\ |X_N(:,3) - X_{ref}(:,3)| - err_\phi \end{bmatrix} \leq 0 \tag{23}$$

Finally, MATLAB's nonlinear solver fmincon [3] is called with the formulated nonlinear cost and constraints functions. For each loop of nMPC, the solver is warm-started by being provided the solution from the previous iteration as the initial guess. In the first iteration, the initial guess is formed by repeating the initial control input over the entire horizon.

The solver by default uses the interior point method for finding an optimum. It is also defaulted to 3000 function evaluations, which is capable of being reduced to 500 with no apparent effect on experimental results (yet a significant impact on computation time).

Figures 1 through 3 demonstrate how the MPC for the TF problem successfully manages to have the car navigate the track while respecting the limits imposed on the control input.

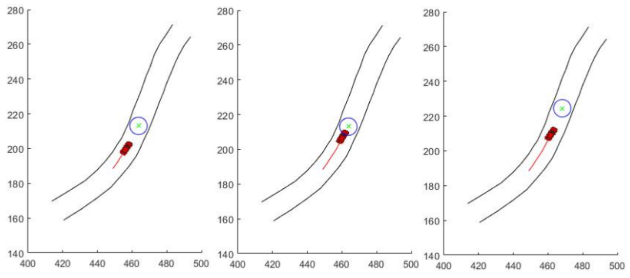

Fig. 1. Demonstration of the lead car successfully following waypoints. The figures also provide a sense of the distance that separates two consecutive waypoints. Axes represent X and Y distances in [m]

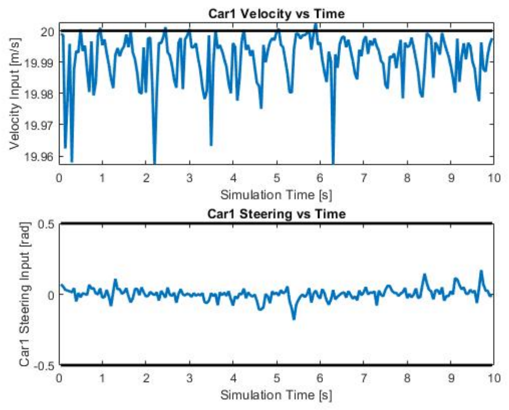

Fig. 2. Demonstration of the control inputs limitation being respected by the lead car for the TF problem with no obstacles included.

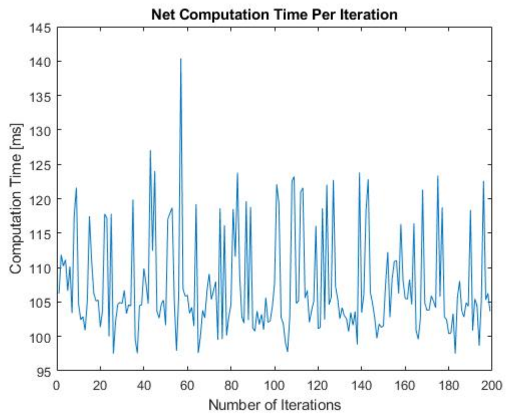

Fig. 3. Demonstration of the computation time needed for an MPC iteration for the lead car per loop. Maximum recorded timing of 140ms. Average timing hovering around 110 ms.

V. Adaptive Cruise Control - Non-Linear MPC [SWAPNIL]

The Adaptive Cruise Control problem is also solved using Non linear MPC and is formulated very similar to the trajectory synthesis problem. The idea is that the following vehicle has the same dynamics and general constraints to stay within the track as the lead vehicle. The additional constraints mainly include 1) The reference points which in this case will be the lead vehicle itself rather than a point on the center line. 2) A constraint is set for a minimum distance from the lead vehicle to adhere to ACC conditions. A maximum distance constraint is also set to make the vehicle follow the lead vehicle continuously and stay within the vicinity. 3) Because the reference is taken as the lead car position, additional constraint for a minimum distance from the closest point on the c line is defined to keep the following car near the center of the track. The maximum velocity for the following vehicle is set higher than the lead car to simulate the Adaptive Cruise Control behavior when the following car comes close to the lead car. 4)The track has sharp curves and depending on the distance between the two cars, the heading of the lead car might be completely different from that of the following car at that time instant. Therefore , the reference heading for the follower car is was used as the heading data for the closest point on the center line rather than the heading of the lead car while the reference position was the position of the lead car.

$$h_{ineq} = \begin{bmatrix} |U_N| - U_{bounds} \\ err_{lb} - |Pos_{car2} - Poscar1|_2 \\ |Pos_{car2} - Pos_{car1}|_2 - err_{ub} \\ |Pos_{car2} - Cline_{ref}(:,1:2)| - err_{cline} \\ |X_N(:,3) - Cline_{ref}(:,3)| - err_{\phi} \end{bmatrix} \leq 0 \tag{24}$$

The reference position taken as the position of the lead car is kept fixed for the computational horizon and is not updated for each step in the horizon as the lead car moves. This assumption works well since the prediction horizon is kept small and the distance between the two cars is also kept within a certain threshold. This assumption helps to simplify the problem and reduce the computation time.

VI. Obstacle avoidance - Non-Linear MPC

Obstacles are treated as added constraints for the optimization problem. This applies for both vehicles. The challenge with the obstacle avoidance lies in the increased computational requirement for equation (25) that

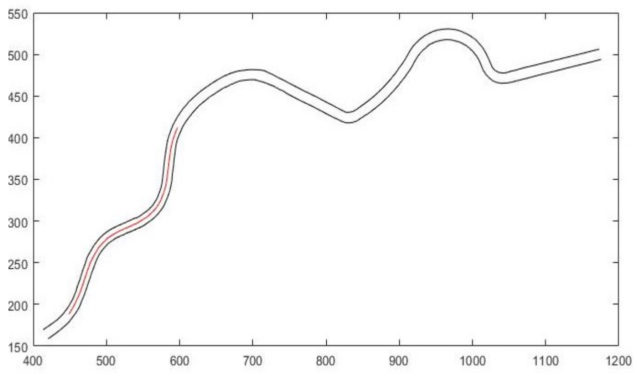
Fig. 4. The section of the track in which the car in figure 5 have reached. This graph is simply to prove the sucess of the ACC demonstrated above on a large portion of the track.

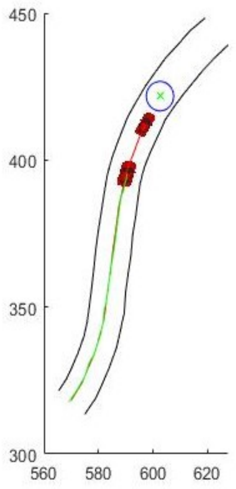
Fig. 5. Demonstration of the two cars exhibiting chase behavior in simulation. Axes represent X and Y positions in [m]

ensures that the L2 norm of all predicted states respects the obstacles.

$$\| \begin{bmatrix} x_{car} - x_{obs} \\ y_{car} - y_{obs} \end{bmatrix} \|_2 \geq thresh \tag{25}$$

Similar to the tactic of splitting the track into sub-objectives for the TF problem, the obstacles are incorporated into the MPC problem through a nearest search algorithm (Algorithn 2). Only the obstacle that is nearest to each car in L2 norm distance will be involved in the nonlinear bounds function. This removes redundancy in comparing with each obstacle on the track, which considerably hinders the computational effort of the optimizer.

Although not implemented, the algorithm can be even further enhanced by ignoring the obstacle problem entirely if the closest obstacle to the car lies beyond an L2 norm threshold - that is - there is no valid concern

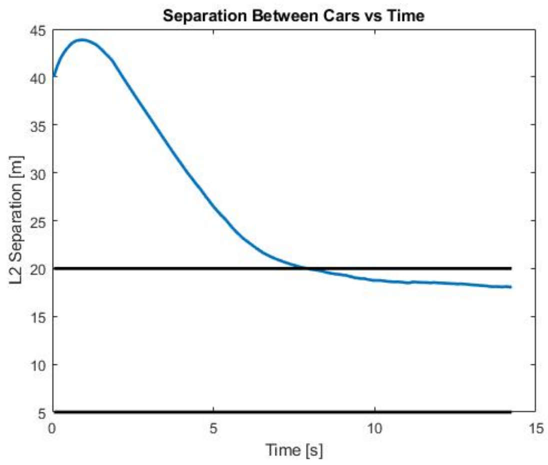

Fig. 6. Demonstration of the car separation reducing to within the restricted range for 14 seconds of simulation time. The two black lines represent the desired distance range to maintain between the two cars.

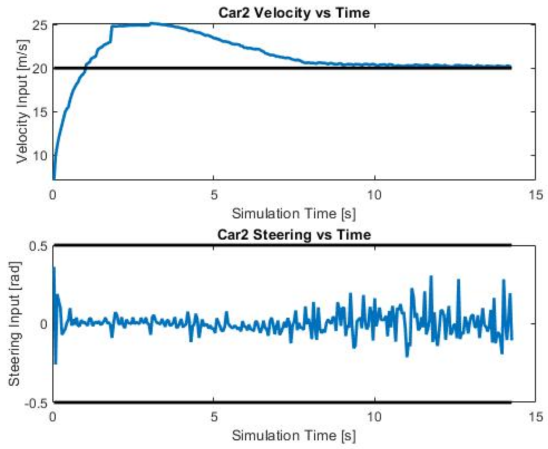

Fig. 7. Demonstration of the evolution of the control input of the chase vehicle as it approaches the lead car. An interesting observation is that the nonlinear optimizer enforced slack on the velocity limit to ensure that the chase vehicle keeps up. This is a direct result of the lead vehicle moving at a constant velocity which is the maximum set velocity for each of the two vehicles.

to involve the obstacles in the optimization problem if the closest obstacle is not even within a concerning range. However, considerations to evaluate the distance of each predicted state from the obstacle arise (maybe the predicted states do end up becoming conerning, unlike the initial state). Thus care should be taken in the choice

Algorithm 2 Nearest Obstacle Search Pseudo Code

```
1: for Each Loop do
2:     $vec(d_{obs}) = norm(vec(Pos_{allObs}).-Pos_{car})$
3:     $closest\_index = find\_index(min(vec(d_{obs})))$
4:     $closest\_Pos_{obs} = Pos_{allObs}(closest\_index)$
```

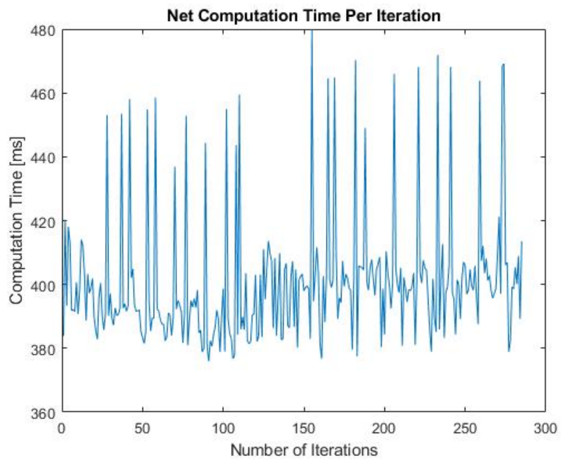

Fig. 8. Demonstration of the NET computation time for an MPC loop for BOTH vehicles. The computations average around 410 [ms] of computation time, and peak around 480 [ms] of computation time. When judging such data, it is important to take into consideration an important factor: In a real-time application, each vehicle runs it's own independent controller on an independent (and very likely more powerful) computation system.

of that L2 norm distance.

Consequently, solving for the obstacle avoidance problem in parallel to TF and ACC boils down to augmenting h_{ineq} as in equation (26) to take into consideration the obstacle bounds.

$$h'_{ineq} = \begin{bmatrix} h_{ineq} \\ h_{obs} \end{bmatrix} \leq 0 \tag{26}$$

Where h_{obs} is formed in equation (27) as a consequence of equation (25)

$$h_{obs} = thresh - \| \begin{bmatrix} x_{car} - x_{obs} \\ y_{car} - y_{obs} \end{bmatrix} \|_2 \leq 0 \tag{27}$$

Figure 13 represents graphical proof of all three problems successfully running simultaneously. The success of the three simultaneous problems raises multiple interesting questions regarding the operation of the optimization function with nonlinear cost and bounds provided. The phenomenon of the solver inducing slack to satisfy bounds is even more apparent with the obstacle avoidance running as the chase vehicle's obstacle avoidance bounds clash with its centerline orientation deviation bounds. Yet empirical results showed that the solution successfully rejects the centerline limits bounds in favor of correctly avoiding the obstacle. An even more interesting observation is cases where the obstacle avoidance for each car enforces two different directions around the obstacle, but that can be rationalized with the different evolution of references for each car.

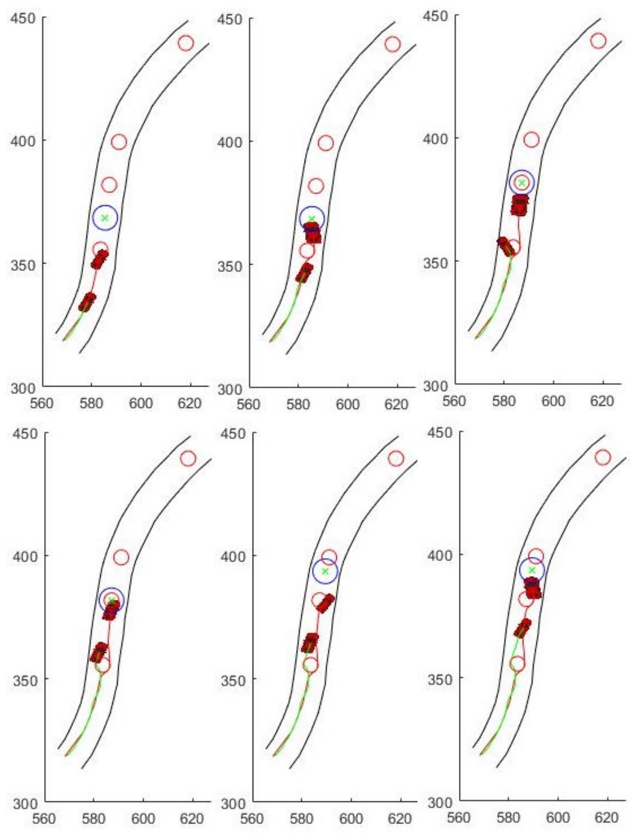
Fig. 9. Demonstration of obstacle avoidance functioning desireably for both vehicles (obstacles in red, waypoints in blue), with both the ACC and TF systems functioning.

VII. Remarks and Discussions [SWAPNIL]

A. Connection to Real-time Application

This problem represents a complex situation which is very close to what happens in the real world. Tackling each problem separately is only a simplified version of the above problem. Real world situations involve complex situations where an pedestrian might come in as an obstacle while following other vehicle in a highway or a slow moving traffic condition. Therefore, simultaneous testing of such functionality is necessary and this project successfully demonstrates a similar situation.

B. Rationalizing the success of nonLinear MPC vs failure of Linear MPC

As discussed before, failure of the Linear MPC was due to the loss of controllability of the linearized model around certain equilibrium points. The simplified bicycle model used for simulating the problem was a non-holonomic model. This was the cause for the failure of the Linear MPC approach. Non Linear MPC does not involve any form of linearization and therefore proved to be a successful approach with the model used.

C. Possible directions to continue the work

The time history of inputs to the following vehicle change abruptly. Better formulation with constraints on the rate of change of the inputs should be explored to make the control inputs smooth.

The model used to approach the problem above is a simple kinematic model with three states. More complex models which involve up to six states and also include lateral and longitudinal dynamics can be used to simulate the same problem. This will make the problem more complex with a large number of constraints and high computation time.

The prediction horizon is considered very small and the speed of the vehicles is limited to a very conservative value. The formulation can be made more robust so that the prediction horizon can be increases as well as cases of vehicles with higher speed can be simulated

VIII. Connection To Literature [ANSHUL, MOHIEDDINE]

Most of the research involving Adaptive Cruise Control (ACC), trajectory following, Obstacle Avoidance, Lane Keep Assist (LKA) have been performed by experts, however they are mostly handled individually. This project deals with an integrated MPC controller for trajectory following and ACC. The literature shows works by people trying to use 2 different vehicle models and using either a coupled or a decouple approach for controlling the system.

[4] uses 2 different models - Kinematic Vehicle model and Dynamic vehicle model, to estimate the effect of model complexity for a coupled/decoupled longitudinal-lateral control of a vehicle using MPC.

A. ***Kinematic vehicle model***

This model assumes no slipping. Longitudinal acceleration, a only affects vehicle speed while longitudinal vehical speed also affects the lateral control. This model contains coupling between the kinematic equations and the states.

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\theta}\\ \dot{v}\end{bmatrix} = \begin{bmatrix} vcos(\theta)\\ vsin(\theta)\\ \frac{v}{D}tan(\delta)\\ a\end{bmatrix} \tag{28}$$

where the states are $z = [X, Y, \theta, v]^T$ and control inputs are $u = [\delta, a]^T$

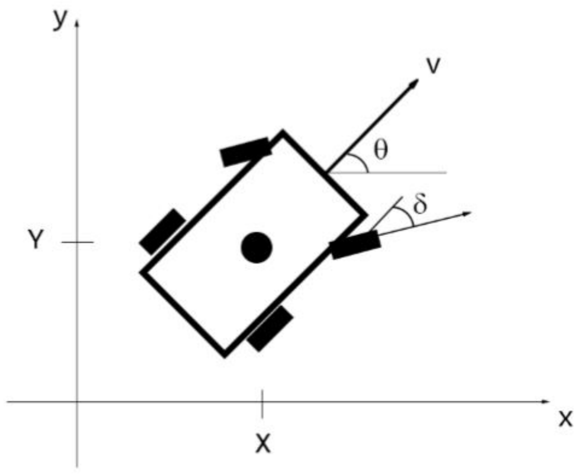

Fig. 10. Kinematic model [4]

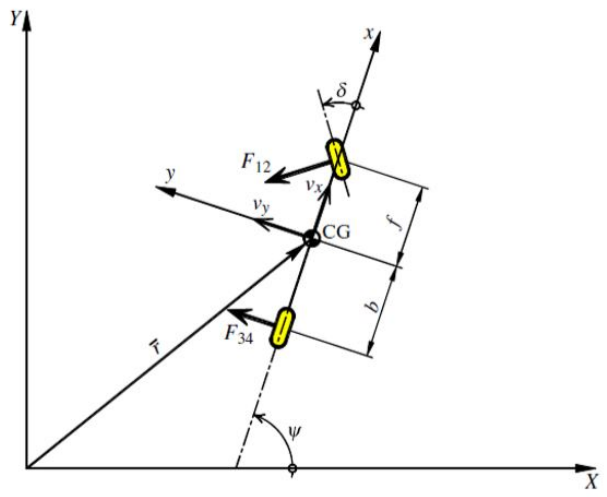

Fig. 11. Dynamical model [4]

B. Dynamic Vehicle model

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{\theta}\\ \dot{v_x}\\ \dot{v_y}\\ \dot{\phi}\end{bmatrix} = \begin{bmatrix} vcos(\theta)\\ vsin(\theta)\\ \frac{v}{D}tan(\delta)\\ \frac{-F_{12}}{m}sin(\delta)+a\dot{\phi}v_y\\ \frac{F_{34}}{m}+\frac{F_{12}}{m}cos(\delta)-\dot{\phi}v_x\\ \frac{a}{J}F_{12}cos(\delta)-\frac{b}{J}F_{34}\end{bmatrix} \tag{29}$$

$$F_{12} = -C_12\alpha_{12}, F_{34} = -C_{34}\alpha_{34} \tag{30}$$

$$\alpha_{12} = tan^{-1}(\frac{v_y+\dot{\phi}f}{v_x})-\delta, \alpha_{34} = tan^{-1}(\frac{v_y-\dot{\phi}b}{v_x}) \tag{31}$$

where the states are $z = [X, Y\theta, \dot{\phi}, v_x, v_y]^T$ and control inputs are $u = [\delta, a]$. For decoupled control, $u = \delta$, but for coupled control, $u = [\delta a]^T$.

Lateral Control tries to keep the vehicle within the path hence it can be used to follow the trajectory. The main goal of the controller is to output steering angle such that path deviations are minimal while satisfying all the constraints. On the other hand, Longitudinal Control tries to maintain vehicle speed and hence can be used for ACC. The main goal of the controller is to output vehicle acceleration. The literature uses vehicle models with coupled kinematics and dynamics and hence lateral and longitudinal dynamics may affect each other. For a decoupled control, 2 different controllers handles longitudinal and lateral controller separately.

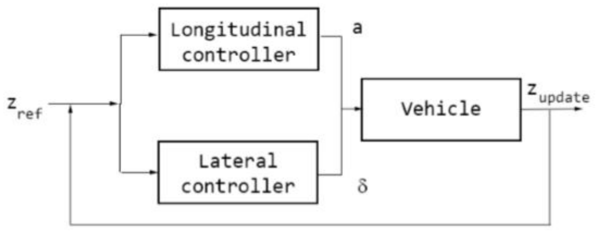

Fig. 12. Demonstration of decoupled strategy for trajectory following and ACC [4]

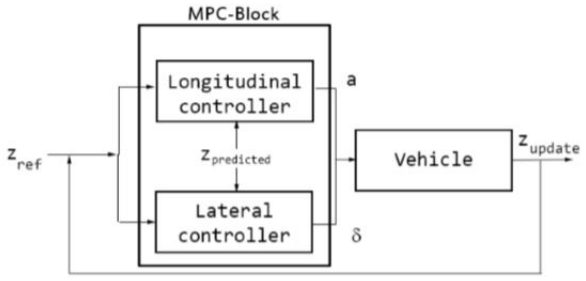

Fig. 13. Demonstration of coupled strategy for trajectory following and ACC [4]

To follow a trajectory, a time varying reference needs to be determined and this can be done by determining the waypoints as the vehicle progresses. The waypoints are separated in such a way that the vehicle can easily reach the next waypoint with its current velocity. This is directly in relation with the [4]

The reference state $z_{ref} = (z_{s0}, z_{s1}, ...z_{sHp-1}, z_{sHp},)$ and reference input

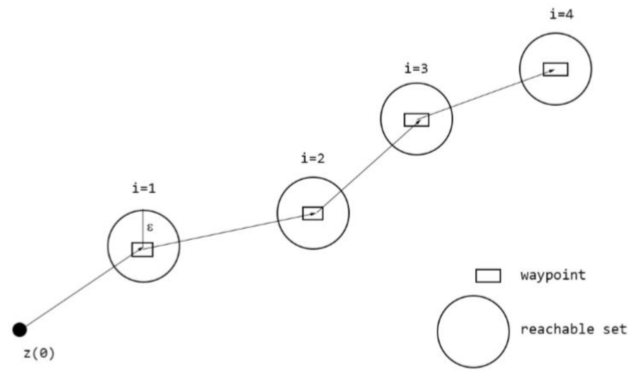

Fig. 14. Implementation of MPC for Trajectory following using waypoint strategy [4]

$u_{ref} = (u_{s0}, u_{s1}, ...u_{sHp-1}, u_{sHp},)$ are updated at each sampling time after the problem has been discretized.

The final MPC formulation thus becomes

$$\begin{aligned} \min_u \tilde{z}_{H_p}^T P \tilde{z}_{H_p} + \sum_{i=1}^{H_p-1} \tilde{z}_i^T Q \tilde{z}_i + \sum_{i=0}^{H_c} \tilde{u}_i^T R \tilde{u}_i \\ \text{subject to } z_0 = z(0) \\ \tilde{z}_i = (z_i - z_{ref_i}) \\ \tilde{u}_i = (u_i - u_{ref_i}) \\ \tilde{z}_{i+1} = (A_i \tilde{z}_i + B_i \tilde{u}_i) \\ |\tilde{z}_i| \leqslant \epsilon \\ |u_{i+1} - u_i| \leqslant \Delta u \\ |u_i| <= u_{max} \end{aligned} \tag{32}$$

Upon discretizing the system, we can rewrite the above formulation as

$$\begin{aligned} \min_u \frac{1}{2} \bar{u}^T H \bar{u} + c^T \bar{u} + d \\ \text{subject to } D\bar{u}_i \leqslant c \end{aligned} \tag{33}$$

where

$$\begin{aligned} H_i &= 2(\bar{B}_{i|i}^T \bar{Q} \bar{B}_{i|i} + \bar{R} \\ f_i &= 2\bar{B}_{i|i} \bar{Q} \bar{A}_{i|i} \bar{z_{i|i}} \\ d_i &= \bar{A}_{i|i} \bar{z_{i|i}} \bar{Q} \bar{A}_{i|i} \bar{z_{i|i}} \\ c &= \bar{B}_i^{-1}(\epsilon - \bar{A}_i \tilde{z}_i) \end{aligned} \tag{34}$$

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, c = \begin{bmatrix} a_{max} - a_{ref} \\ \delta_{max} - \delta_{ref} \\ -a_{min} + a_{ref} \\ -\delta_{min} + \delta_{ref} \end{bmatrix}, \tag{35}$$

$$\bar{A}_{i|i} = \begin{bmatrix} A_{i|i} \\ A_{i+1|i} A_{i|i} \\ \vdots \\ \alpha_{i,1,0} \end{bmatrix} \tag{36}$$

$$\bar{B}_{i|i} = \begin{bmatrix} B_{i|i} & 0 & 0 & \dots & 0 \\ A_{i|i} B_{i|i} B_{i+1|i} & 0 & \dots & 0 & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{i,2,1} B_{i|i} & \alpha_{i,2,2} B_{i+1|i} & \ddots & \dots & 0 \\ \alpha_{i,1,0} B_{i|i} & \alpha_{i,1,2} B_{i-1|i} & \dots & \dots & B_{i+H_p-1|i} \end{bmatrix} \tag{37}$$

$$\bar{z}_{i+1|i} = \begin{bmatrix} \tilde{z}_{i+1|i} \\ \tilde{z}_{i+2|i} \\ \vdots \\ \tilde{z}_{i+H_p|i} \end{bmatrix}, \bar{u}_{i+1|i} = \begin{bmatrix} \tilde{u}_{i+1|i} \\ \tilde{u}_{i+2|i} \\ \vdots \\ \tilde{u}_{i+H_p|i} \end{bmatrix} \tag{38}$$

where $\alpha_{i,j,l} = \prod_{i=l}^{N-j} A_{i+i|i}$
such that $\bar{z}_{i+1|i} = \bar{A}_i \bar{z}_{i|i} + \bar{B}_{i|i} \bar{u}_{i|i}$

The [4] uses a controller for speed profiling and path following. In the paper, reference velocity is generated which is the maximum velocity for a vehicle to avoid slipping on a free road.

$$v_{max} = \sqrt{\frac{g\mu}{\rho_r}}, \text{where } \rho_r = 1/r \tag{39}$$

where ρ_r is Road Curvature and r is radius of road curve

For any 3 points, p_{i-1}, p_i, p_{i+1},and vectors joining them as $v_1 = p_i - p_{i-1}, v_2 = p_{i+1} - p_i$. Curvature at p_i is given as

$$\rho_{r,i} = \frac{2sin(\theta/2)}{\sqrt{||v_1|| \cdot ||v_2||}} \text{ , where } \theta = cos^{-1}(\frac{v_1}{||v_1||} \cdot \frac{v_2}{||v_2||}) \tag{40}$$

However, this project deals with a cruise control instead of speed profiling. One way to perform cruise operations while following the trajectory is mentioned in [5] where control barrier functions are used to generate solutions guaranteeing forward invariance where safety specifications are expressed as set invariance. This problem is of leveraged complexity compared to the task at hand, as it tackles elements that are not exclusive to the solution of the ACC problem. Another way to incorporate cruise control is by modifying the formulation presented for Trajectory following such that the ego vehicle follows a set speed in absence of a lead vehicle but maintains a constant distance gap or time gap if a lead vehicle is present in front of it. This functionality has to be achieved while the ego vehicle follows the trajectory. Thus, extra states can be augmented for the Trajectory following case such that cruise control can be performed as described in [6]. An extra state determining the longitudinal jerk can also be added as per [7] to ensure safety of the driver.

Some research has also been done on Integrated Adaptive Cruise Control (IACC) where ACC and DYC (Direct Yaw-Moment Control) are combined. Below is the implementation for the IACC where ACC model and Lateral vehicle models are augmented.

C. *Longitudinal Dynamic Model for ACC*

$$\begin{aligned} \Delta d = d - d_{des}, \text{ where } d_{des} = \tau_h v_x + d_0 \\ \Delta v = v_p - v_x \\ j = \dot{a_x}, \text{ where } a_x = \frac{1}{\tau_{ax} s + 1} a_{x_{des}} \end{aligned} \tag{41}$$

Upon discretizaton, the system for ACC becomes

$$x_{1,k+1|k} = A_1 x_{1,k} + B_1 u_{1,k} + G_1 w_{1,k} \tag{42}$$

where $x_{1,k} = [\Delta d_k, \Delta v_k, a_{x,k}, j_k]^T, u_{1,k} = a_{x,des}, w_{1,k} = a_{p,k}$ and

$$\begin{aligned} A_1 = \begin{bmatrix} 1 & T_s & -\tau_h T_s & 0 \\ 0 & 1 & -T_s & 0 \\ 0 & 0 & 1 - T_s/T_L & 0 \\ 0 & 0 & -1/T_L & 0 \end{bmatrix}, \\ B_1 = \begin{bmatrix} 0 \\ 0 \\ \frac{T_s K_L}{T_L} \\ fracK_L T_L \end{bmatrix}, \\ G_1 = \begin{bmatrix} 0 \\ T_s \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{43}$$

D. *Lateral vehicle Modell*

$$\dot{x_2} = \Psi x_2 + \Pi u_2 + \Gamma w_2 \tag{44}$$

where $x_2 = [\beta, \omega_r]^T, u_2 = M_{z,des}, w_2 = \delta$

$$\Psi = \begin{bmatrix} -\frac{k_f + k_r}{m v_x} & \frac{b k_r - a k_f}{m v_x^2} - 1 \\ -\frac{b k_r - a k_f}{I_z} & -\frac{a^2 k_f + b^2 k_r}{I_z v_x} \end{bmatrix}, \Pi = \begin{bmatrix} 0 \\ \frac{1}{I_z} \end{bmatrix}, \Gamma = \begin{bmatrix} \frac{k_f}{m v_x} \\ \frac{a k_f}{I_z} \end{bmatrix} \tag{45}$$

While the longitudinal model does not change with v_x, Lateral model on the other hand changes with v_x and hence is time varying. This could cause a cumbersome discretization as the lateral model has to be discretized at every iteration. A Linear Parameter Varying (LPV) method can thus be applied which is as follows.

$$\begin{aligned} \Psi = \Psi_0 + \Psi_1 p_1 + \Psi_2 p_2 = \Psi_0 + \sum_{i=1}^{2} \Psi_i P(i) \\ \Psi_0 = \begin{bmatrix} 0 & -1 \\ \frac{b k_r - a k_f}{I_z} & -\frac{a^2 k_f + b^2 k_r}{I_z v_x} \end{bmatrix}, \\ \Psi_1 = \begin{bmatrix} -\frac{k_+ k_r}{m} & 0 \\ 0 & -\frac{a^2 k_f + b^2 k_r}{I_z v_x} \end{bmatrix}, \\ \Psi_2 = \begin{bmatrix} 0 & \frac{b k_r - a k_f}{m} \\ 0 & 0 \end{bmatrix}, \end{aligned} \tag{46}$$

where $p_1 = \frac{1}{v_x}, p_2 = \frac{1}{v_x^2}, P = [p1, p2], P_1 = [p_{1,max}, p_{1,max}], P_2 = [p_{1,min}, p_{1,min}]$

Local linearisation of Ψ can be computed as $\Psi_{b,j} = \Psi_0 + \sum_{i=1}^{2} \Psi_i P(j), j = 1, 2$
Upon discretization, we get

$$\begin{aligned} A_{b,j} = \sum_{i=0}^{\infty} \frac{\Psi_{b,j}^i T_s^i}{i!} \\ B_{b,j} = \sum_{i=1}^{\infty} \frac{\Psi_{b,j}^{i-1} T_s^i}{i!} \Pi, j = 1, 2 \\ G_{b,j} = \sum_{i=1}^{\infty} \frac{\Psi_{b,j}^i T_s^i}{i!} \Gamma, j = 1, 2 \end{aligned} \tag{47}$$

and thus $x_{2,k+1|k} = A_2 x_{2,k} + B_2 u_{2,k} + G_2 w_{2,k}$ and $A_2 = \sum_{j=1}^{2} w_{b,j} A_{b,j}, B_2 = \sum_{j=1}^{2} w_{b,j} B_{b,j}, G_2 = \sum_{j=1}^{2} w_{b,j} G_{b,j}, \sum_{j=1}^{2} w_{b,j} = 1, w_{b,j} > 0$

We can then use A_1, A_2 to form A and similarly B, and G also.

$$\begin{aligned} A = \begin{bmatrix} A_1 & \\ & A_2 \end{bmatrix}, \\ B = \begin{bmatrix} B_1 & \\ & B_2 \end{bmatrix}, \\ G = \begin{bmatrix} G_1 & \\ & G_2 \end{bmatrix}, \end{aligned} \tag{48}$$

IX. Appendix

See the next page for attached code.

References

[1] G. Orosz and C. He, "A matlab dataset of the Google Earth Data for the Circuit of Americas," 2018, this data was obtained from the course ROB535-Self Driving Cars at the University of Michigan Ann Abor.

[2] G. Liu, H. Ren, S. Chen, and W. Wenzhu, "The 3-dof bicycle model with the simplified piecewise linear tire model," 12 2013, pp. 3530–3534.

[3] (2019) Fmincon documentation for Matlab R2019b. [Online]. Available: https://www.mathworks.com/help/optim/ug/fmincon.html

[4] C. Olsson, "Model complexity and coupling of longitudinal and lateral control in autonomous vehicles using model predictive control," 2015.

[5] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1216–1229, 2017.

[6] A. Idriz, "Safe interaction between lateral and longitudinal adaptive cruise control in autonomous vehicles," 2015.

[7] J. Zhang, Q. Li, and D. Chen, "Integrated adaptive cruise control with weight coefficient self-tuning strategy," *Applied Sciences*, vol. 8, no. 6, p. 978, 2018.