Master thesis

Optimal Control and Planning for Autonomous Driving

Tarek BOUAMER

Advisors

Dr .Son Tong

September 2018

Abstract

Safety is an emerging tasks in the field of Self-driving Cars that includes Perception, Planning and Decision Making fields to improve the autonomy in all driving conditions especially in the urban driving where the cars shares the environment with other vehicles and pedestrians. Several research activities has been done and some promising results were achieved.

In this master thesis, we have focused on Trajectory planning and Execution task that enables our Amesim Car to overtake safely around predefined environment attempting to reduce the error between the planning and execution. A kinodynamic motion planner like-driver was developed to mimic the human driver actions and to provide us with an executable path.

In addition, An optimal trajectory controller was designed to stabilize the vehicle and track perfectly the reference under system constraints. A simple and complete model of Amesim car was identified to be used in both path planning and controller design.

Finally, a co-simulation is carried for different scenarios: double lane change test and racing track with different control schemes

Acknowledgement

I want to thank Dr. Son Tong for his guidance and giving me the opportunity to work in this interesting project. I also want to thank members of RTD LMS Siemens Industry Software for their technical support.

Many thanks to Paris Saclay Univesity, Evry Val d' Essonne university and Erasmus for providing me with funding during my master thesis.

I would like to thank my dear professors supervisors insightful comments and motivation.

Lastly, I want to thank my family who has been supportive and helping me through my education.

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

By 2013, Self-driving Cars or fully Autonomous cars have gone from science fiction to reality, where many companies start working on their own self-driving car technologies but the path to self-driving vehicles has taken longer than that.

Just after the birth of the cars in 1925, inventors start thinking to make the vehicle autonomous. Computer-Controller car essay was published in 1968 by John McCarthy, where the author introduced the concept of “automatic chauffeur,” that is capable of navigating in a public road via a visual input [1] . This idea paved to Discmans Ernst and his team in 1980 to developed an automatic visual motion control vehicle guidance with artificial intelligence (AI) to navigate along highways in high speeds [2] [3].

In last three decades , an increasing effort from both academia and industry has been done toward Self driving cars due to the technological advancement in Sensing and Computational areas together with social benefit (human negligence and traffic fatalities …) where Self-Driving cars contribute effectively in reducing the human causes of vehicle accident and better transit management in highways, parking and intersections.

The famous project NAVLAB by CMU demonstrated a further advances in perception capability and environment understanding by Pomerleau [4]; where the author trained an adaptive visual module using artificial neural network to classify images into “drivable road” and “non-drivable road” regions as the task of defining the road is not easy in some conditions (surface material, lighting, ...).

First autonomous driving competitions have been held by DARPA (Defense Advanced Research Projects Agency) in 2004, to navigate 142 miles through the Mojave Desert within as quick as possible in limited time. Since then; DARPA spurs the development and technologies needed to create the first fully autonomous car. DARPA URBAN challenge held in 2007 demonstrates the possibility to drive fully autonomous in city Traffics [5]. In the same period; self-parking systems start to get attention toward developing an automatic parallel parking assistance system.

In 2009, google lunches her Waymo’s self-driving car project and after 5 years of testing and data collection they have revealed their first fully autonomous car without steering wheel, throttle or brake pedal [6];

Sadly, the first autonomous car accident occurs in USA where Tesla’s Autopilot self- fails to activate emergency brake at the right moment an hits the front trailer. The second crash was in last march 2018, where an Self Driving Uber car crashes a pedestrian woman even though the perceiving system was able to detect her 6 seconds before and no action was taken. These two incidents renewed a debate about safety level of self-driving cars .

The cars driving systems has jumped from human driving level to semi-autonomous level then to fully autonomous where five (5) different levels of driving automation can be distinguished:

- Level Zero (0): *Zero autonomy*; the driver performs all driving tasks.

- Level One (1) : *Driver Assistance*; the vehicle is still controlled by the driver, with some driving assist in accelerating and braking.

- Level Two (2) : *Partial Automation*; now the vehicle assists the acceleration and steering, but the driver still can disengage these actions in critical situations.

- Level Three (3) : *Conditional Automation*; the vehicle monitor the environment based on sensors reading and controls itself under certain conditions where the driver still can disengage the vehicle decision when it is needed .

- Level Four (4) : *High Automation*; The vehicle is capable of autonomously driving under certain conditions like Lane change and some safety priorities if the driver fails to make a right decision.

- Level Five (5) : *Full Automation;* The vehicle is capable of performing all driving modes under all conditions.

In this thesis, we focus on the control and motion planning aspects that falls on the third (3) autonomy level assuming that we have complete information about the vehicle surrounding.

1.2 Scope

The 3rd autonomy level of self-driving cars includes two main modules (1) *Perception* module, that enables the car to localize its self and understand its surrounding based on onboard sensors LIDARs and cameras, and a (2) Decision Making module, that defines the behavior of the car depending on the collected data from the perception module where the most important driving behaviors are (a) lane change, (b) lane following (c) valet parking and (d) obeying traffic laws (priorities and traffic light).

Once the motion specifications is defined, a *motion planning* algorithm is needed to navigate in the environment to determine a feasible trajectory then a *feedback control* system determines the appropriate control inputs to correct the trajectory following error to increase the overall safety of the system. In Figure 1. 1; block diagram shows the hierarchy levels of the Decision Making Module.

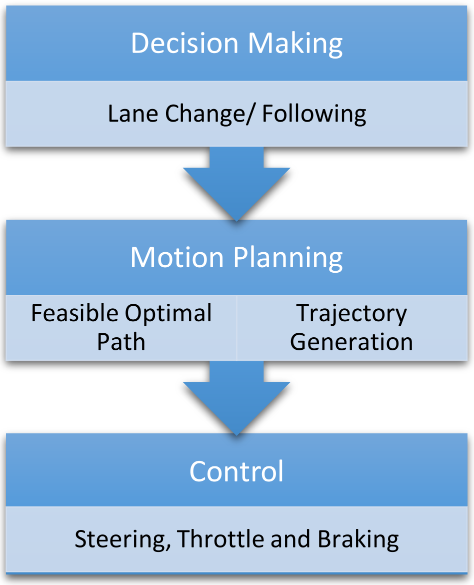


Figure 1. 1: Hierarchy of decision-making processes, motion planning and feedback control

1.3 State of the art

Different motion planning algorithms have been proposed for Self-Driving cars to compute a safer and feasible trajectory from initial to a goal configuration, where they have been initially developed for mobile robots applications .

graph-search approaches are well known path planning methods that perform a global search strategy in a discretized graphical space to find the shortest drivable path in roadmap. Dijkstra Algorithm [7] has been used as local mission planner to find new sequence of waypoints to adapt with traffic conditions. Dijkstra algorithm is considered as vast and slow path planner, for better searching performance; A* Algorithm was introduced with heuristic function which gives priority to nodes that are supposed to be better than others. In autonomous driving, The A* was used mainly in parallel parking application with Voronoj based cost function as in [8]. The main drawback of A* is that the resulting discreet path cannot be executed by the vehicle, a continuous version of A* or hybrid A* was introduced in [9] as part of DARPA challenge for parking lots and also for specific maneuvers like hard turn left (U-turns) where the vehicle is represented by its location, orientation and motion direction (forward/backward), the key element in hybrid A* is to find the heuristic rules.

Graph search algorithms are appropriate whenever a map is available and roadmap is predetermined, unfortunately; in autonomous driving, the use of graph search algorithms is limited only on parking application and route planner. This explains the popularity of *sampling-based* motion planner in self-driving cars. These methods explore the reachability by randomly sample in high dimensional state space configuration under constrains.

In [10], the authors introduces a suboptimal versions of the *Probabilistic Roadmaps* (PRM) and *Rapidly-exploring Random Trees* (RRT). The PRM approach is quite simple and general where each time a new sample node is determined in free space and connected to its neighboring with straight line (edge) as described in [11].

Rapidly-Exploring Random Tree (RRT) is achieved by picking a random sample from the free configuration space and extending the tree toward it [12]. The algorithm is fast and suitable for online planning however there are no guarantees that the resulting path is exactable (no continuous curvature) [13] .

In [14] RRT* was introduced, the authors address this problem and they have integrated an exact steering procedures within standard RRT to almost ensure the possibility of the path execution compared to the RRT.

In order to execute the generated trajectory; a feedback control system is required to stabilize the vehicle and correct the tracking errors due to the vehicle model uncertainties and other external disturbances (slipper road…).

For low speeds application; a simple kinematic model of the vehicle can be used to design controller where different control schemes have been deployed: Proportional Integral Derivative (PID) [15], feedback linearization [16] and Model Predictive controllers [17], However in high speeds and hard maneuvers; a full dynamic model of the vehicle is used to design stable and robust controller where in [18],and [19] the authors designed full nonlinear model predictive controller based on vehicle dynamics and tire model which shows better tracking performance compared to [17]. a linear MPC was also discussed in [20] less computationally expensive and it shows a satisfactory trajectory tracking. The paper [21] presents a feedback-feedforward control scheme; it maintains the vehicle stability and improves the tracking performance by reducing lateral deviation. A perfect tracking performance has been achieved by MPC and this explains the growing attention in automotive industry in last past year.

1.4 Thesis Structure

This master thesis text is structured as follows:

- In Chapter 2, the simulation platform is as the software used to implements and test the control and planning algorithms

- In Chapter 3, the vehicle model is derived and parameter identification is performed.

- In Chapter 4, a motion planner algorithm is describer and instigated for lane change scenario.

- In Chapter 5, Simulation result for different control approaches.

- In Chapter 6, a conclusion and future work are given.

Chapter 2

Simulation Platform

Siemens Industry Software (SISW) is a business unit of Siemens Digital Factory Division. The company provides software, services and systems in the areas of managing the product lifecycle and management of industrial operations. SISW works collaboratively with clients to offer industrial software solutions that help companies worldwide to achieve a sustainable competitive advantage by realizing, making real their important innovations.

2.1 Simulation with LMS Imagine.Lab Amesim

LMS Imagine.Lab Amesim software in Siemens Industry Software NV is a Multiphysics simulation platform that provides libraries of different physical domain, such as mechanical, electromechanical and powertrain.

In this project, it will be used to simulate a vehicle and its dynamic for the purposes of testing ADAS control algorithms. By connecting blocks, a complex and realistic model can easily be built for simulations. Furthermore, a very detailed sub-model of single components can be made.

Amesim car is a high fidelity with 15 degree of freedoms (15DOf), the model that includes tire and suspension dynamics with more than 72 state variables which make her pretty close to the real cars. Hence, it is very useful to test and validate the planning and control approaches. The car is illustrated in Figure 2. 1 and more details about the car can be found in [22].

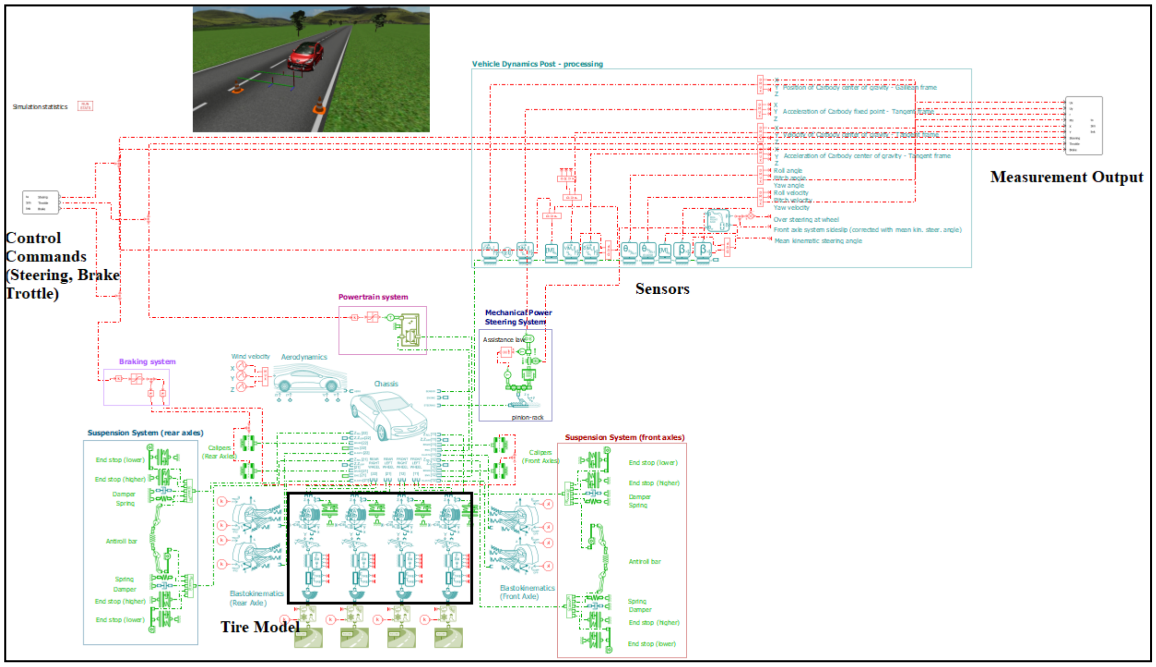

Control Commands (Steering, Brake Trottle)
Measurement Output
Sensors
Tire Model
Vehicle Dynamics Post - processing
Powertrain system
Mechanical Power Steering System
Braking system
Aerodynamics
Chassis
Suspension System (rear axles)
Suspension System (front axles)
Elastokinematics (Rear Axle)
Elastokinematics (Front Axle)

Figure 2. 1: Amesim 15 DOF dynamic Model

2.2 Matlab/Simulink-Amesim Co-simulation

In this project, we are going to use a co-simulation between AMEsim software that contains the car dynamics and Matlab/Simulink. mainly, the control system and motion planning algorithm are developed in Matlab and then a real time simulation in performed in Simulink

Chapter 3

Vehicle Model

To Successfully design a controller, a mathematical model which represents the 15Dof Amesim Car has to be derived and matches enough the behavior of the Amesim car with less expensive computations to reduce the time it takes to solve the optimization problem.

The bicycle model; most used model; is adopted in our work to model Amesim Car; it includes two main parts: (1) Mechanical Model and (2) Tire Model. It simplifies the model geometrically to two wheels front and rear wheel at the center axis under symmetrical assumption.

The next step, we are going to derive the bicycle model of the Amesim car and show the resulting analysis.

3.1 Bicycle model

The used Amesim Car is Front Wheel Drive (FWD) where only the front wheels are actuated in the longitudinal direction. In the other hand; the back wheels are assumed to roll freely. we represent the engine actuation on the front wheels as longitudinal forces. The Car changes its orientation with front wheel steering input only.

By neglecting the car Torsion (normal tire load $F_{z_f}, F_{z_r} = constant$) and assuming that the car is symmetrical about its Centre of Gravity (CG) axle (Front width and Rear width), The plane motion can be described in a body fixed frame placed in the Centre of Gravity (CG) including the lateral (x) and longitudinal (y) dynamics as well the body orientation (yaw) and its rate described in [23] (see Figure 3. 1).

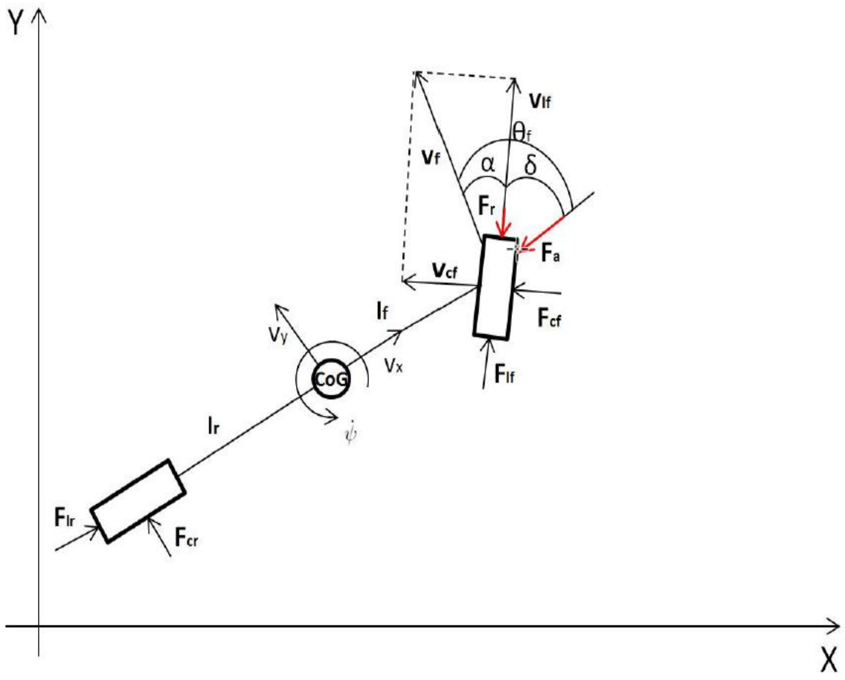


Figure 3. 1: Bicycle model of car.

V_{x_f},V_{y_f} are the longitudinal and lateral velocity vectors for the front wheel in the tire frame . r represents the yaw rate and δ is the input steering angle. The forces F_l and F_c are the longitudinal and lateral (cornering) wheel forces. The subscripts { r, f } stand for rear and front components.

Using Newton's and Euler's law, we can write the dynamics of the car in terms of the velocities in body frame :

$$m\,\dot{U}_x = F_{x_f} + F_{x_r} + m\,r\,U_y \tag{3.1}$$

$$m\,\dot{U}_y = F_{y_f} + F_{y_r} - m\,r\,U_x \tag{3.2}$$

$$I_z\ \dot{r} = L_f\,F_{y_f} - L_r\,F_{y_r} \tag{3.3}$$

where, m is the vehicle mass, I_z the vehicle inertia around the z-axis and L_f is a distance between Center of Gravity (CG) and the front axle, L_r is the distance between (CG) and the rear axle. F_x, F_y are the longitudinal and lateral forces acting on the car respectively. U_x, U_y denote longitudinal and lateral velocities of the car, respectively.

The position and orientation (kinematics) of the car in absolute initial frame can be described using dynamics of the car to get a complete set of equation of the car:

$$\dot{X} = U_x \cos\psi - U_y \sin\psi \tag{3.4}$$
$$\dot{Y} = U_x \sin\psi + U_y \cos\psi \tag{3.5}$$
$$\dot{\psi} = \frac{U_x}{\mathrm{L}_f + \mathrm{L}_r} \tan\delta \tag{3.6}$$

The forces acting (F_x, F_y) on the car can be expressed in terms of the longitudinal and lateral tire forces F_l and F_c :

$$F_x = F_l \cos\delta - F_c \sin\delta \tag{3.7}$$
$$F_y = F_l \sin\delta + F_c \cos\delta \tag{3.8}$$

Where the tire forces F_l and F_c can be expressed as [23]:

$$F_l = f_l(\alpha, s, \mu, F_z) \tag{3.9}$$
$$F_c = f_c(\alpha, s, \mu, F_z) \tag{3.10}$$

Where α is slip angle, s is the slip ratio, μ the friction coefficient for the road and F_z is the vertical tire load acting on the wheels. The tire forces are highly nonlinear and under the following assumptions are simplified to a simple nonlinear equations:

- Constant nominal loads on the front and rear wheels F_{z_f}, F_{z_r} .
- s constant satisfying $v = R\ \omega$. (v body velocity, ω wheel velocity R wheel radius).
- Neglect the friction coefficient which is considered as external disturbance.

These simple nonlinear equations are calculated using Pacejka Magic Formula [24]. The slip angle of the front and rear wheels is given by:

$$\alpha_f = -\arctan(\frac{r\,L_f + U_y}{U_x}) + \delta \tag{3.11}$$

$$\alpha_r = \arctan(\frac{r\,L_r - U_y}{U_x}) \tag{3.12}$$

Considering again the constant slip ratio $v = R\,\omega \;=>\; s = 0$. the longitudinal tire forces $F_{l_{f,r}}$ in equations (3.9 – 3.10) are assumed also to be zero since they are proportional to the slip ratio. Thus The forces acting on the car (F_x, F_y) can be expressed in terms of the lateral tire forces F_c only.

3.2 Magic Formula

The magic formula f_c is an empirical formula used to calculate the tire forces, the trigonometric form of the magic formula is given as (Amesim Tire library [22]):

$$f_c = \mathcal{D} \sin\left(\mathcal{C} \operatorname{artan}(\mathcal{B}\,\sigma)\right) + \mathcal{S}_v \tag{3.13}$$

$$\sigma = (1 - \mathcal{E}) * (\alpha + \mathcal{S}_h) + \left(\frac{\mathcal{E}}{\mathcal{B}}\right) \operatorname{artan}(\mathcal{B}\,(\alpha + \mathcal{S}_h)) \tag{3.14}$$

- s$\mathcal{B}$ the stiffness factor.
- $\mathcal{C}$ the shape factor.
- $\mathcal{D}$ the peak factor.
- $\mathcal{E}$ the curvature factor.
- $\mathcal{S}_h$ the horizontal shift.
- $\mathcal{S}_v$ the vertical shift.

In the Amesim car, a complex magic formula is used to calculate the Tire forces, where the magic formula coefficients are in function of the nominal Tire force F_z, also nominal Tire force is not constant and changes in function of the input steering angle δ and the longitudinal and lateral velocities (V_{x_f},V_{y_f}) producing a pitch angle.

Solving the optimization problem that includes many explicit algebraic and differential equations will be time consuming, if not impossible. Thus a simplified common Magic Formula will be used instead and experimental identification will be performed to describe the lateral tire forces that matches the ones of the real car:

$$f_{c_i} = F_{z_i}\, \mathrm{D}_i \sin\left(\mathrm{C}_i \,\mathrm{artan}\,(\mathrm{B}_i\, \alpha_i)\right) \qquad i{:}\{f, r\} \tag{3.15}$$

The simplified formula uses only the slip angle α to calculate the cornering forces of the front and rear wheels, The nominal forces F_{z_f}, F_{z_r} are given by:

$$F_{z_f} = \frac{m\, g\, L_r}{(L_f + L_r)} \tag{3.16}$$

$$F_{z_r} = \frac{m\, g\, L_f}{(L_f + L_r)} \tag{3.17}$$

The parameters $\mathrm{B}_i, \mathrm{C}_i$ and D_i will identified experimentally and the Figure 3. 2 shows the lateral forces with different B values.

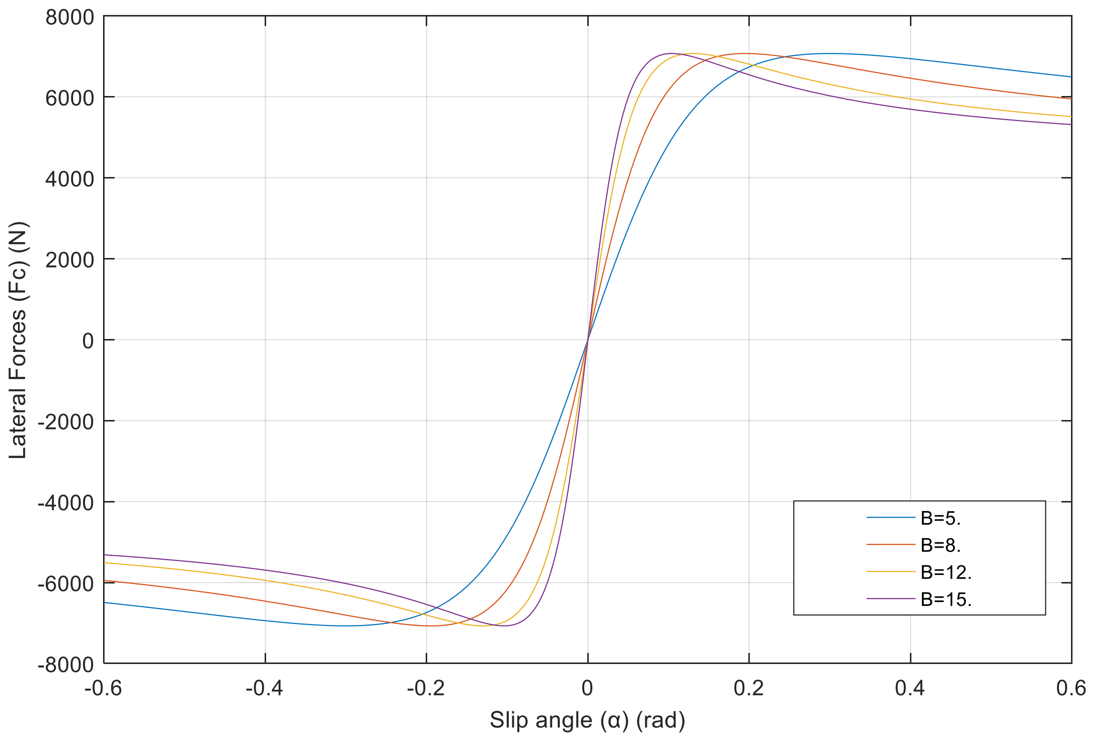


Figure 3. 2: Lateral tire force with different Pacejka B values

3.3 External forces

The Amesim Car; as any car; is also affected by different external forces, mainly the rolling resistance force F_{res} and the drag force F_{drag}. By definition the rolling resistance force is the force resisting the motion when a wheels (vehicle) rolls on a surface and it can be expressed as:

$$F_{res} = D_{res}\, m\, g \cos(\theta) \tag{3.18}$$

where D_{res} is the roll resistance coefficient and θ is the angle between the road and the horizontal. For a plane motion in flat surface $\theta = 0$. the drag force is aerodynamic force that opposes the vehicle's motion through the air and according to Rajamani [25]is given by :

$$F_{drag} = \frac{1}{2}\rho\; A_F\, C_D\, (V_x + V_{wind})^2 \tag{3.19}$$

where ρ is the density of air, C_D is the aerodynamic drag coefficient and A_F is approximately 80 % of the total area of the front of the car. V_x is the longitudinal velocity and V_{wind} the wind velocity, the wind velocity is not deterministic and small compared to the longitudinal velocity, so we consider only the vehicle velocity to compute the drag force in our model.

Another important forces and the main one is the input force that makes the car accelerate (throttle) or decelerate (Brakes), obviously the two forces has different modeling in acting on the acceleration of the car; so unique linear model will be adopted for control design and logic mapping will used to map the acceleration into input throttle in case of acceleration or input brake in deceleration action:

$$F_{in} = T_{max}\, a \tag{3.20}$$

Now, the total force acting on the car on the longitudinal direction :

$$F = F_{in} + F_{drag} + F_{res} \tag{3.21}$$

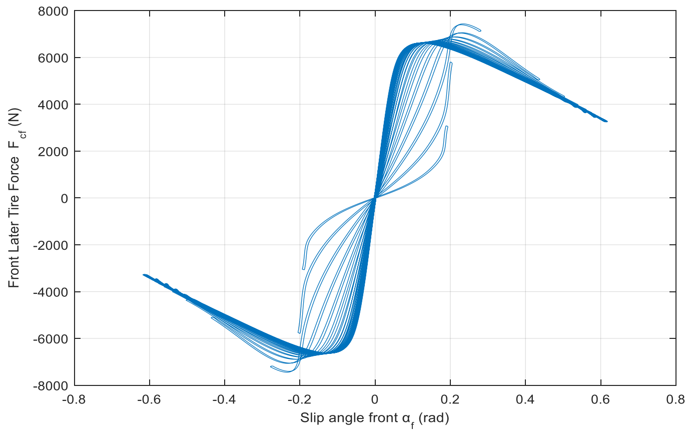
3.4 Identification

In this section, the magic formula parameters (B C D) will be modeled in steady states as follows:

- ❖ We have simulated the car 18 times for different speed profiles from 10 km/h to 100km/h, a PI controller has been used to regulate the speed at the desired one.

- ❖ For each simulation, A ramp input steering of 0.1 rad/s was introduced to the car in 3 phases: from 0 to δ_{max} then from δ_{max} to $-\delta_{max}$ and then to 0 again.

- ❖ The lateral tire forces of the four wheels and the slip angles has been recorded (N =17 347 point). We have combined the lateral force of each pairs { f, r} to fit the bicycle model. The collected data are plotted in Figure 3. 3.



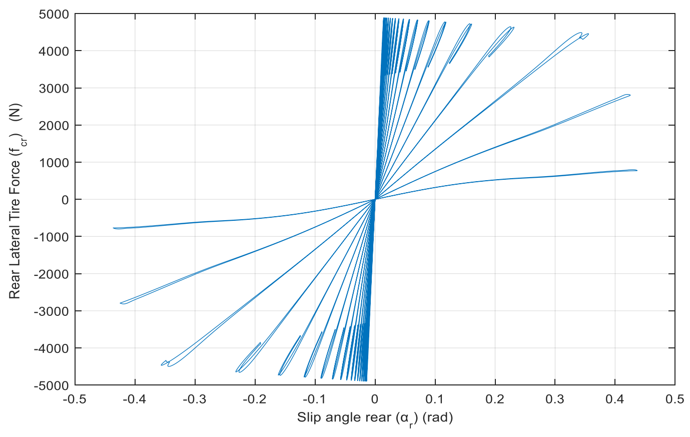


Figure 3. 3: Measured Lateral tire forces in (a) Front and (b) Rear Wheels

Using least square identification, the magic formula parameter has been identified:

$$\min_{B_i,C_i,D_i} \sum_{k=1}^{N} \left| F_{c_i}(\alpha_{i_k}) - \hat{F}_{c_i}(\alpha_{i_k}) \right|^2$$

$$\hat{F}_{c_i}(\alpha_{i_k}) = F_{z_i} D_i \sin(C_i \arctan(B_i \, \alpha_i)) \tag{3.22}$$

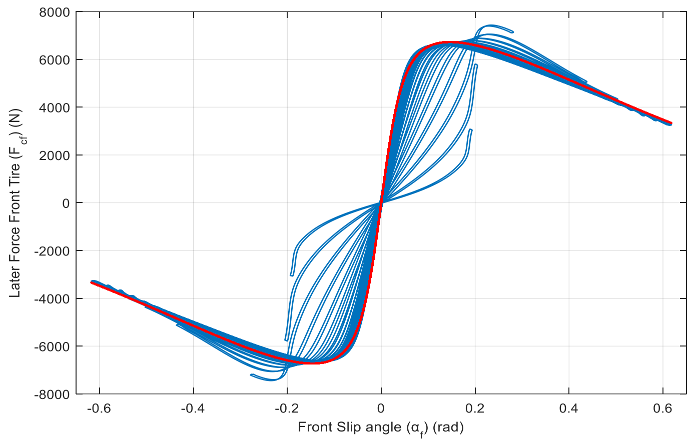
$$B_i \geq 0$$

$$C_i \geq 0$$

$$D_i \geq 0$$

Solving this Least Square problem gives the following results as plotted in Figure 3. 4:

B_f= 11.01	C_f =1.569	D_f= 1.017
B_r= 50.17	C_r =1.268	D_r= 0.6057



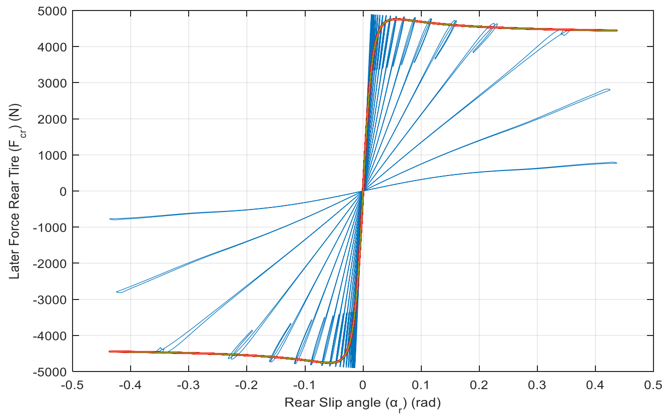


Figure 3. 4: Fitted magic formula.

to determine the resistance force F_{res}; we let the car roll freely with different initial velocities (10 km/h to 100km/h) and we recorded the longitudinal forces applied in all 4 wheels and $D_{res} = 0.001357$.

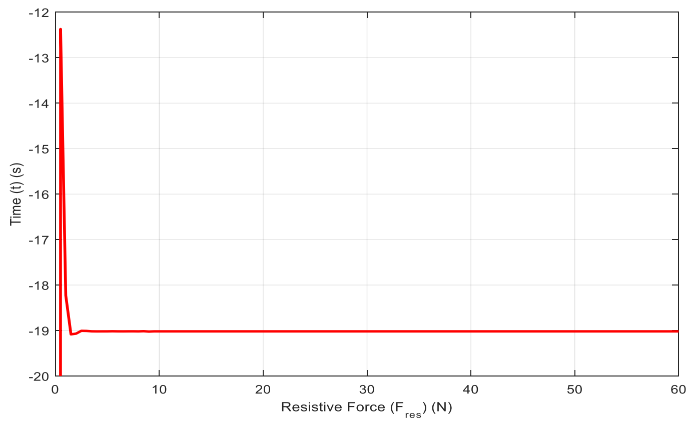


Figure 3. 5: Steady states Rolling resistance.

Finally, we put the equations together and the bicycle model of the car can be written as follows and Table summarize the identified parameters :

$$\begin{aligned} \alpha_f &= -\arctan(\frac{r\,L_f + U_y}{U_x}) + \delta \\ \alpha_r &= \arctan(\frac{r\,L_r - U_y}{U_x}) \end{aligned} \tag{3.22}$$

$$\begin{aligned} F_{c_f} &= F_{z_f}\, \mathrm{D}_f \sin\left(\mathrm{C}_f \,\mathrm{artan}\left(\mathrm{B}_f\, \alpha_f \right)\right) \\ F_{c_r} &= F_{z_r}\, \mathrm{D}_r \sin\left(\mathrm{C}_r \,\mathrm{artan}\left(\mathrm{B}_r\, \alpha_r \right)\right) \end{aligned} \tag{3.23}$$

$$\begin{aligned} \dot{U}_x &= \left(\frac{1}{m}\right)(F_{in}\cos\delta - F_{drag} - F_{res} - F_{c_f}\sin\delta + m\,r\,U_y) \\ \dot{U}_y &= \left(\frac{1}{m}\right)(F_{in}\sin\delta + F_{c_f}\cos\delta + F_{c_r} - m\,r\,U_x) \\ \dot{r} &= \left(\frac{1}{I_z}\right)\left(L_f\,(F_{c_f}\cos\delta + F_{in}\sin\delta) - L_r\,F_{c_r}\right) \end{aligned} \tag{3.24}$$

$$\begin{aligned} \dot{X} &= U_x\cos\psi - U_y\sin\psi \\ \dot{Y} &= U_x\sin\psi + U_y\cos\psi \\ \dot{\psi} &= \left(\frac{U_x}{\mathrm{L}_f + \mathrm{L}_r}\right)\tan\delta \end{aligned} \tag{3.25}$$

Table 3. 1: Amesim Car Identified parameters.

Parameter	**Value**	**[Unit]**
m	1430	kg
I_z	1300	kg^2
L_f	1.056	m
L_r	1.344	m
D_f	0.6057	
C_f	1.569	
B_f	11.01	
D_r	0.6057	
C_r	1.268	
B_r	50.17	
D_{res}	0.001357	
C_D	0.44	
T_{max}	3000	N

Chapter 4

Path planner like driver

Safety is one of the critical tasks for self-driving cars in a shared environment with other dynamic systems (vehicles, pedestrians,…) which has been tackled from different levels. In this part, our main concern is to develop a motion planner algorithm to allow the autonomous car to move from initial state configuration to a given final state configuration under environmental constraints (pedestrians, other cars, road borders …) seen as physical constraints.

Different algorithms has been investigated to develop a global optimal motion planner satisfying the following key properties :

- *The car model*: a model, as in control level, is required in finding a safe path, the more the adopted model matches the kinematics and dynamics of the car , the more the path is safer and trackable. Different models has been used to model the car mainly Car-like robot, Bicycle model…etc.
- *Completeness:* not all path planner algorithm succeed to reach the goal configuration within critical urban configuration (parking)
- *Optimality:* besides to the dynamic and kinematic constraints, optimal motion planners are enforced to find a path that solve one of the following common problem in Self-Driving : short distance, minimum time or minimum curvature (smoothness).
- *Time Complexity:* finding a path is time consuming and the main challenge is to be able to predict or bound the time required in finding a feasible path.

In this thesis, our main concern is safety in other wors a precision and completeness are having priority in our choices. To do so; we have used a high fidelity model (kinematics + dynamic) in finding path. It ensures that dynamic constraints are satisfied whenever it is feasible and trackable by trajectory controller where the tracking error will be comparatively small. To avoid the non-complete path, searching algorithms can be a best choice for its anytime and exploring properties.

To satisfy the above requirement, Path planner like-driver is developed based on Kinodynamic-RRT planning; known also as Trajectory design; for its (a) Rapid exploration capabilities, (b) high precision in replacing the standard space configuration by the kinematic and dynamics of the vehicle, and (c) coupling the path planning with trajectory generation so no transformation is needed.

The major now task is to determine the appropriate *control inputs set* to drive the vehicle from its initial states to a final states within an environment (scenario) which can be also seen as feasible trajectory finder in open loop structure considering the physical limitations of the control set (maximum steering, throttle) . Moreover , any point in the free space has geometrical and dynamical configuration. The use of Rapid Exploring Randomized Tree (RRT) moves the problem from trying to find optimal path to attempting to get a feasible path that is *good enough* as long it satisfies all constraints.

the kinodynamic problem is formulated as follows :

- $\mathcal{C}$ denotes the configuration space that describes the vehicle in the road (environment) where $q \in \mathcal{C}$ represents the geometrical transformation applied on vehicle to move from its initial state. Whereas $\mathcal{X}$ denotes the state space (car + environment) in which the state $x \in \mathcal{X}$ is defined by $x \in (q, \dot{q})$.
- The non-holonomic constraints in $\mathcal{X}$ are expressed in as $\dot{x} = f(x, u)$ where f defines the system dynamics and $u \in \mathcal{U}$ represents the set of admissible control inputs to our vehicle traveling from initial state x_{int} toward goal state x_{goal}.

- Assuming that a perceiving system is capable to detect and model the road with obstacles $\mathcal{C}_{osbt}$ (collision) present in sensors visibility range, we consider the neighboring as free collision space $\mathcal{C}_{free} = \mathcal{C}/\mathcal{C}_{obst}$. However planning in state space $\mathcal{X}$ is different, $\mathcal{X}_{free} = \mathcal{X}/\mathcal{X}_{ric}$ where $\mathcal{X}_{ric}$; known as the region of inevitable collision; defines the region where the vehicle is in collision with obstacle or in *nothing can be done* situation. it means if $q \in \mathcal{C}_{obst}$ implies that $x \in \mathcal{X}_{obst}$ since $x \in (q, \dot{q})$ and $\mathcal{X}_{obst}$ is only a subset of $\mathcal{X}_{ric}$ ($\mathcal{X}_{obst} \subseteq \mathcal{X}_{ric}$). The Figure 4. 1 shows different configuration and the difference between $\mathcal{X}_{ric}$ and $\mathcal{X}_{obst}$: our car is the blue one where the red one in the front seen as obstacle. The front car stopped abruptly and now we can see the change of $\mathcal{X}_{ric}$ for different initial speed profile.

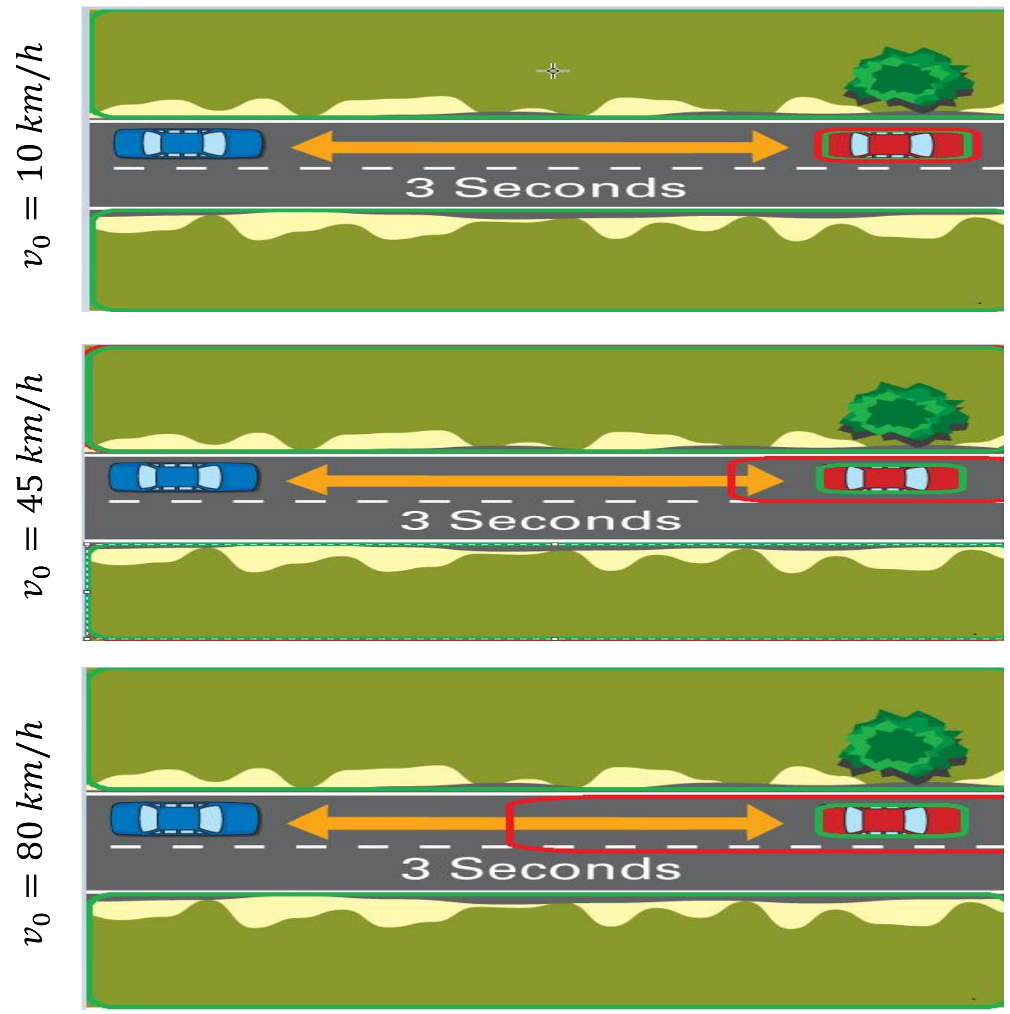


Figure 4. 1: Difference between Free State Space $\mathcal{X}_{free}$ and Configuration Space $\mathcal{C}_{free}$.

- The solution to this problem is to find a set of input $u(\tau)$: $[0, T] \in \mathcal{U}$ which results a free collision trajectory sample $x(\tau)$: $[0, T] \in \mathcal{X}_{free}$ in driving the car from x_{int} toward goal state x_{goal} in bounded sample time T. We will use Amesim car model as high fidelity motion planner and an iterative simulation technique will be performed to sample over $\mathcal{X}_{free}$ .the following Algorithm describe the proposed Kinodynamic motion planner over specific scenario:

Algorithm: Kinodynamic Motion planner

Input: x_{int}, x_{goal}, *scenario*

Output: *time parametrized Path { τ , u(τ) , x(τ) }*

```
1:   Repeat
2:       Define the Map;
3:       Locate Obstacles in the Map;
4:       Get Initial States of Vehicle;
5:       While ( forever ) do
6:           u = Get Control Input ( scenario );
7:           x(τ) = Simulate Amesim Mode ( u, T );
8:           If Collision Free( x(τ) ) then
9:               Break;
10:          end If
11:
12:      end while
13:  Until Scenario is False
```

Lane change, lane following, overtaking and others are referred in Kinodynamic Motion planner algorithm as scenarios, for each scenario we have parametrized *logically* the control actions {Steer, Brake, Throttle} that should be taken to perform the task set by the decision making process. A randomized sampling process is used to set *numerically* the logical control action, for example; the logical action in overtaking the front car is to steer to the *left* and the numerical value that defines how much should I steer is set randomly following a distribution function $Steer_{left}(z) = F(Z \leq z)$.

Therefore, Rapidly Exploring Random Trees (RRT) is used to handle the randomized sampling process within Kinodynamic Motion Planner as quick as possible and uniform structure.

Algorithm: RRT Algorithm

Input: x_{int}

Output: Status

```
1:     $V \leftarrow \{ x_{int} \}$ ; $E \leftarrow \{ \emptyset \}$; $D \leftarrow \{ \emptyset \}$; $i = 0$;
2:     While i < max_iterations do
3:         $G \leftarrow \{ V, E \}$
4:         $x_{rand} \leftarrow Random\_Sample\ (V)$;
5:         $(V, E) \leftarrow Extend(G, D, x_{rand})$
6:         If ( $check\ (V,\ scenario\{goal\}) == True$ ) then
7:             Status = Success;
8:             Break;
9:         end if
10:        i=i+1
11:    end While
```

As any RRT; a graph G is created of nodes denoted V $(vertices)$ that stores the final states of the car where the control inputs sequence to travel from initial node (parent) to current node (child) is stored in the edges E. D denotes the Deleted nodes set that contains all nodes that are in collision to avoid repetition.

The *Extend* function enables the tree to grow where each time we select a random node as parent to branch by applying random actions, we add the resulting node to V and applied actions to E if it is in the free collision set. In the case of the resulting node is in collision, we discard the applied action and we save the node in delete set. $Random\_Sample$ function choose randomly a node from V and try to explore wide range area.

Algorithm: Extend

Input: $V, E, D,\ x_{rand}$.

Output: V, E.

```
1:    $V' \leftarrow V; E' \leftarrow E; D' \leftarrow D$
2:    $u_{rand}$ = Get Control Input ( scenario );
3:    $x_{new}(\tau)$ = Simulate Amesim Model ($x_{rand}, u_{rand}$, T );
4:    If collisionFree( $x_{new}(\tau)$ ) and $x_{new}(\tau) \notin D$ then
5:        $V' \coloneqq V' \cup \{ x_{new}(\tau) \}$
6:        $E' \coloneqq E' \cup \{ u_{rand}(\tau) \}$
7:    else
8:        $D' \coloneqq D' \cup \{ x_{new}(\tau) \}$
9:    end if
10:   Return $G' = (V', E')$
```

For lane change scenario in Figure 4. 2; we define two regions current lane region (in green) and target lane region (red) and depending on the vehicle position the control actions and their occurrence probabilities are defined by logical preference (see Table 4. 1).

Table 4. 1: Simple driving rules in lane change

		Current lane (green)	**Target lane (red)**
Steering	*Hard_Left*	0.3	0
	Left	0.5	0
	Straight	0.2	0.2
	Right	0	0.4
	Hard_Right	0	0.4
Acceleration	*Throttle*	0.2	0.3
	No Action	0.5	0.5
	Brake	0.3	0.2

One action is taken at the time from steering set and another action from acceleration set $U = \{\text{Steering}, \text{Acceleration}\}$ thus we define two random variable P_s for Steering and P_a for acceleration. If $P_s = 0.6$ and $P_a = 0.9$ means that *Left* with *Brake* are chosen as control actions. Now, in same way and as discussed above; a numerical value for the selected control actions is set randomly following a uniform distribution function between the maximum and minimum value of each action. Note that *No Action* refer to no input acceleration.

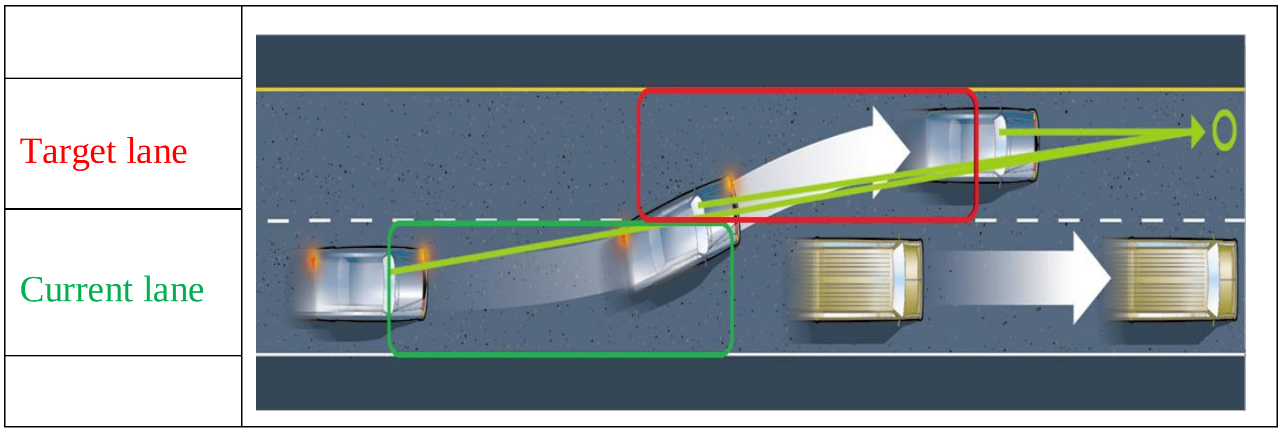


Figure 4. 2: Lane change rules regions.

Figure 4. 3 describes the general structure of path planner like-driver with Amesim Car.

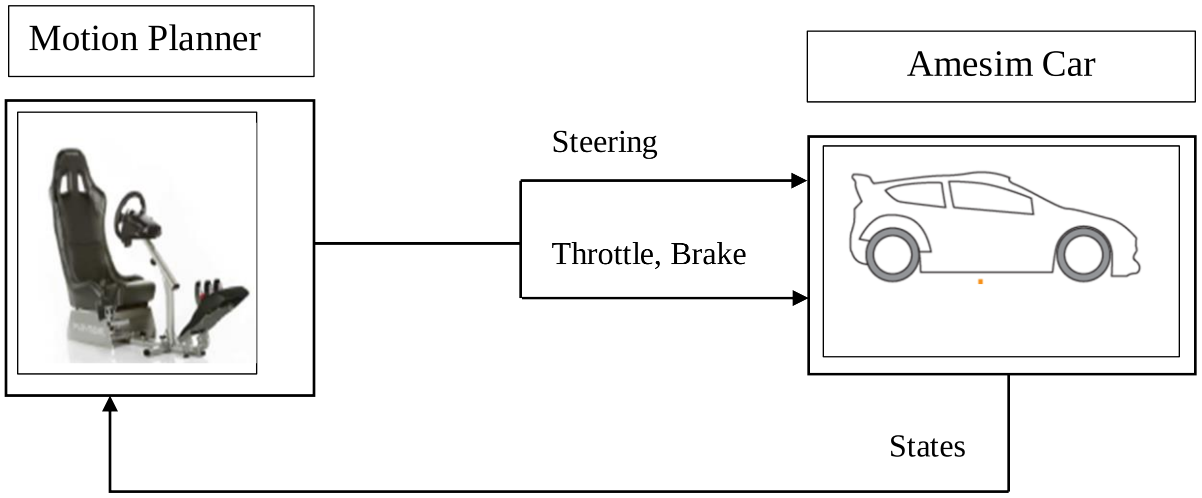


Figure 4. 3: Motion planner Block diagram.

- Motion planner block contains: the scenario map and rules of specific scenario.

- Amesim Car block: describes the high fidelity model of the car that includes the dynamics and tire model.

Chapter 5

Trajectory Control

So far we, the path planner enable the vehicle to find a feasible path with high precision following a sequence of control inputs. However, driving the car in open loop structure will not be sufficient to track a full path which will introduce an accumulative error that drives the car from the reference trajectory. Also, the open loop form is too weak against the different disturbances and external forces (strong wind, sliding surface,…). In order to track successfully the resulting trajectory and ensure the stability of the vehicle, a robust trajectory controller is required.

Model Predictive Controllers (MPC), over literature, have been effectively employed in different applications (UAV, ADAS, …) for their ability to handle systems' dynamics and kinematics constraints, predicate the behavior of the system over a time horizon and optimize the problem to find the best feasible control combination . In autonomous driving, MPC has been widely used such as in Falcone [18]; where the author has compared between the use of the Linear and nonlinear MPC in terms of computational time in double lane change application, the nonlinear MPC depends on the speed profile of the vehicle and very computationally expensive compared to the linear MPC. Nevertheless, the linear MPC uses a linearized model which is not accurate compared to nonlinear MPC model. It is always tradeoff between accuracy and time complexity.

Due to the technological advancement of embedded computers, we prioritize the accuracy over the time complexity in designing trajectory controller, thus a nonlinear model predictive controller (NMPC) based on the bicycle model is implemented in the next section.

5.1 MPC problem Formulation

Model predictive controller is an optimal control approach formulated as optimization problem which involves a prediction model, objective function and constraints. Usually, MPC is carried out with a receding horizon from time (t_k) to a defined time (t_n) (horizon time) where N future output signals are predicted.

The objective function $\mathcal{J}$ is built based on the predicted states $x(t_k), x(t_k + T_s), \dots, x(t_n)$ and control signals $u(t_k), u(t_k + T_s), \dots, u(t_n)$ (sample time $T_s = \frac{t_n}{N}$) and optimized with respect these control signals subjected to constraints; The optimized control signals are inputs to the systems. Each sample time, we repeat the same process.

The dynamic model used in our MPC is the bicycle described in chapter 3:

$$\frac{dx(t)}{dt} = f(x (t) , u(t)) \tag{5.1}$$

$$x(t_0) = x_0 \tag{5.2}$$

$$y(t) = h(x (t) , u(t)) \tag{5.3}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^p$ t is the time and x_0 are the initial conditions. The functions f and h denotes state and output equations respectively. Now, for trajectory tracking application, we define the tracking error as follows:

$$e(t) = y_{ref}(t) - y(t) \tag{5.4}$$

For nonlinear model predictive controller (NMPC), both constraints and objective function are defined nonlinear functions and nonlinear program (NLP) is used to solve the problem, then the NMPC optimal control problem is formulated for a given current state x_k as follows:

$$\min_{u(t)} \int_{t_k}^{t_n=t+N\,T_s} \mathcal{J}(\ u(t), x(t)\ )\ dt + \ \mathcal{J}_f(\ u(t_n), x(t_n)\ ) \tag{5.5}$$

Subject to $\frac{dx(t)}{dt} = f(\ x\ (t)\ , u(t)\ \ )$ (5.6)

$y(t) = h(\ \ x\ (t)\ , u(t)\ \ )$ (5.7)

$x(t_k) = \ x_k$ (5.8)

$u_{min} \leq \ u(t)\ \leq \ u_{max}$ (5.9)

Other constraints

In order to be able to solve the optimal control problem numerically, a transformation is required to obtain a finite dimensional optimization problem. We define two approaches (1) Single shooting and (2) multiple shooting where the later split the horizon into subintervals and parametrizes the state and control trajectories to be included in optimization problem. This numerical property and implementation issue enable the MPC to handle perfectly uncertain and highly nonlinear systems [26]. Thus the finite dimensional optimization problem can be rewritten as follows :

$$\min_{q_k,\dots q_{k+N-1}} \sum_{i=k}^{k+N-1} \mathcal{J}_{i,d}(\ q_i, x_i\ ) + \ \mathcal{J}_{f,d}(\ q_{k+N}, x_{k+N}\ ) \tag{5.10}$$

Subject to $\frac{dx(t)}{dt} = f(\ x\ (t)\ , u(t)\ \ )$ (5.11)

$y(t) = h(\ \ x\ (t)\ , u(t)\ \ )$ (5.12)

$x(t_k) = \ x_k$ (5.13)

$u_{min} \leq \ u(t)\ \leq \ u_{max}$ (5.14)

$u(t) = q_i$ multiple shooting transformation
From $t \in < t_k, \dots, t_N >$ to $i\ \in < k, \dots, k + N - 1 >$ (5.15)

Other constraints

Now the cost function is expressed as the sum of functions $\mathcal{J}_{i,d}$ and terminal function $\mathcal{J}_{f,d}$ which are based on discrete samples of state and input trajectories. Similarly, the constraints are considered at sampling time instants. There are several methods that can be used to solve the above optimization problem Interior point (IP) methods and Sequential Quadratic Programming SQP. Frequently, SQP method is used in solving NMPC optimization problem.

5.2 SQP

Is an iterative method aiming to find optimum solution to nonlinear constrained problem, Consider the nonlinear optimization problem:

$$\min_{\boldsymbol{z}} \boldsymbol{F}(\boldsymbol{z}) \tag{5.16}$$

Subject to $$\boldsymbol{C_E}(\boldsymbol{z}) = \boldsymbol{0} \tag{5.17}$$

$$\boldsymbol{C_I}(\boldsymbol{z}) \leq \boldsymbol{0} \tag{5.18}$$

With $\boldsymbol{F}$ being the objective function, $\boldsymbol{z}$ vector of optimization variables, $\boldsymbol{C_E}$ and $\boldsymbol{C_I}$ are the equality and inequality constraints functions respectively. The idea of Sequential Quadratic Programming (SQP) is to search for local minima by the following advancing approach:

$$\boldsymbol{z}_{(\boldsymbol{i}+\boldsymbol{1})} = \boldsymbol{z}_{\boldsymbol{i}} + \boldsymbol{\alpha}_{(\boldsymbol{i})}\, \boldsymbol{p}_{(\boldsymbol{i})} \tag{5.19}$$

Until it converges to some stopping criterion. $\boldsymbol{\alpha}_{(\boldsymbol{i})}$ is the step length and $\boldsymbol{p}_{(\boldsymbol{i})}$ is the optimizer variable or solution step to the local Quadratic Problem (QP). the nonlinear objective function is approximated locally by its quadratic approximation form with linearized constraints. In each iteration, the local QP is recalculated for current $\boldsymbol{z}$ then the local QP can be rewritten as follows:

$$\min_{\boldsymbol{p}} \boldsymbol{F}(\boldsymbol{p}) \cong \boldsymbol{F}(\boldsymbol{z}) + \boldsymbol{\nabla F^T}(\boldsymbol{z})\,\boldsymbol{p} + \frac{1}{2}\,\boldsymbol{p^T}\,\boldsymbol{\nabla^2 F}(\boldsymbol{z})\,\boldsymbol{p} \tag{5.20}$$

Subject to

$$\boldsymbol{\nabla C_E}^{\boldsymbol{T}}(\boldsymbol{z})\,\boldsymbol{p} + \boldsymbol{C}_E(\boldsymbol{z}) = 0 \tag{5.21}$$

$$\boldsymbol{\nabla C_I}^{\boldsymbol{T}}(\boldsymbol{z})\,\boldsymbol{p} + \boldsymbol{C}_I(\boldsymbol{z}) \leq 0 \tag{5.22}$$

The following algorithm summarizes the SQP approach:

Algorithm: SQP

Input: $\boldsymbol{F}$, $\boldsymbol{C_E}$, $\boldsymbol{C_I}$, initial guess.

Output: $\boldsymbol{z}_{(i)}$

```
1:    Repeat
2:            Local QP $\boldsymbol{F}(\boldsymbol{z}) \rightarrow \boldsymbol{F}(\boldsymbol{p})$ for $\boldsymbol{z}_{(i)}$
3:            Linearize { $\boldsymbol{C}_E$ , $\boldsymbol{C}_I$ } for $\boldsymbol{z}_{(i)}$
4:            Evaluate $\min_{\boldsymbol{p}} \boldsymbol{F}(\boldsymbol{p}_{(i)})$
5:            Select $\boldsymbol{\alpha}_{(\boldsymbol{i})}$
6:            $\boldsymbol{z}_{(\boldsymbol{i+1})} = \boldsymbol{z_i} + \boldsymbol{\alpha}_{(\boldsymbol{i})}\,\boldsymbol{p}_{(\boldsymbol{i})}$
7:            $i = i + 1$
8:    Until : stopping criterion met
```

Among the different objective functions terms used in NMPC, we are interested in the additive quadratic form based on penalty least square. this form is straightforward and very efficient in reference trajectory tracking described in (5.4):

$$\mathcal{J} = \sum_{i=k}^{k+N-1} \left(x_{ref,i} - x_i\right)^T \boldsymbol{Q_i} \left(x_{ref,i} - x_i\right) + u_i{}^T \boldsymbol{R_i}\, u_i + \left(x_{ref,N} - x_N\right)^T \boldsymbol{Q_N} \left(x_{ref,N} - x_N\right) \tag{5.23}$$

Equation 5.23, Can be re-written in the block diagonal form as follows:

$$\mathcal{J} = \left(x_{ref} - x\right)^T \boldsymbol{Q} \left(x_{ref} - x\right) + u^T \boldsymbol{R}\, u + \left(x_{ref,N} - x_N\right)^T \boldsymbol{Q_N} \left(x_{ref,N} - x_N\right) \quad (5.24)$$

Where the $\boldsymbol{Q}$ is trajectory diagonal penalty (weighting) matrix consist of positive semidefinite matrices $\boldsymbol{Q_i}$ of dimension (n)**,** also $\boldsymbol{R}$ is control diagonal penalty matrix consisting of blocks of $\boldsymbol{R_i}$ of dimension (m).

The kinodynamic path planner provides us with time parametrized path $x_{ref} = (U_x \quad U_y \quad r \quad \psi \quad x \quad y)$ and the corresponding control inputs $u_{ref} = (steering \quad throttle \quad brake)$. From equation $\mathcal{J}$, our design take only x_{ref} as tracking reference and try to find the new optimized control input $u = (\delta \quad a)$ to the vehicle under some control penalties, then NMPC is described as follows:

$$\min_{x,u} \mathcal{J} = \left(x_{ref} - x\right)^T \boldsymbol{Q} \left(x_{ref} - x\right) + u^T \boldsymbol{R}\, u + \left(x_{ref,N} - x_N\right)^T \boldsymbol{Q_N} \left(x_{ref,N} - x_N\right) \quad (5.25)$$

Subject to

$$x_{k+1} = f(x_k, u_k) \quad (5.26)$$

$$u_k \leftarrow \begin{Bmatrix} \delta_{min} \leq \delta \leq \delta_{max} \\ a_{min} \leq a \leq a_{max} \end{Bmatrix} \quad (5.27)$$

$$x(0) = x_0 \quad (5.28)$$

The NMPC described above have been implemented in Matlab using ACADO Toolkit interface that contains a collection of dynamic optimization algorithms [27]. C code will be automatically generated to be used in Simulink or Embedded system in the future work.

We distinguish three different possible Trajectory control structures:

- *Open loop control:*

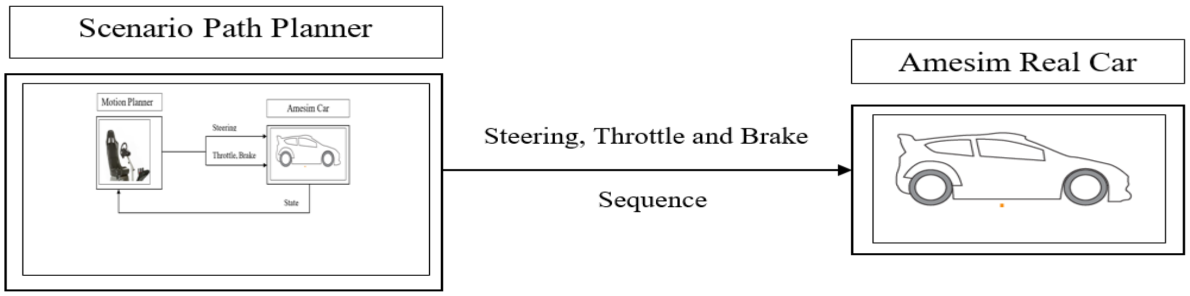


Figure 4. 4: Open Loop Control.

- *Feedback Trajectory Control:*

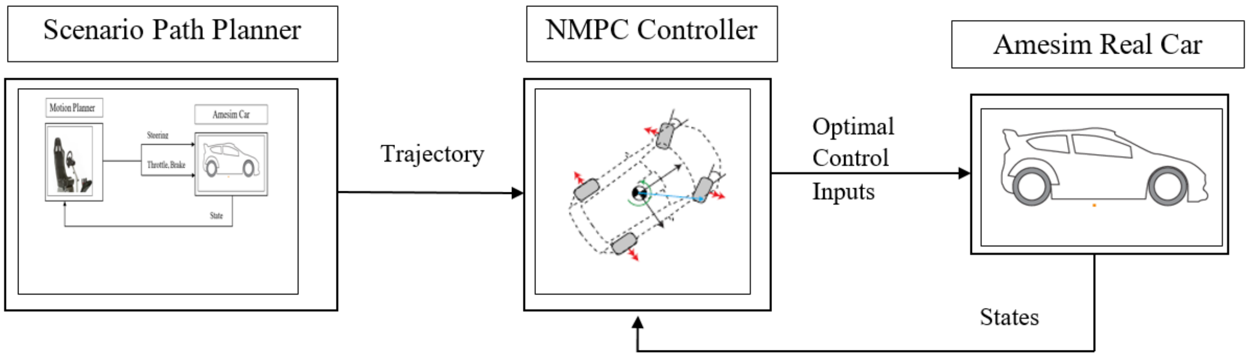


Figure 4. 5: NMPC feedback Trajectory Controller

- *Feedback + Feedforward Trajectory Control:*

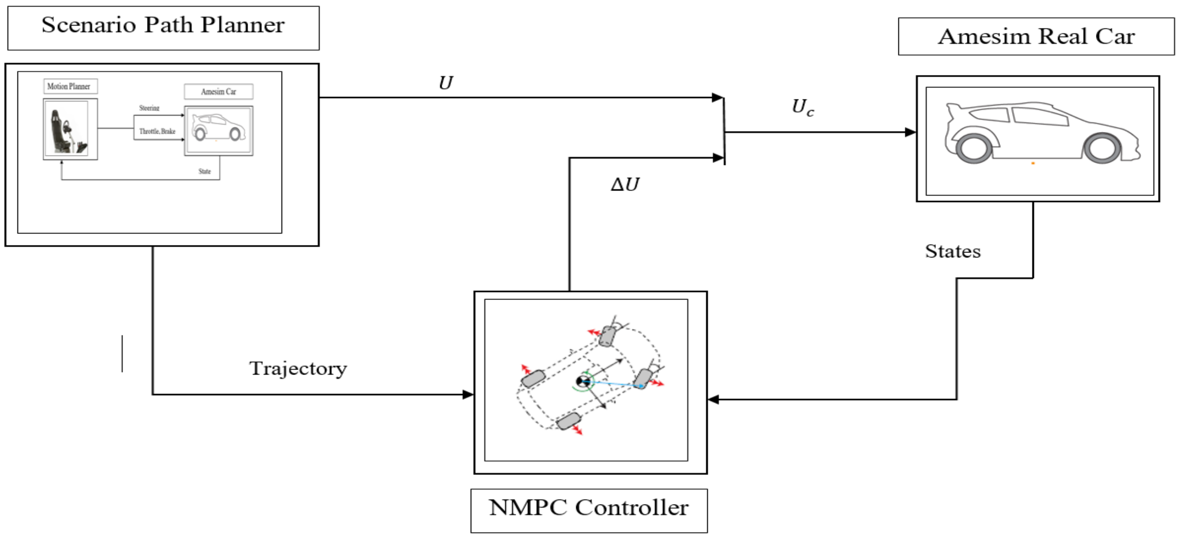


Figure 4. 6: Combined Trajectory Control.

Chapter 6

Simulation Result

6.1 Double lane change scenario

The severe double lane-change maneuver is a dynamic process consisting of rapidly driving a vehicle from its initial lane to another lane parallel and returning back to the initial lane while avoiding a defined obstacle and without exceeding lane boundaries. The double lane change scenario ISO 3888-1 was used to evaluate the effectiveness of planner and proposed control schemes. The ISO 3888-1 track dimensions are mentioned on Table 6. 1 as shown in Figure 6. 1:

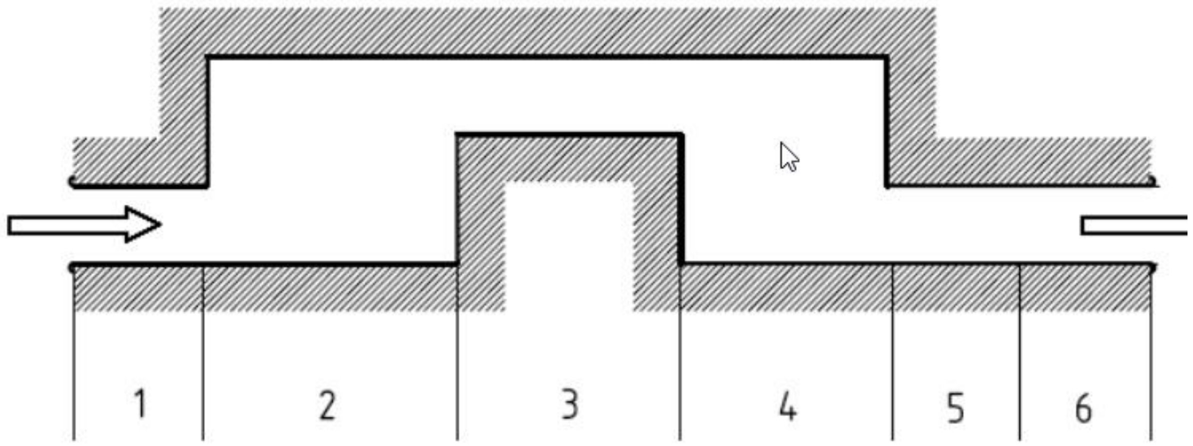
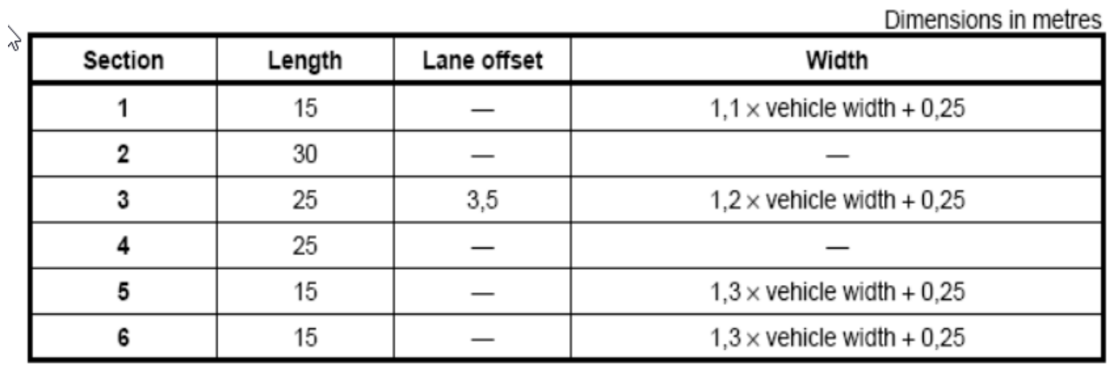


Figure 6. 1: ISO 3888-1 Track for double lane change test.

Table 6. 1: ISO 3888-1 Track Dimensions



Dimensions in metres

Section	Length	Lane offset	Width
1	15	—	1,1 × vehicle width + 0,25
2	30	—	—
3	25	3,5	1,2 × vehicle width + 0,25
4	25	—	—
5	15	—	1,3 × vehicle width + 0,25
6	15	—	1,3 × vehicle width + 0,25

6.2 RRT Motion planner in Double lane Change

In his section, we have explored the ISO 3888-1 track using our motion planner like driver in two different driving parametrization :

- **Case 1:** Only steering command and no Throttle or Brake actions. Then the path planner explores the environment using control set defined as u:{ $steering$ } and initial velocity at the starting point.

- **Case 2:** full command set u:{ $steering, throttle, brake$ }. The idea here is to explore the environment with specific constant speed profiles. Simple PI controller is used within path planner to regulate the speed at specific speed when it is needed.

We define 7 different regions for the whole track as follows:

- 1 : STARTING.
- 2 : TURN LEFT.
- 3 : BACK STEERING RIGHT.
- 4 : TURN RIGHT.
- 5 : BACK STEERING LEFT.
- 6 : STRAIGHT.
- 7 : FINISH.

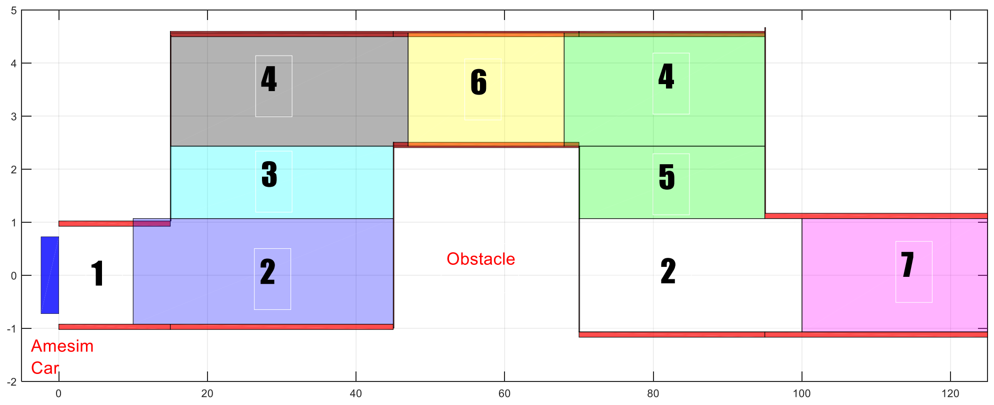


Figure 6. 2 Double lane change regions parametrizations.

a. Case 1:

Considering that the car is having initial velocity $U = 80\ km/h$, the path planner is executed to perform the double lane change with only steering command. to do so, we have parametrized the steering actions for each region as follows:

Table 6. 2: Double lane change with steering parametrization only.

		Regions						
		1	2	3	4	5	6	7
Steering	*Hard_Left*	0	0.3	0				
	Left		0.6	0.1	0	0.5	0.3	
	Straight	1	0.1	0.4	0.1	0.4		
	Right	0		0.5	0.6	0.1		
	Hard_Right			0	0.3	0		
Acceleration	*Throttle*	0						
	No Action	1						
	Brake	0						

The path planner successfully explored the track and reached the finish region with steering control action. The RRT exploration tree (in blue) is shown in Figure 6. 3

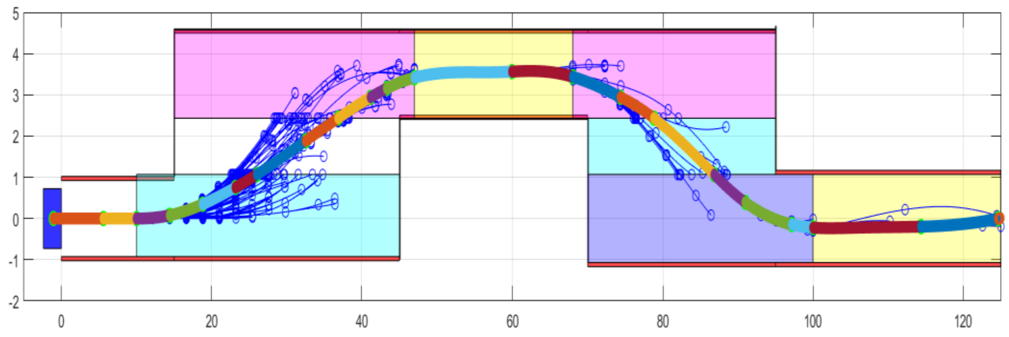


Figure 6. 3: RRT motion planner exploration with steering action only.

The green nodes in figure are the waypoints of the resulting trajectory, between each waypoint we apply new control set that takes the car to the next waypoint. At the end, the resulting trajectory is a sum of sub trajectories where in Figure 6. 4 are represented with different colors.

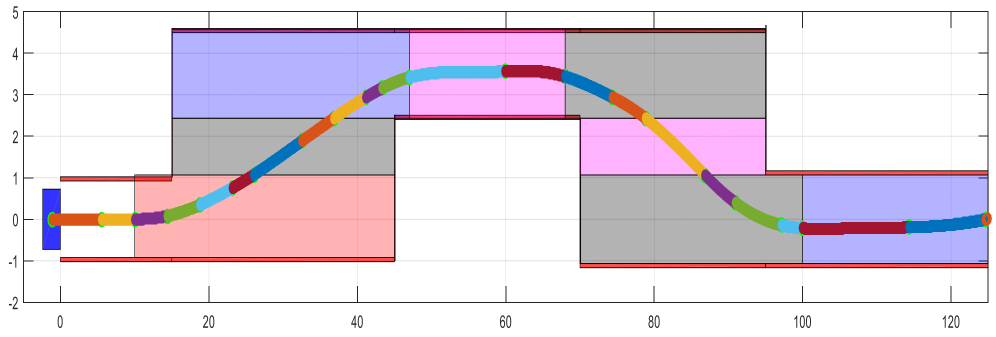


Figure 6. 4: Feasible trajectory with steering only.

To elaborate more the success of the planning, we draw the Vehicle positions in the resulting trajectory (Figure 6. 5). Clearly the car is not in collision with road borders.

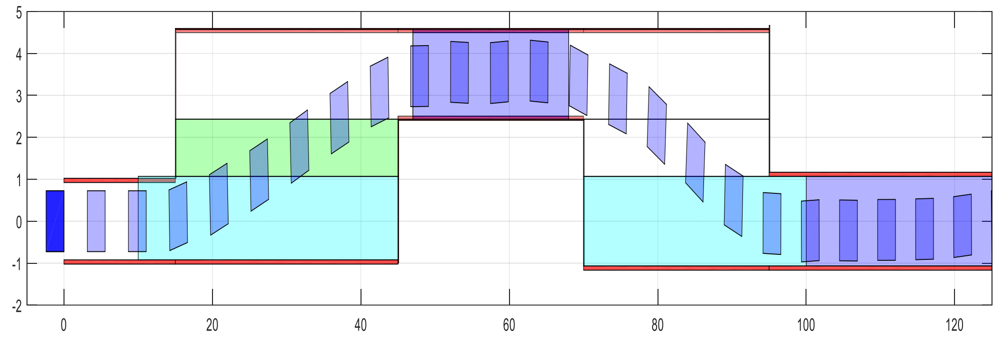


Figure 6. 5: Car position during the planning with steering only .

The control sequence applied to enable the car to reach the finish region is shown in Figure 6. 6, where no brake or throttle commands were used.

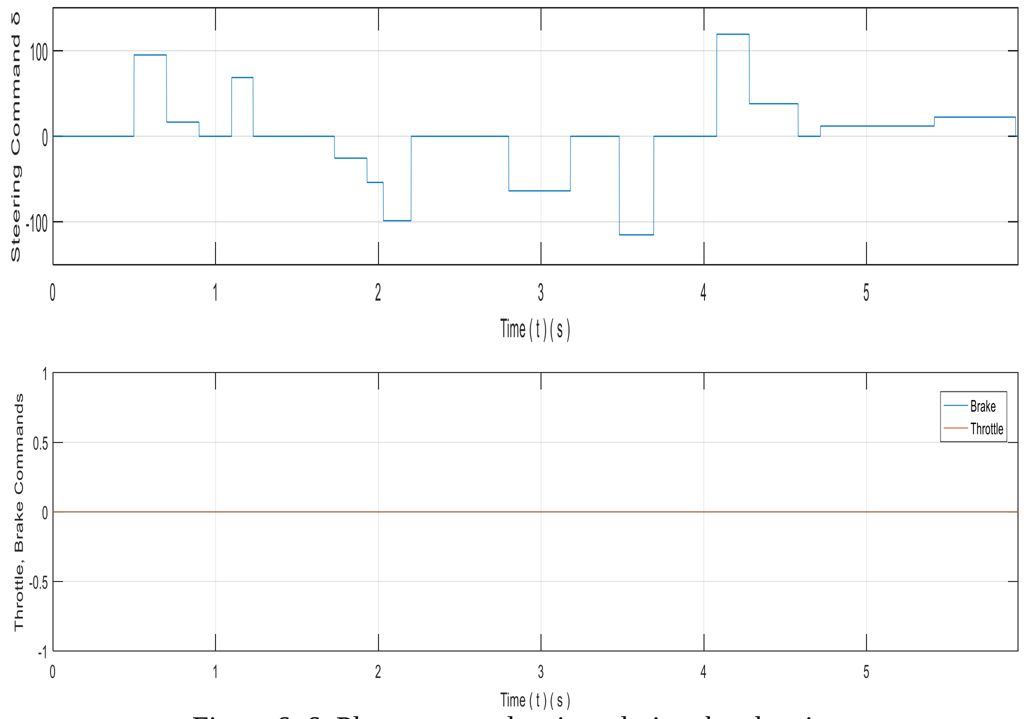


Figure 6. 6: Planner control actions during the planning.

b. Case 2:

In the same tack and for the same scenario, we executed our motion planner again to explore with possibility of acceleration or decelerating toward the finish region, Brake and Throttle appearance are configured as shown in the following Table 6. 3:

Table 6. 3: Double lane change with steering and acceleration rules.

		Regions						
		1	2	3	4	5	6	7
Steering	*Hard_Left*	0	0.3	0				
	Left		0.6	0.1	0	0.5	0.3	
	Straight	1	0.1	0.4	0.1	0.4		
	Right	0		0.5	0.6	0.1		
	Hard_Right			0	0.3	0		
Acceleration	*Throttle*	0.25	0.2				0.3	0.4
	No Action	0.50	0.6				0.6	0.5
	Brake	0.25	0.2				0.1	

The motion planner successfully reaches the finish region with connected trajectory using complete set of control inputs which increases the number of possible combinations then the resulting tree is bigger than the previous one:

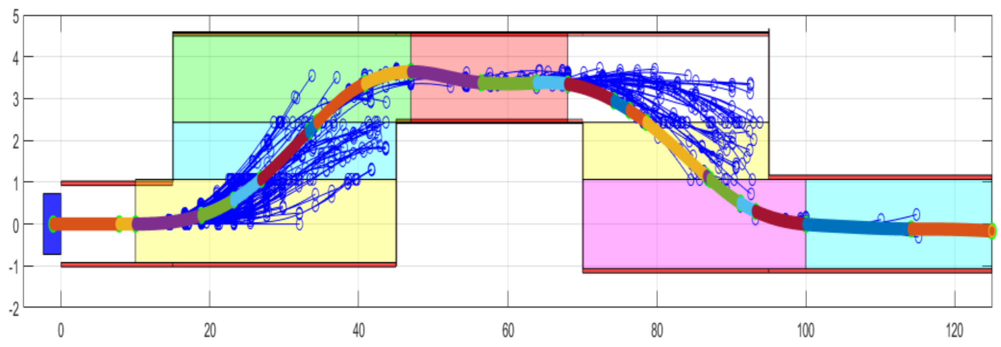


Figure 6. 7: Motion planner exploration.

The resulting connected trajectory and the vehicle evolution in the free state space are shown in Figure 6. 8 and Figure 6. 9:

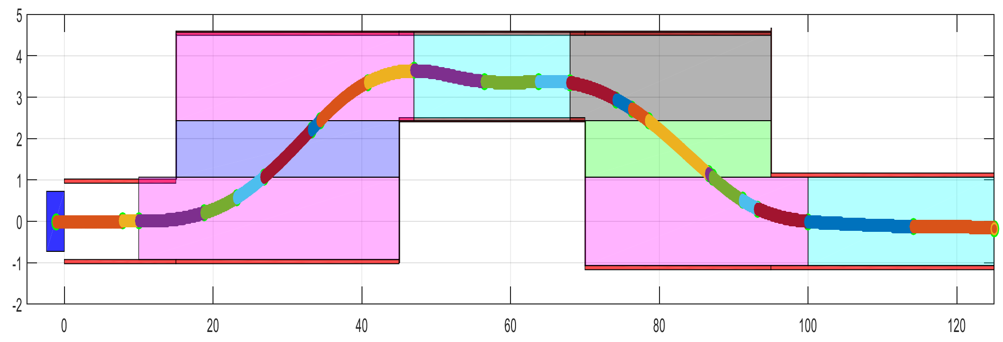


Figure 6. 8: Resulting feasible trajectory.

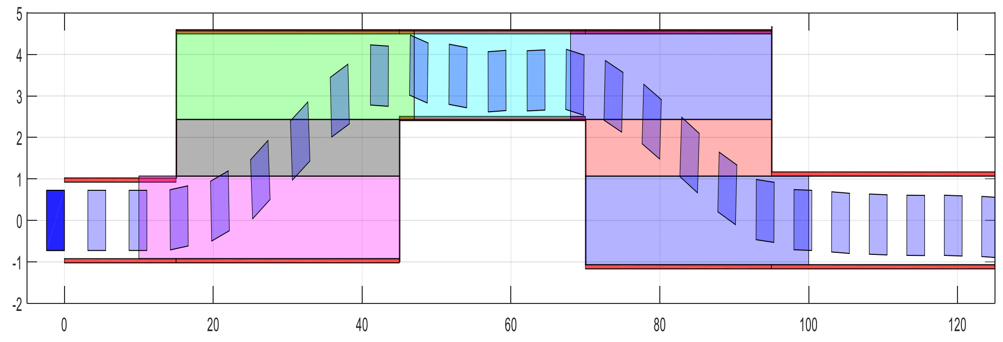
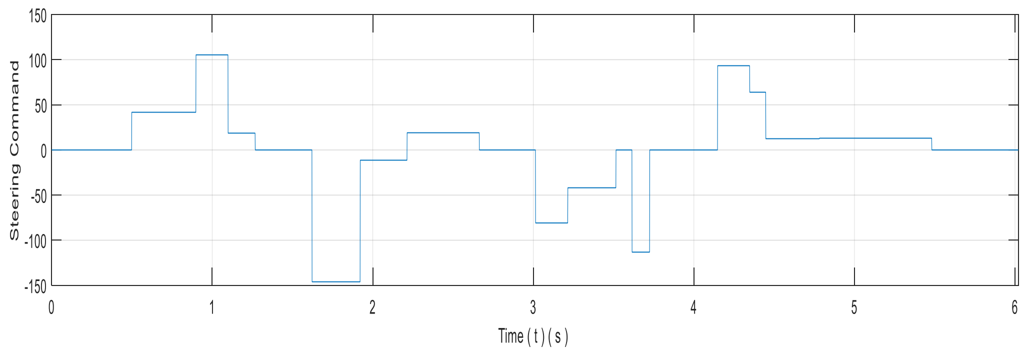


Figure 6. 9: Car locations during the planning.

The input command combination Steering, Throttle and Brake used by the planner are shown Figure 6. 10 .



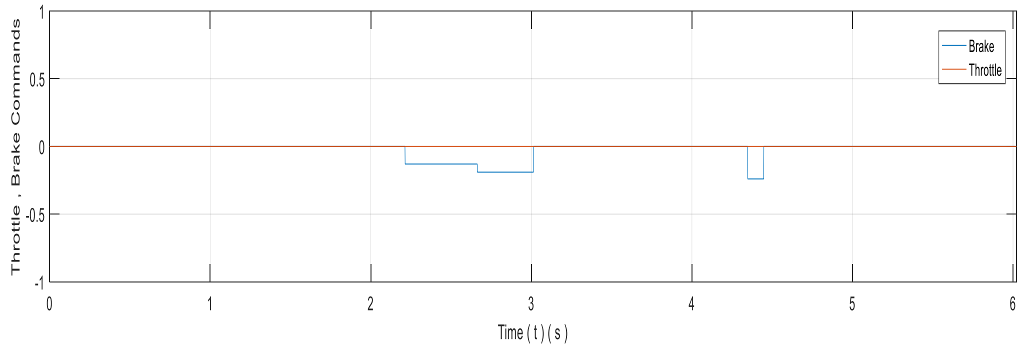


Figure 6. 10: Planner control actions during the planning.

Clearly, it can be seen that only brake command was used to reduce the speed. Most of the acceleration attempts make the car in collision with road borders.

6.3 Nonlinear Model Predictive Controller (NMPC)

In this section, we are going to discuss the performance of the designed NMPC for track trajectory purpose. we have used two different trajectories generated by Amesim In *Circuit Nevers Magny Track* in Bourgogne - France (Figure 6. 11):

- Constant Speed Trajectory.
- Variable Speed Trajectory.

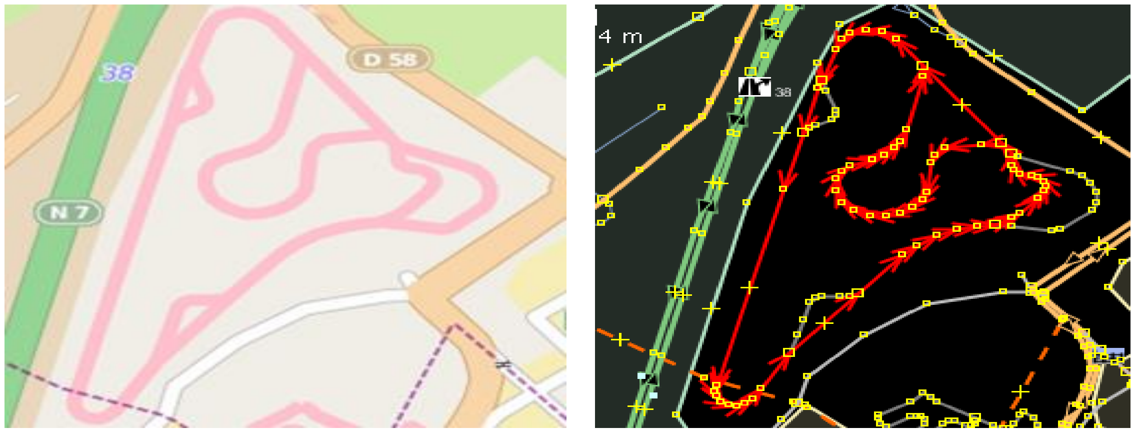


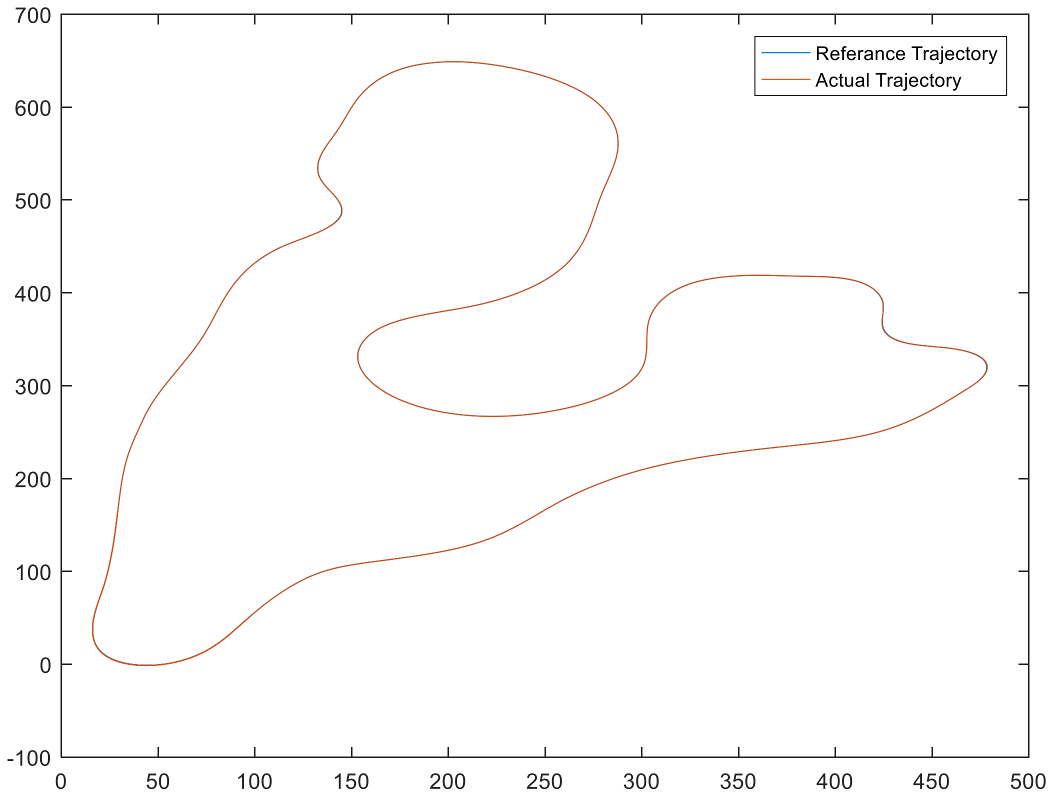
Figure 6. 11: Circuit Nevers Magny Track

a. Constant Speed Trajectory:

The velocity of the car is maintained constant $U = 10\ m/s$ in the whole track staring from an initial position ($\psi = \ -93\ deg,\quad x = 29\ m \quad y = 162.5\ m$), the horizon length N = 40 with sampling time $T_s = 0.001\ s$, the weighting coefficients are set to be:

States Penalties	q_{U_x}	q_{U_y}	q_r	q_ψ	q_x	q_y	Control Penalties	r_δ	r_a
	0.9	0	0.5	10	1	1		0.05	0.01

The NMPC tracks perfectly the reference trajectory where small drift about 20 cm appears in hard turns. The reference and result trajectories are plotted in Figure 6. 12.



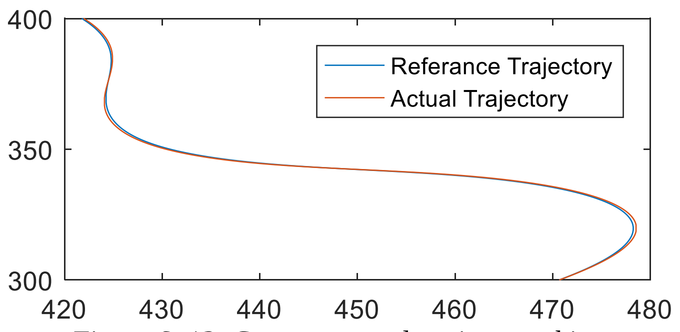


Figure 6. 12: Constant speed positon tracking.

The controller also maintains the speed of the car at 10 m/s in most of the time by accelerating using throttle and decelerating using brakes.

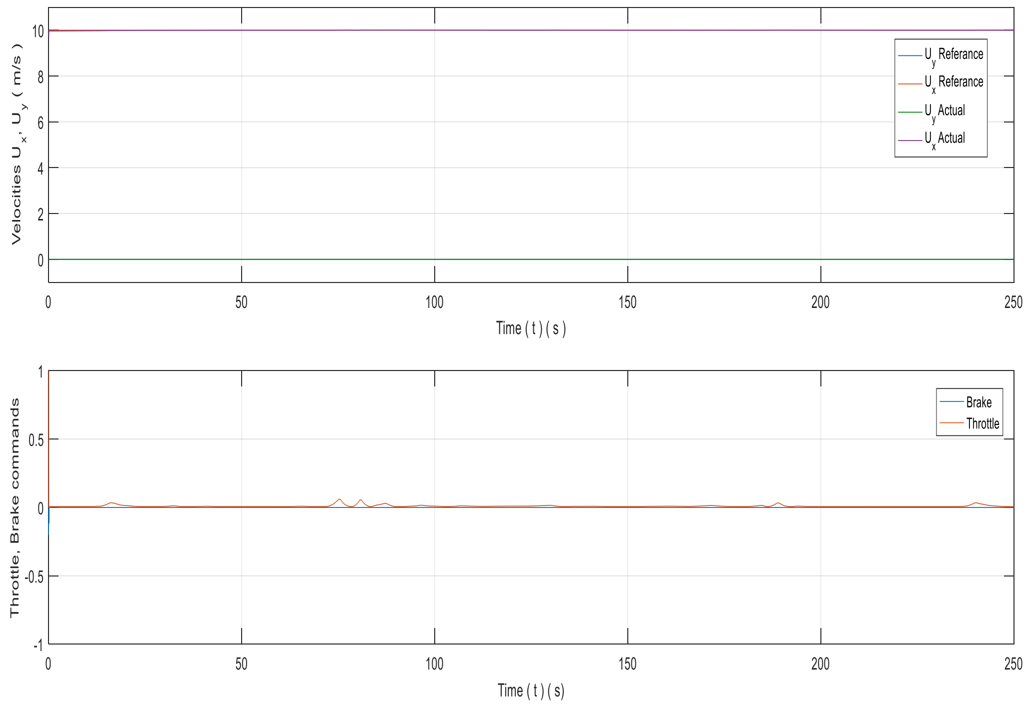


Figure 6. 13: (a) Constant speed Tracking, (b) Throttle & Brake optimal control inputs.

The main challenges is to follow a trajectory orientations the Yaw (ψ) and the Yaw rate (r) to avoid large drifting in hard turns. we would like also to avoid oscillations in control inputs especially in the steering which is undesirable in driving behavior. Therefore, we have tuned our NMPC to compromise between the two objectives (1) small error and (2) no oscillations in control signals.

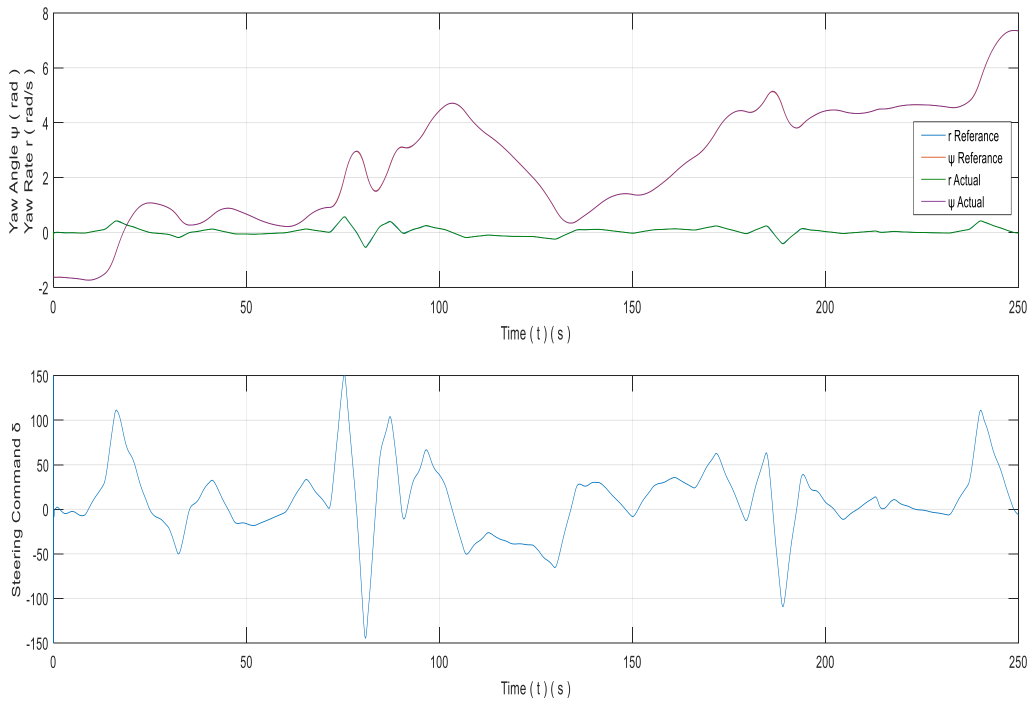


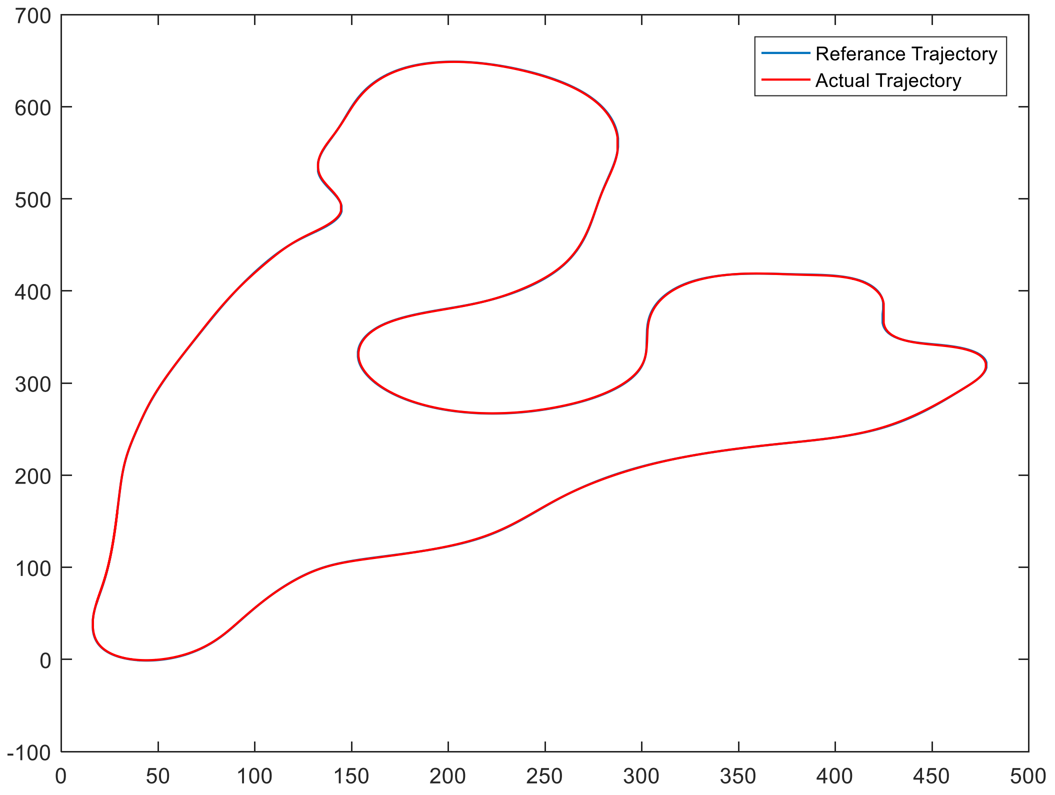
Figure 6. 14: (a) Orientation tracking with (b) optimal steering command.

b. Variable Speed Trajectory:

Now, the controller tracks the same trajectory with variable speed ranges from (10 m/s to 30 m/s). The initial speed of the car is set to be 30 m/s and we start again from the same initial position ($\psi = -93\,deg,\quad x = 29\,m \quad y = 162.5\,m$) without changing the weighting coefficients and MPC Horizon length:

States Penalties	q_{U_x}	q_{U_y}	q_r	q_ψ	q_x	q_y	Control Penalties	r_δ	r_a
	0.9	0	0.5	10	1	1		0.05	0.01

The NMPC shows a good performance in tracking trajectory where slightly larger drift about (40 cm) appears in hard turns especially in higher speed profiles.



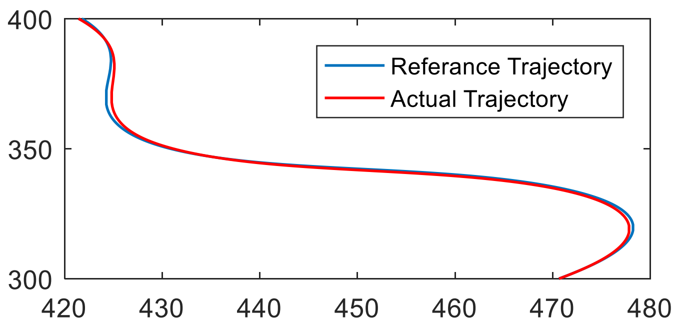


Figure 6. 15: Variable speed positon tracking.

The controller perfectly trackers the desired speed and orientation by finding the optimal control inputs Steering Angle, Brake and Throttle as shown in Figure 6. 17:

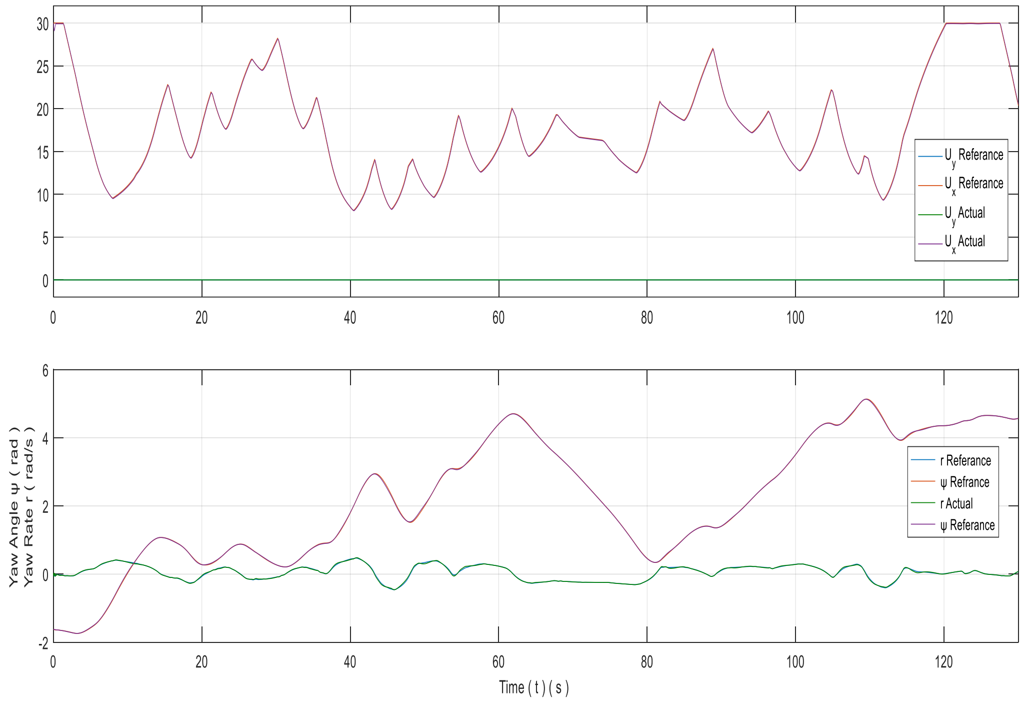


Figure 6. 16: (a) Variable speed and (b) Orientation Tracking.

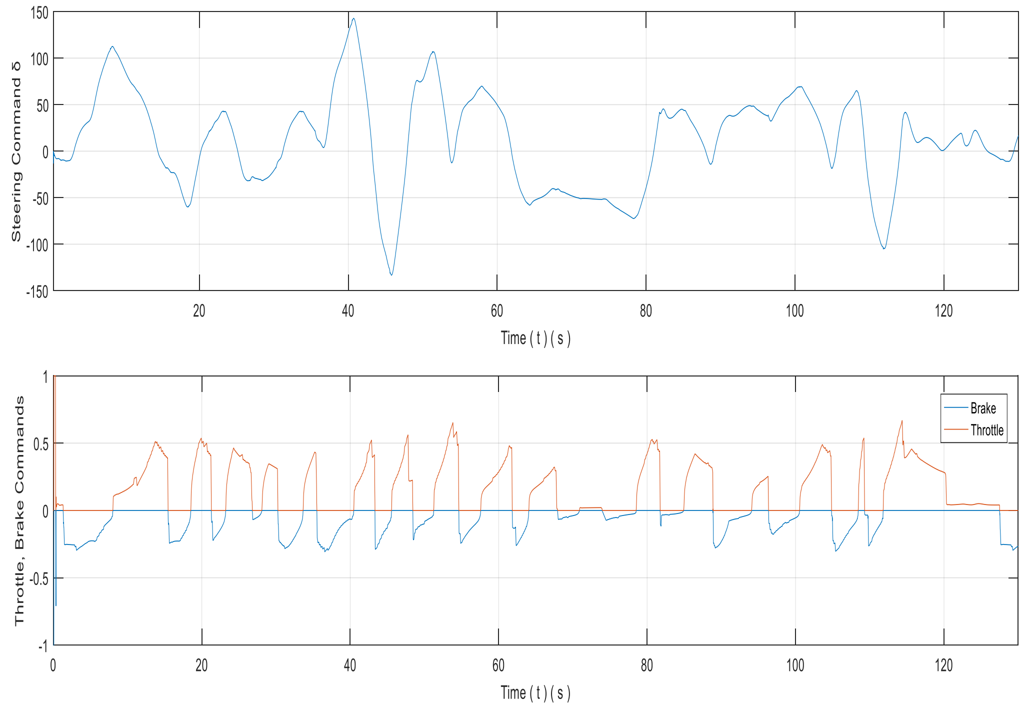


Figure 6. 17: Optimal control Inputs (a) Steering (b) Brake & Throttle.

Our NMPC is developed using the complete model that includes the dynamics of the car and tire forces that allow the vehicle to slip while tracking reference trajectory. The pervious reference trajectory is good reference to test the performance of the controller in slippery track, the evolution of the tire forces and slip angles of the front and rear wheels are shown in Figure 6. 18:

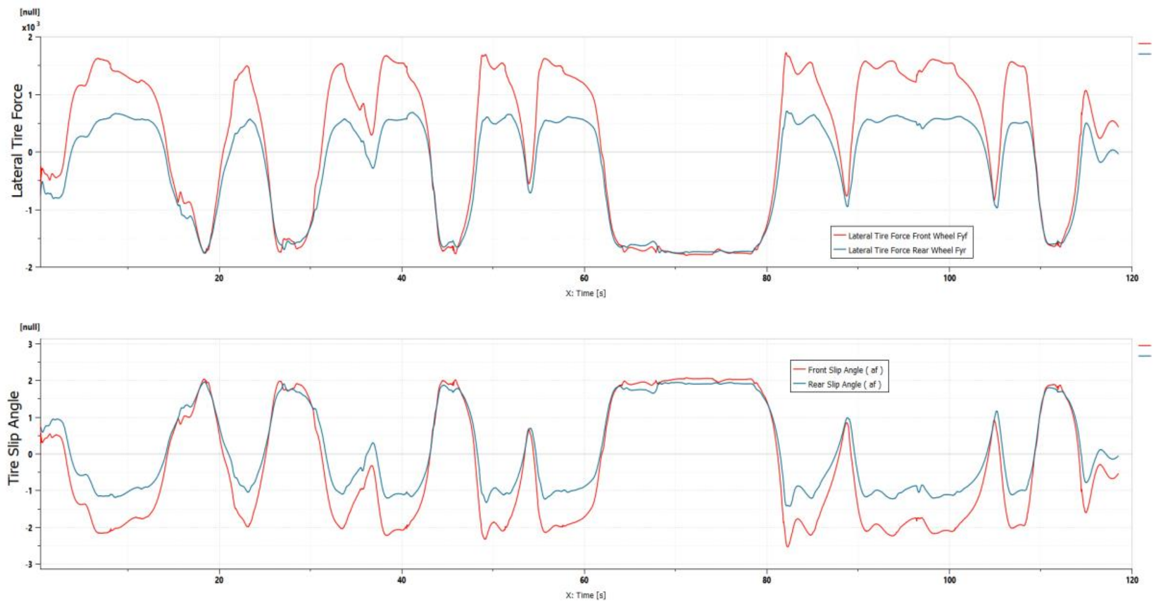


Figure 6. 18: (a) Lateral tire forces and (b) Slip angles of front and rear wheels.

6.4 Double lane change control schemes

For double lane change application, control trajectory design is quite challenging since we would like to meet the following criterion:

- Safe double lane change with very small error in position and orientation (free Collision).
- No oscillation in control inputs (mainly steering)
- Very small time delay and tracks the speed very well.

To achieve these requirement we will present different control approaches:

6.4.1. Open Loop Controller

Since the resulting trajectory is achieved by applying random combination of control inputs in small time durations, it can be also straightforward to use an open loop control structure to execute the time parametrized path as described in Figure 4. 4: Open Loop Control.Figure 4. 4.

The Figure 6. 19, Figure 6. 20 shows that applying the control inputs in open loop structure will give bad results and the car is not following the reference trajectory at all since the error is accumulating during the execution between each two sub paths.

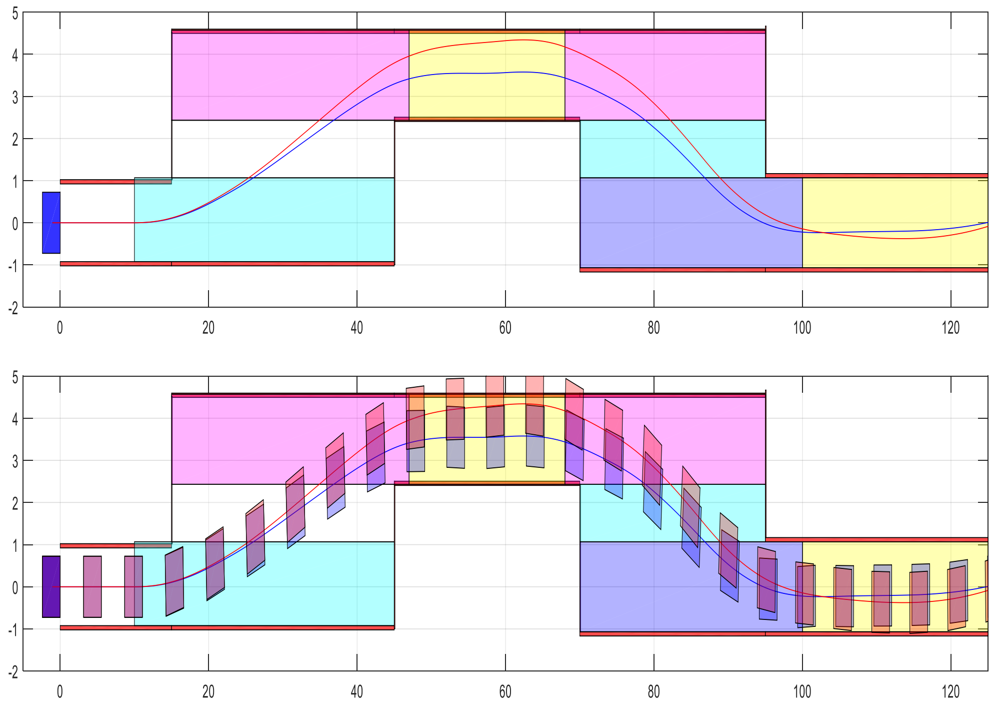


Figure 6. 19: Open loop control (a) resulting trajectory (b) car in collision.

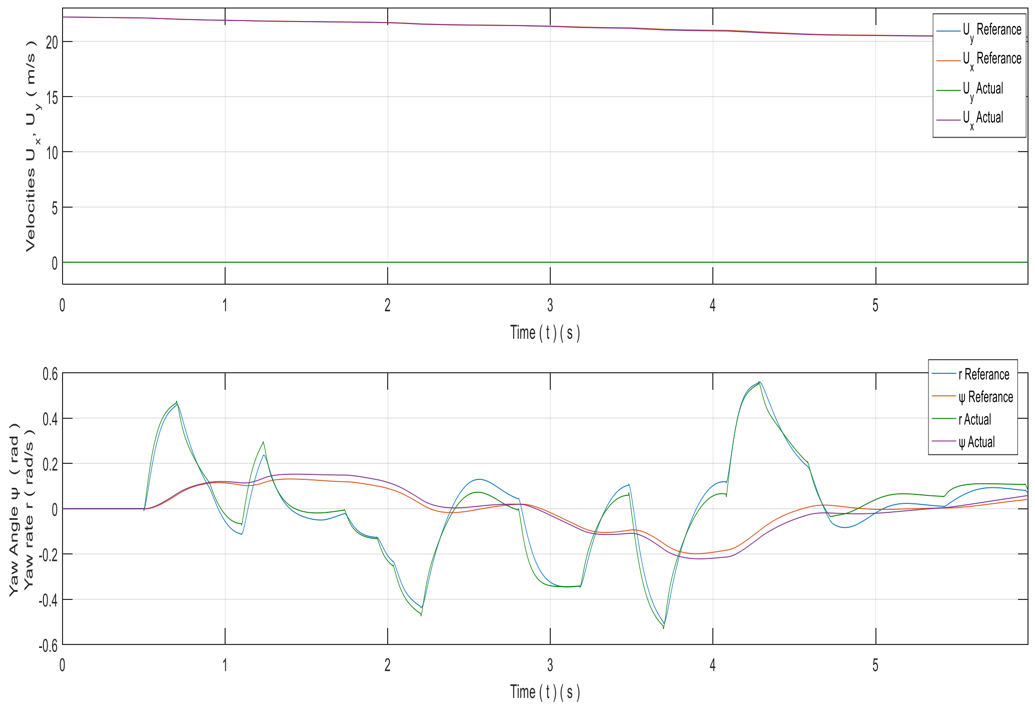


Figure 6. 20: Open loop orientation deviation.

Besides to that, the car is an uncertain systems that suffers from its internal parameter variations as tire model, suspension system… and external disturbances (wind, slippery road ….). Therefore, the open loop structure is replaced by feedback structure

6.4.2. Feedback Controller

For safety requirement, we will use our NMPC with two different tunings coefficients and we compare the performance of the two:

In NMPC (1) there is no difference between the longitudinal and lateral quantities $q_x = q_y$ and higher control penalty on the steering to avoid the steering oscillation . in the other hand in NMPC (2), our main concern is to avoid collision with road borders therefore we have increased the weighting coefficient of lateral component and orientation, and we have also reduced the penalty on the steering.

	States Penalties						Control Penalties	
	q_{U_x}	q_{U_y}	q_r	q_ψ	q_x	q_y	r_δ	r_a
NMPC (1)	0.9	0	0.5	10	1.2	1.2	0.085	0.04
NMPC (2)				20	1	1.5	0.065	

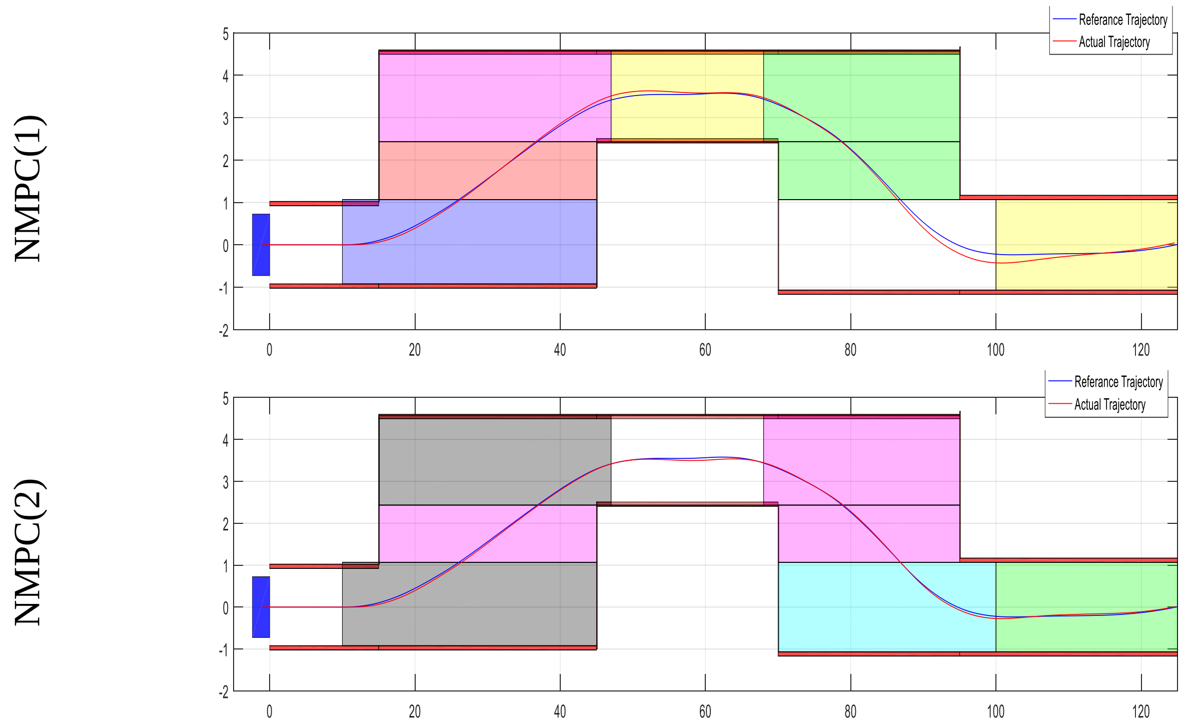


Figure 6. 21: Resulting paths of both MPC s.

In Figure 6. 21 ,The NMPC (2) tracks better the trajectory since the objective function is very sensitive to the lateral and orientation errors. As consequence, In NMPC (2) the car avoid to hit the right borders of road in retuning back to the ego line while in NMPC (1) the car is in collision as shown in zoom Figure 6. 23.

NMPC(1)

NMPC(2)

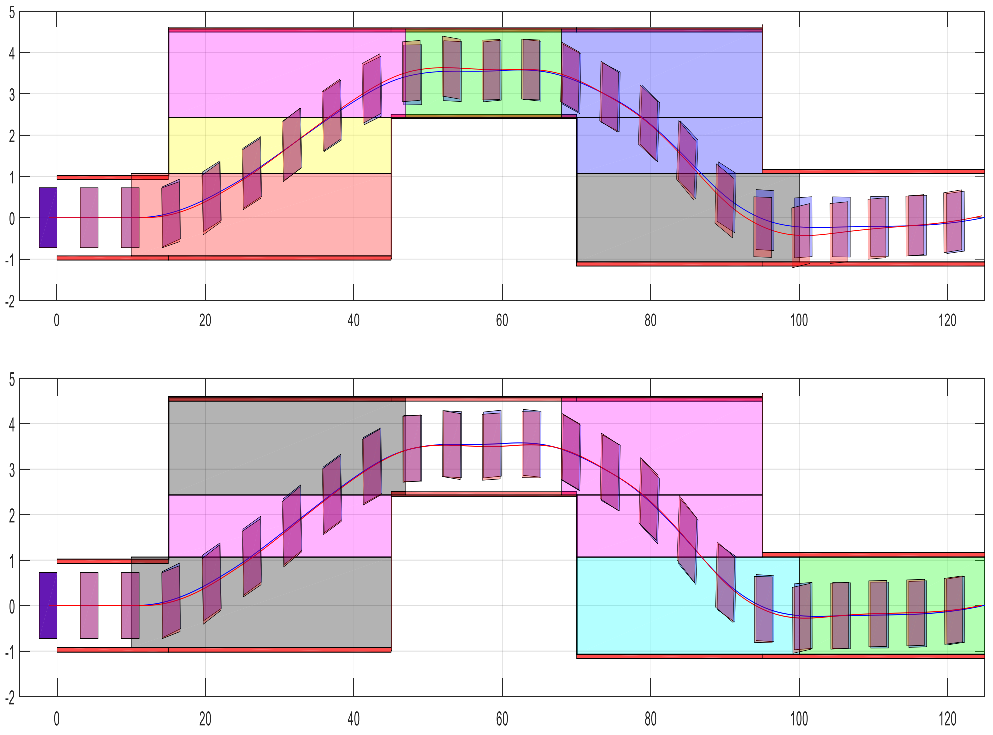
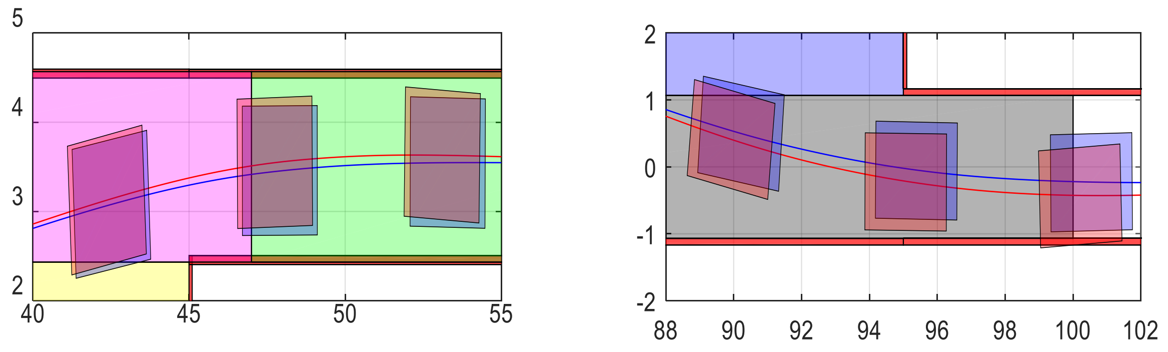


Figure 6. 22: Car locations during the tracking.

In the NMPC (2) introduces small delay compared to NMPC (1), the cars of the resulting and reference trajectories almost overlapped in Figure 6. 22, whereas; the larger delay is introduced in NMPC (1).

NMPC(1)



NMPC(2)

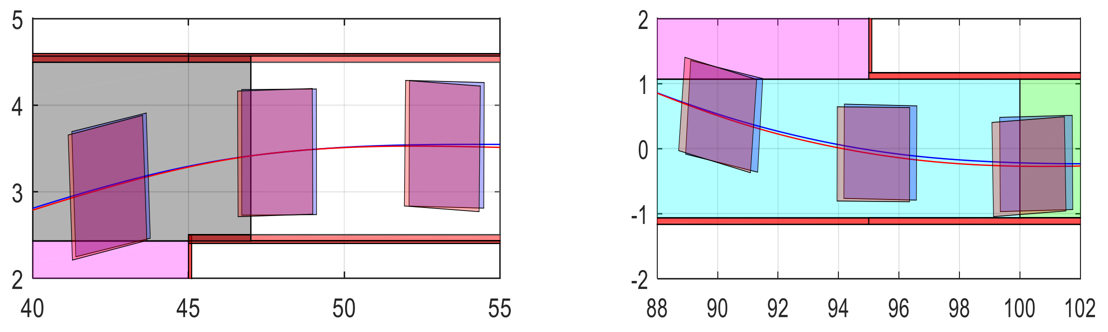


Figure 6. 23: (a) Collision (b) No Collision.

Slightly difference between the two NMPC s in tracking the velocity and orientation references as shown in Figure 6. 24, Figure 6. 25.

NMPC(1)

NMPC(2)

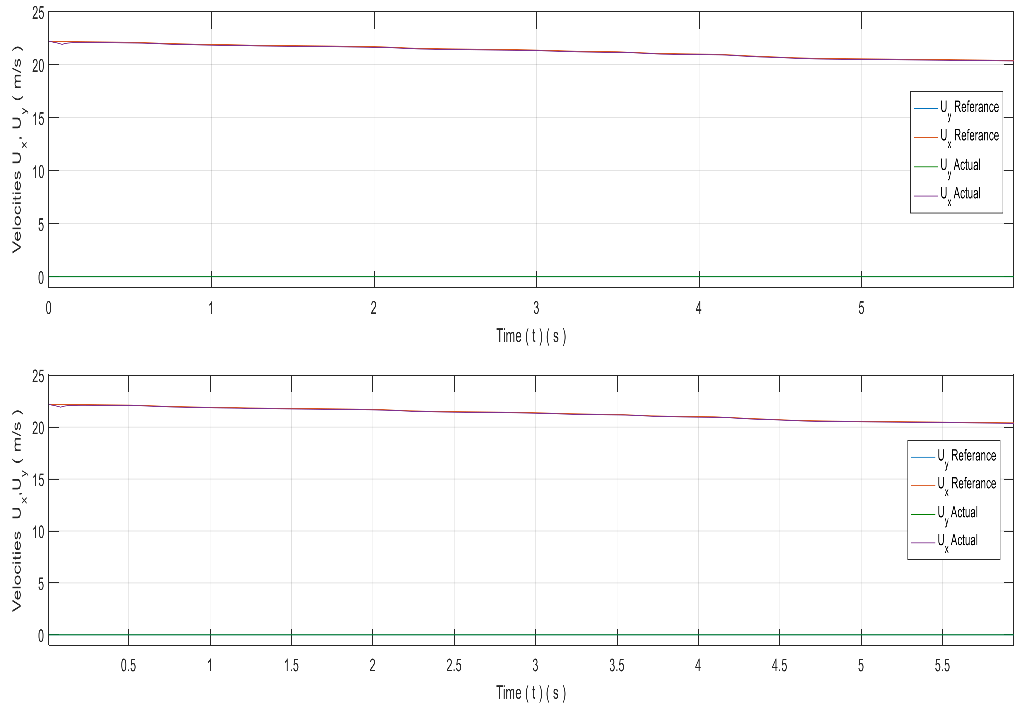
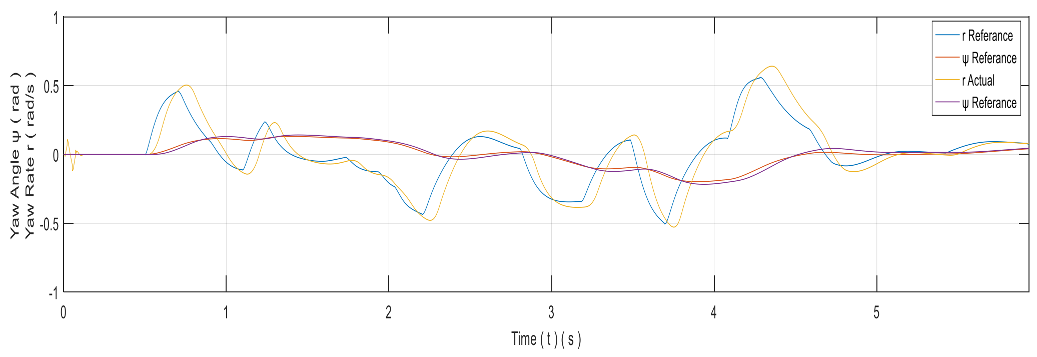


Figure 6. 24: Speed Tracking of both NMPC configurations.

NMPC(1)



NMPC(2)

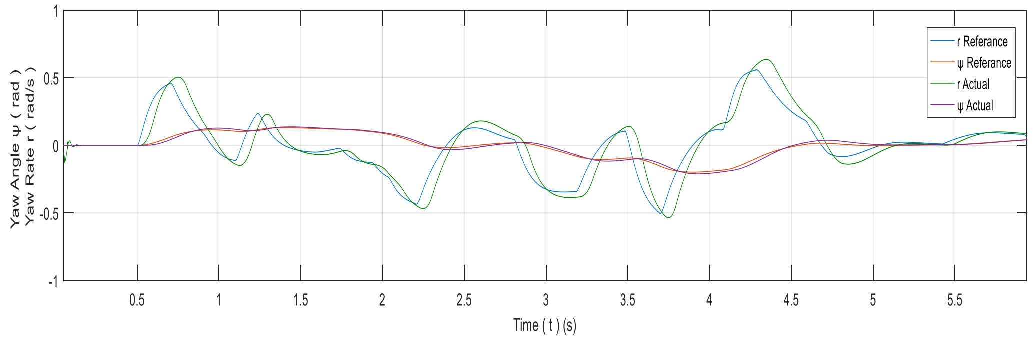
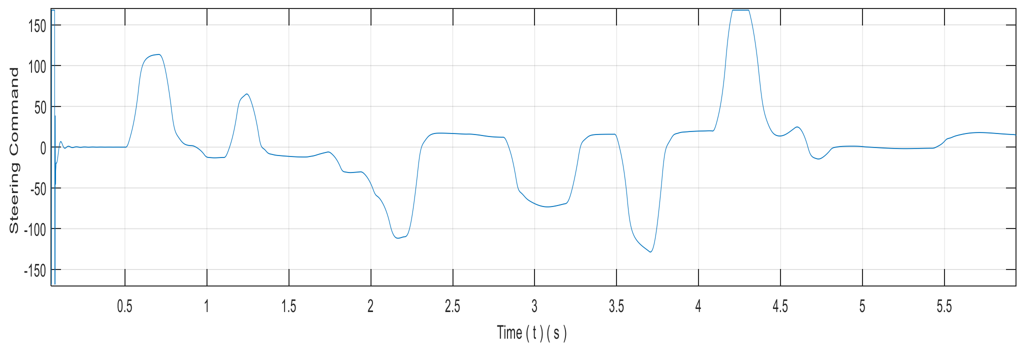


Figure 6. 25: Orientation Tracking of both NMPC configurations.

The NMPC(1) introduces small oscillations at the starter compared to NMPC (2) after that it goes fine for both of them. The throttle and brake control signals for both NMPCs are almost the same.

NMPC(1)



NMPC(2)

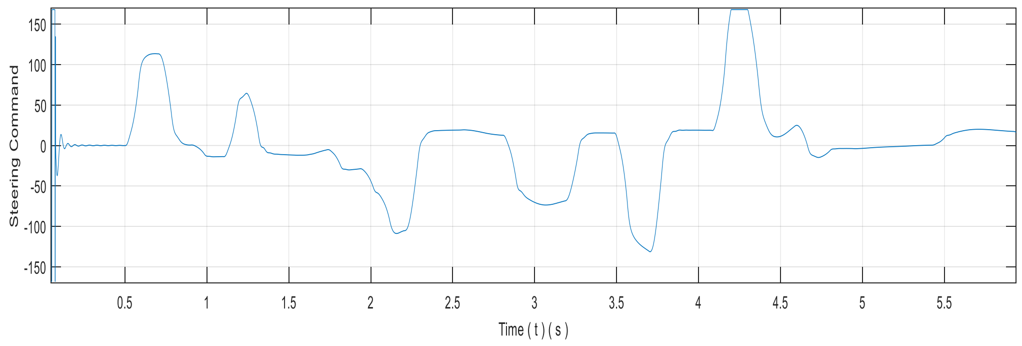


Figure 6. 26: Steering Command of both NMPC configurations.

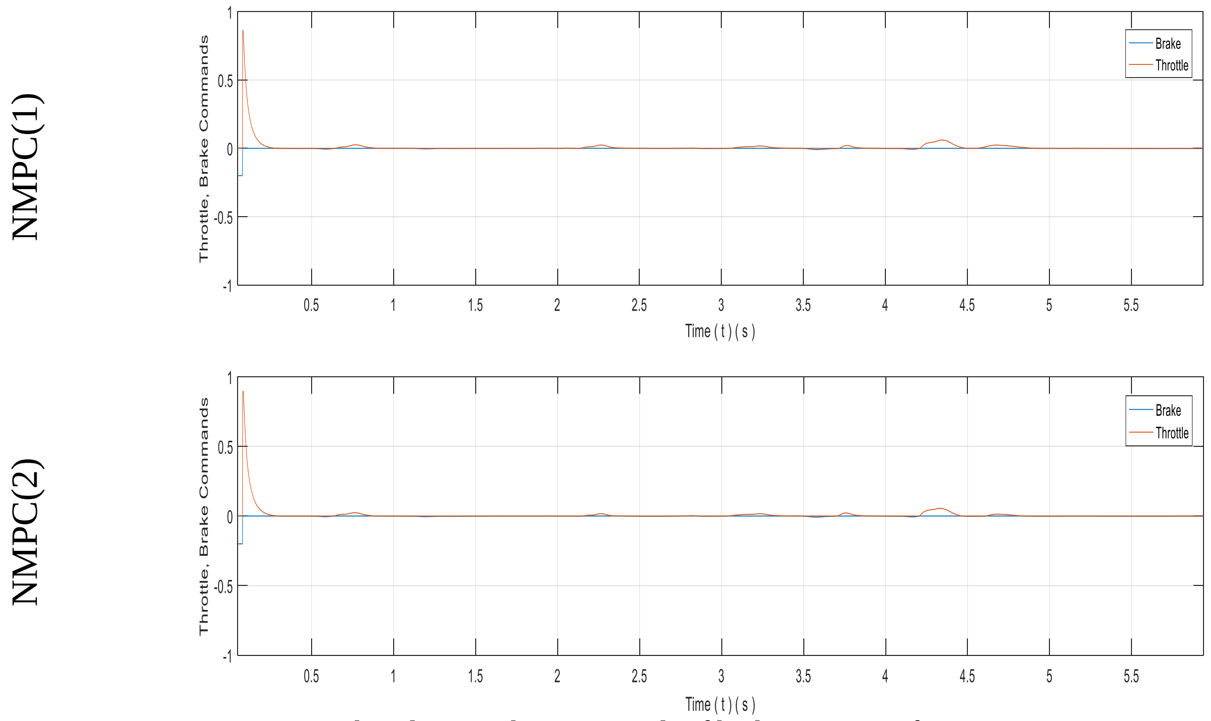
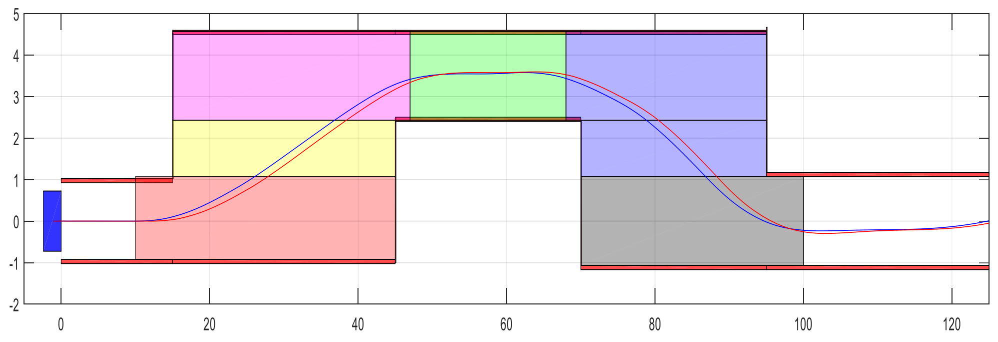


Figure 6. 27: Throttle & Brake Commands of both NMPC configurations.

6.4.3. Feedforward + Feedback Controller

In this section, we use the control sequence from the path planner as feedforward control inputs as driver and NMPC will play the role of the correction block to keep the car in the reference trajectory.

The performance is acceptable at some level, however still need to be enhanced and develop coordination mechanism to have a better control inputs unlike in Figure 6. 30. A delay is introduced in controller as shown in Figure 6. 28 which is totally undesirable.



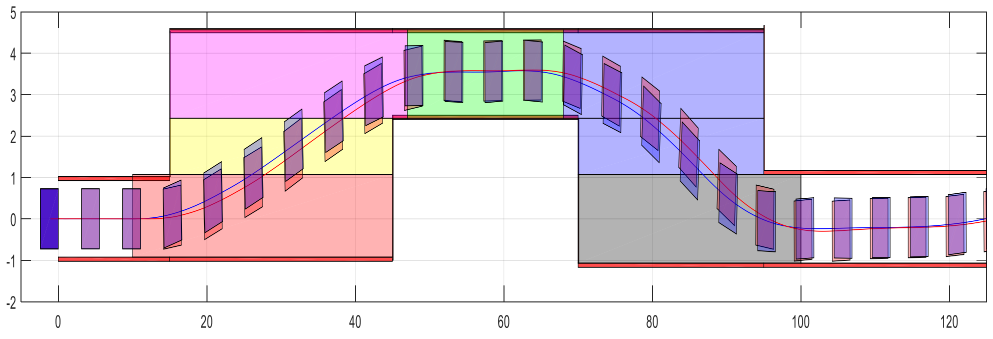


Figure 6. 28: Feedforward + NMPC tracking performance.

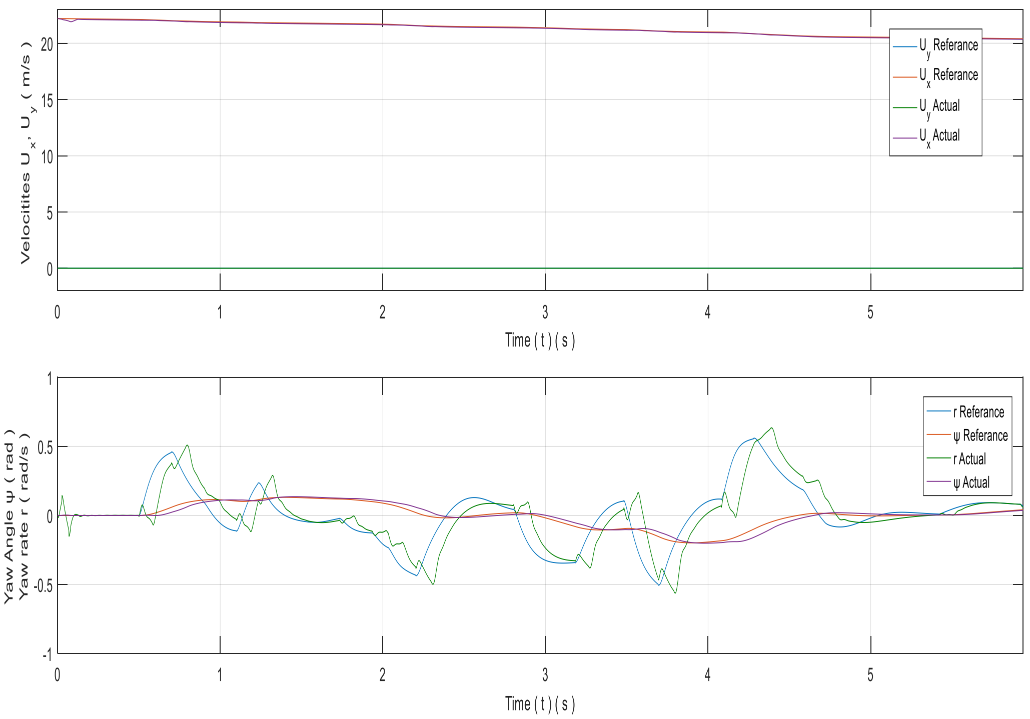
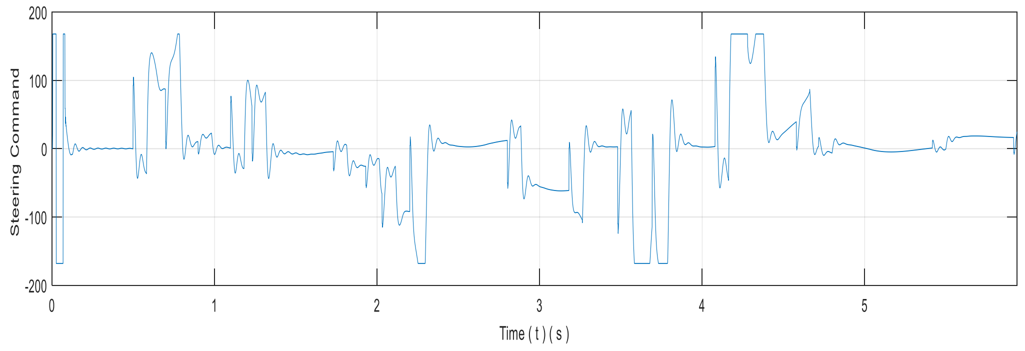


Figure 6. 29: Speed & Orientation tracking performance.

A very bad steering and acceleration control signals that influence the comfort of the vehicle.



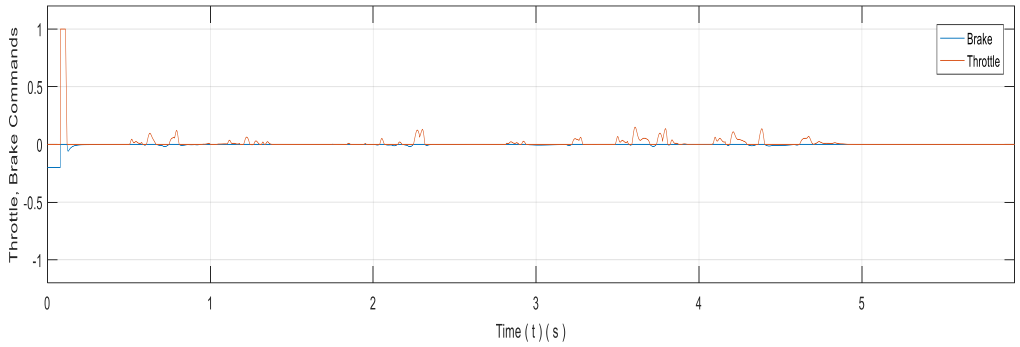


Figure 6. 30: Combined (a) Steering ,(b) Throttle and Brake of NMPC and Feedforwrd.

Conclusion

In this thesis, we have addressed the safety problem in self-driving cars from the planning and control levels individually and combined in double lane change scenario.

Due to the technological advancement, we are now able to integrate a more complex models as (Amesim or Tire model) in developing planning algorithms or designing trajectory controllers. In our Path planner; we have parametrized simple kinodynamic RRT to mimic the human driving action to ensure the execution of path and drive safely, moreover; our path planner like driver combines between the path planning and trajectory generation tasks. For instance this path planner is parametrized for lane change and double lane change scenarios which are common driving behaviors.

In addition to path planning, a trajectory controller was developed to follow the generated path and correct the tracking error during the execution. The developed NMPC tracks perfectly the generated path in double lane change as well in slippery tracks. Feedforward + NMPC is a promising structure that takes the advantage of available control signals from the planner and allow the MPC to compensate the errors. The effectiveness of planner and control approach are simulated with high fidelity model.

The planner with high fidelity model is going to be used as trainer to generate safe trajectories for critical situations. Improvements can be done in (1) Feedforward + NMPC control approach (2) path planning (online, optimal trajectories which is very challenging and experimentally tested.

Bibliography

[1] "Stanford Univesity," [Online]. Available: http://www.formal.stanford.edu/jmc/progress/cars/cars.html.

[2] Ernst Dieter Dickmanns Volker Graefe, "Dynamic monocular machine vision," *Machine vision and applications, vol. 1,* p. 223–240, 1988.

[3] Ernst D Dickmanns, "“Vehicles capable of dynamic vision," *IJCAI ,* p. 1577–1592, 1997.

[4] D. Pomerleau, "Rapidly Adapting Artificial Neural Networks for Autonomous Navigation," in *Advances in Neural Information Processing Systems 3*, Morgan-Kaufmann, 1991, pp. 429--435.

[5] Martin Buehler, Karl Iagnemma, Sanjiv Singh, "The DARPA Urban Challenge: Autonomous Vehicles in City Traffic," Springer-Verlag Berlin Heidelberg, 2009.

[6] "Waymo," 2009. [Online]. Available: https://www.google.com/selfdrivingcar/). .

[7] Jonathan Bohren , "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge," *J. Field Robot,* vol. 25, p. 598–614, 2008.

[8] Julius Ziegler, Moritz Werling, Joachim Schröder, "Navigating car-like Robots in unstructured Environments using an Obstacle sensitive Cost Function," in *IEEE Intelligent Vehicles Symposium*, Eindhoven, The Netherlands, 2008.

[9] M. Montemerlo, "Junior: The Stanford Entry in the Urban Challenge," in *Journal of Field Robotics*, Wiley, 2008.

[10] Sertac Karaman, Emilio Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *CoRR,* pp. 1105-1186, 011.

[11] Lydia E. Kavraki, Jean-Claude Latombe, Mark H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation,* vol. 12, no. 4, p. 566–580, 1996.

[12] Steven M. LaValle, James J. Kuffner, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research,* 2001.

[13] Yoshiaki Kuwata, Justin Teo, Gaston Fiore , "Real-Time Motion Planning With Applications to Autonomous Urban Driving," *IEEE Transactions on Control Systems Technology,* vol. 17, no. 5, pp. 1105-1118, 2009.

[14] Sertac Karaman ; Emilio Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, USA, 2010.

[15] JosÉ E. Naranjo, Carlos Gonzalez, Ricardo Garcia, "Lane-Change Fuzzy Control in Autonomous Vehicles for the Overtaking Maneuver," *IEEE Transactions on Intelligent Transportation Systems,* vol. 9, no. 3, pp. 438 - 450, 2008.

[16] Alessandro De Luca, Giuseppe Oriolo, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*, Springer, 1998, pp. 171-253.

[17] E. Kim , J. Kim, "Model predictive control strategy for smooth path tracking of autonomous vehicles with steering actuator dynamics," *International Journal of Automotive Technology,* vol. 15, no. 7, pp. 1155-1164, 2014.

[18] Paolo Falcone, Francesco Borrelli, Jahan Asgari, "Predictive Active Steering Control for Autonomous Vehicle Systems," *IEEE Transactions on Control Systems Technology ,* vol. 15, no. 3, pp. 566 - 580, 2007.

[19] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, Daniela Rus, "Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model," *IEEE Transactions on Intelligent Transportation Systems ,* vol. PP, no. 99, pp. 1-15, 2017.

[20] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *International journal of robust and nonlinear control,* vol. 18, no. 8, 2007.

[21] Nitin R. Kapania, J. Christian Gerdes, "Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling," *International Journal of Vehicle Mechanics and Mobility,* vol. 53, no. 12, pp. 1744-5159 , 2015.

[22] "Siemens Industry Software," [Online]. Available: https://www.plm.automation.siemens.com/en/products/lms/imagine-. [Accessed 2018].

[23] F. Borrelli , P. Falcone , T. Keviczky , J. Asgari , D. Hrovat, "MPC-Based Approach to Active Steering for Autonomous Vehicle Systems," *International Journal of Vehicle Autonomous Systems ,* vol. 3, no. 2-4, 2005.

[24] Bakker, E., Nyborg, L., Pacejka, H. B, " Tyre modeling for use in vehicle," 1987.

[25] R. Rajamani, Vehicle Dynamics and Control, Springer, 2012.

[26] O. Mikulá, "A Framework for Nonlinear Model Predictive Control," Czech Technical University in Prague, 2016.

[27] "ACADO Toolkit," [Online]. Available: http://acado.github.io/matlab_overview.html. [Accessed 20018].