IMPLEMENTATION OF MOTION PLANNER USING PYTHON AND CARLA SIMULATOR

CONNECTED AUTONOMOUS VEHICLE

PROJECT REPORT

Submitted by:

Adil Iqbal – 110089140
Nidhi Bhavesh Desai -110072124
Sharath Srivathsan Ramesh – 110079675
Ezhil Arasu Padmarani Ravichandran – 110087772

Submitted to:

Prof. Ning Zhang

Table of Contents

Table of Figures

Abstract

Many approaches have been proposed for autonomous driving systems based on high-precision sensors, high computational power, and the creation of numerous networks in both academics and business. An autonomous vehicle’s primary aspect is that it needs to be aware of the surrounding scenarios to formulate and execute an appropriate response. In this project, we implemented a motion planner that considers the input from various sensors to decide the most collision free shortest path from numerous available paths. The report includes an assessment of several suggested strategies and an analysis of their efficacy. The surveyed methodologies differ in the autonomous vehicle model used based on the environmental structure and in computational requirements. The side-by-side comparison provided in this survey supports with system level design decisions and provides insight into the advantages and disadvantages of the studied methodologies.

I. INTRODUCTION

Over the past three decades, research into developing driverless car technology has significantly increased in both academia and business. These advancements have been stimulated by recent innovations in sensing and processing technology, as well as the possible transformational effects on vehicular mobility. In 2014 there were 32, 675 traffic related fatalities, 2.3 million injuries, and 6.1 million reported collisions [1]. Of these, an estimated 94% are attributed to driver error with 31% involving legally intoxicated drivers, and 10% from distracted drivers [1]. The likelihood of driver error and carelessness contributing to vehicle collisions could be drastically reduced with autonomous cars. Finally, for the 86% of the US work force that commutes by car, on average 25 minutes (one way) each day, autonomous vehicles would facilitate more productive use of the transit time, or simply reduce the measurable ill effects of driving stress [1]. Also, these can be eliminated, and the road safety could be improved with the ’Implementation of Motion Planner using Python and Carla simulator’ in a vehicle which is operated without human intervention. Contrary to humans, computers do not struggle with focus while driving. Additionally, an autonomous vehicle can prevent collisions by anticipating and responding to potentially dangerous situations on the road.

Based on a range of sensors placed within the car, autonomous vehicles build and maintain a map of their surroundings. Radar sensors follow surrounding vehicles, while video cameras detect traffic lights, read road signs, track other vehicles, and search for pedestrians. Ultrasonic sensors detect curbs and other vehicles when parking, while Lidar (light detection and ranging) sensors calculate distances, spot road boundaries, and recognize lane markers. Sophisticated software then evaluates all this sensory input, maps a path, and delivers instructions to the car’s actuators, which control acceleration, braking, and steering.

In this project, a demo model is developed with the help of Python programming language and CARLA to simulate the complete working of autonomous vehicles which improves road safety and reduces traffic congestion.

II. HIERARCHICAL MOTION PLANNING

Autonomous vehicles are systems that process a stream of observations from sensors such as radars, LIDARs, cameras, GPS/INS units and odometry [1]. These observations are utilised to automatically choose values for controlled variables guiding the motion of the vehicle, along with prior knowledge of the road network, traffic laws, vehicle dynamics, and sensor models. The hierarchical motion planner is based on the kinematic bicycle model which has the dynamics and kinematics that resembles an ego vehicle in consideration. The motion planning process has three key sub-components, which defines the ego vehicle's prediction, behavior, and trajectory [2].

The mission of self-driving cars is to reach from origin to destination safely following traffic rules and minimizing risk due to other obstacles. However, the task of mission planning is a complex one, which requires environment perception from several sensors and perceiving the ego vehicle's state and its surrounding environment in various scenarios. Therefore, mission planning is broken into a set of optimization problems known as hierarchical structure. The motion planning structure is hierarchically decomposed into four components as shown in Figure. 1.

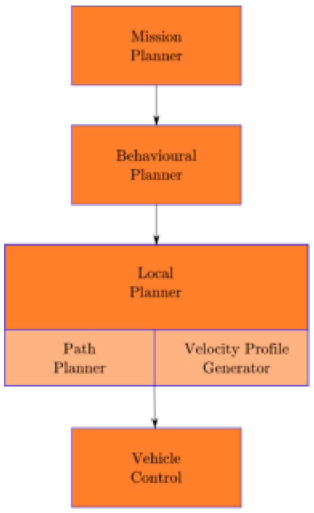


Figure 1. Hierarchical Motion Planning

At the highest level is the mission planner that derives the shortest path from source to destination. Once, the shortest path is derived, the behaviour planner decides maneuvers to be taken by an autonomous vehicle. This is followed by a local planner that selects a continuous collision free path to accomplish a navigational task. The planned motion is then executed with mistakes being reactively fixed by a vehicle control system.

III. MISSION PLANNER

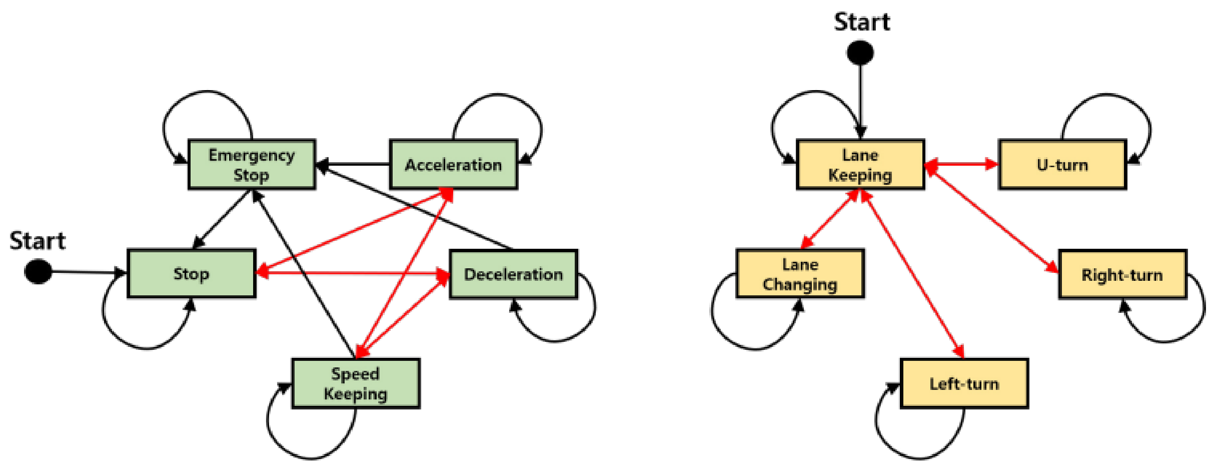
The mission planner is responsible for decomposing driving missions into lane level sub-missions such as lane change, intersection related decisions and many more [3]. This planner computes the shortest path from the source to destination using various algorithms such as Dijkstra's or A*. For each layer of the architecture, the input higher-level mission is decomposed into sub-missions and passed on to the next-lower level. Also, the mission planner will automatically deliver the next set of objectives to the lower layers once the vehicle has completed the current lane-level sub-missions [3].

The mission planner plans a route using the pre-generated maps such as Occupancy grid map and localization map. By representing the road network as a directed graph with edge weights corresponding to the cost of traversing a road segment, such a route can be formulated as the problem of finding a minimum-cost path on a road network graph [1].

IV. BEHVAVIOURAL PLANNER

Since the road geometry was considered in the preceding layer, the behavior planner concentrates on managing on road traffic, including static and dynamic obstacles [3]. The behavioral planner receives a set of constraints and objects from the Mission planner and hence, the goal of this planner is to navigate the selected route following the driving convention and rules of the road [1]. It uses the reference of no traffic, moving obstacles, and all roadblocks as input and generated controller instructions such as lateral driving bias, the preferred leading vehicle, the aggressivity of distance keeping and the maximum speed [3].

Behaviour planner uses the traditional approach known as Finite State Machines to represent the set of rules required for behavior selection of the ego vehicle. The two types of Finite State machines are Control Finite State Machine and Motion Finite State Machine as shown in Figure. 2 [4].



[4]

Figure 2. C-FSM & M-FSM

Each FSM represents individual states which denotes the numerous actions of the autonomous vehicle. The black line denotes a one-way state change whereas red line denoted a two-way state change. Each state machine changes state based on the vehicle's status and information each time [4]. Control Finite State Machine denotes control actions such as stop, emergency stop, acceleration, deceleration, and few more [4]. On the other hand, Motion Finite State Machine denotes various motion actions such as lane keeping, lane changing and directions changing [4].

V. COLLISION CHECKING

Collision checking is the crucial act of making sure an object doesn't collide with any impediments while travelling along a specified trajectory or planned path. A crucial fact to keep in consideration is that collision checking is difficult, computationally demanding problem that appears in numerous fields such as gaming, 3D animation, engineering design, autonomous driving, and other robotic applications [5]. This technique demands perfect information to guarantee safety, needs to be approximated and must be robust to noise. There are two types of collision checking techniques: Swath based collision checking and circle-based collision checking.

A. SWATH BASED COLLISION CHECKING

In swath-based collision checking, it entails rotating and translating a vehicle's footprint along each segment of a predetermined path [5]. The heading at each point rotates and translates each point in the footprint respectively [5]. This collision checking technique computes the union of the path that vehicle has travelled with that that of occupancy grid map and localization map for dynamic objects. However, this requires recursive computation which increases its complexity for larger projects. Swath based techniques are useful for lattice planners, as the swath sets can be computed offline and online collision checking is then simplified using lookup tables [5].

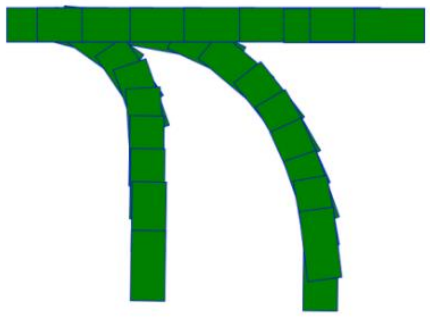
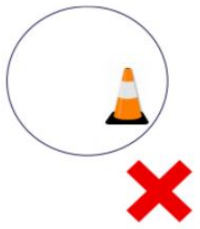


Figure 3. Swath Collision Checking

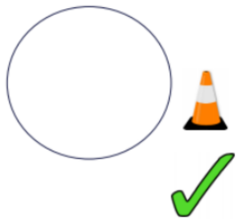
B. CIRCLE BASED COLLISION CHECKING

Circle based collision checking is useful as it is computationally cheap. In this method the ego vehicle and surrounding environment is estimated by forming circles around them. The radius of the circle can be adjusted as per the vehicle's specification and collision requirements. The main task of this technique is to check is the obstacle lies within the circle. If obstacle lies within circle, a collision is detected otherwise it will result into no collision [5]. The positive aspect of circle-based collision checking is it may report a collision that won't occur but never misses a chance to detect the collision that will occur [5].



$$\|(x_i, y_i) - (x_c, y_c)\| \le r$$

Figure 4. Collision Detected



$$\|(x_i, y_i) - (x_c, y_c)\| > r$$

Figure 5. No collision

VI. LOCAL PLANNER

The selected driving behavior by behavioral planner must be translated into a path or trajectory that can be followed by the low-level feedback controller. The resulting course or trajectory needs to be safe for the passenger, dynamically possible for the vehicle, and avoid hitting any impediments that the on-board sensors have detected. The path planner layer is responsible for computing a safe, comfortable, and dynamically feasible trajectory from the vehicle's current configuration to the goal configuration provided by the preceding layer [1]. A collision-free trajectory that complies with dynamic and kinematic limitations on the motion of the vehicle is produced by the local planning component after it receives information about static and moving impediments in the area. Another crucial task of local planner is path trajectory. The two methods for path trajectory are Quintic Splines and Cubical Spirals. Goal points on the way are found by lookahead technique and are calculated based on speed and other factors. For each of our goal states, spiral can be optimized to the goal point. A final collision free path is selected from all the generated spirals.

I. CUBICAL SPIRALS

Cubical Spiral is defined as a polynomial curvature function with respect to length. Since a spiral is a polynomial function of curvature, the curvature value will not change extremely quickly like it can in the case of quintic splines. For Cubical Spirals, closed form solution does not exist, hence integrals need to be evaluated numerically using Simpson's rule as shown in Figure. 6.

$$x(s) = x_0 + \int_0^s \cos(\theta(s'))ds'$$

$$y(s) = y_0 + \int_0^s \sin(\theta(s'))ds'$$

$$\boxed{\int_0^s f(s')ds' \approx \frac{s}{3n}\left(f(0) + 4f\left(\frac{s}{n}\right) + 2f\left(\frac{2s}{n}\right) + \cdots + f(s)\right)}$$

Simpson's Rule

Figure 6. Simpson's Rule

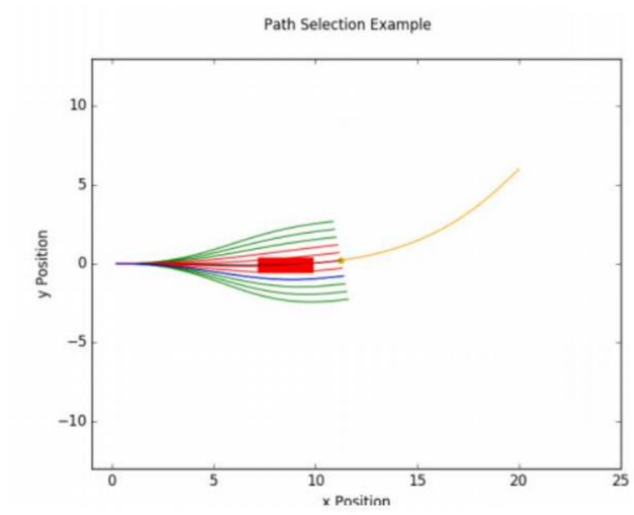


Figure 7. Spirals

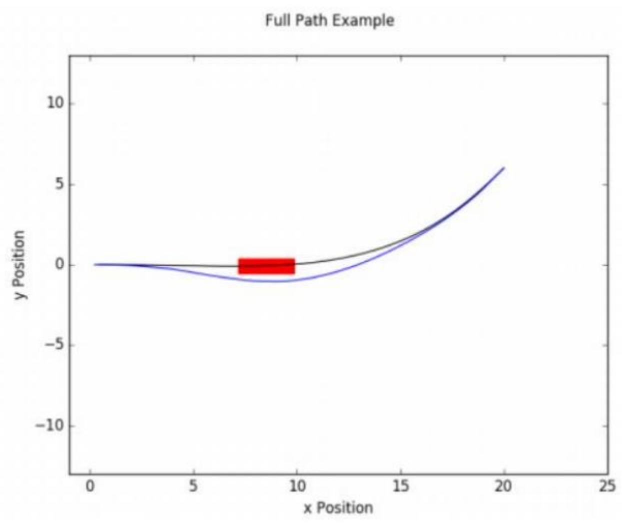


Figure 8. Final Collision free path

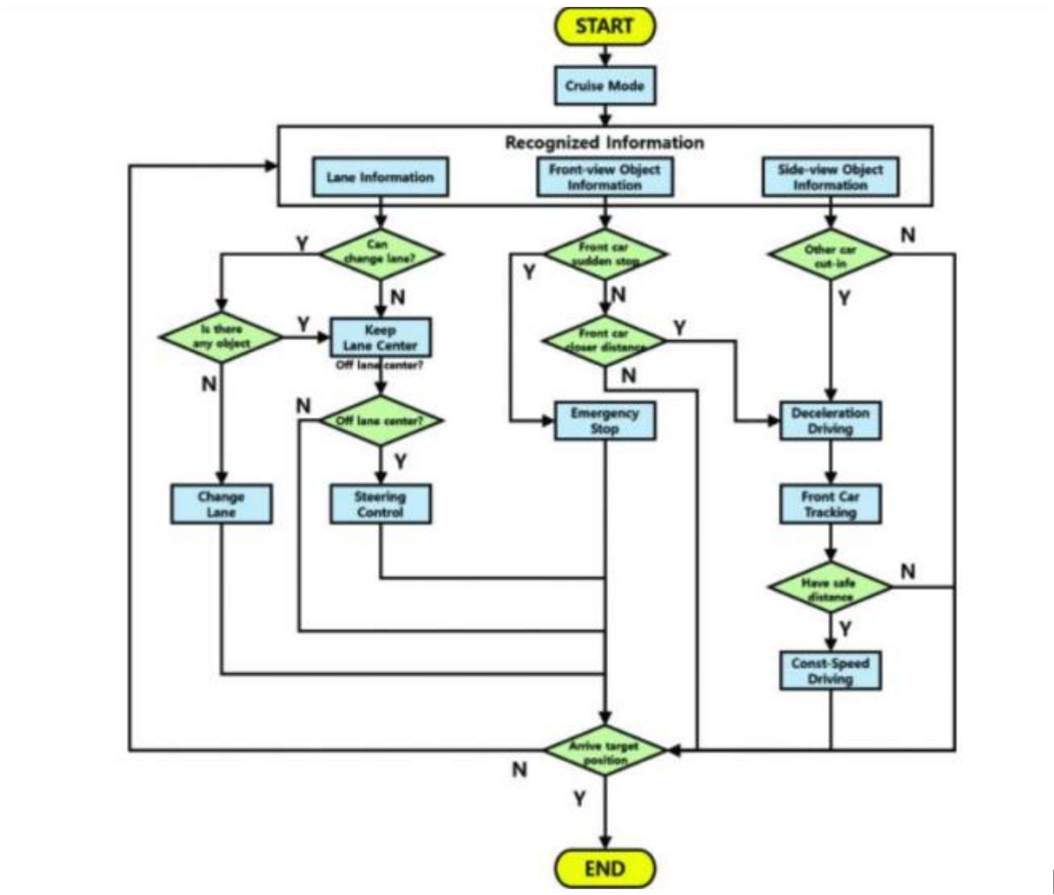
VII. IMPLEMENTATION

A. CARLA SIMPULATOR

The project has been implemented using all the above discussed methodologies in CARLA simulator with the use of Python Programming language. CARLA simulator is an open-source simulator for urban driving. It is developed from the ground up to support training, prototyping, and validation of autonomous vehicles. Environment in the simulator is composed of 3D models of static objects as well as dynamic objects. It supports development, training, and detailed performance analysis of autonomous driving systems.

B. FLOWCHART

The flow of the project and decision making of the autonomous vehicle is based on the below shown flowchart.



[4]

Figure 9. Flowchart

C. SIMULATION RESULTS

In the project simulation, the autonomous vehicles were able to perform the numerous tasks such as waiting at stop signal and red traffic signal, avoiding collision with the parked vehicle, following the lead vehicle & maintaining speed according to the lead vehicle and performed numerous small tasks such as lane keeping, direction changing and turning. During simulation, collision count was 0.

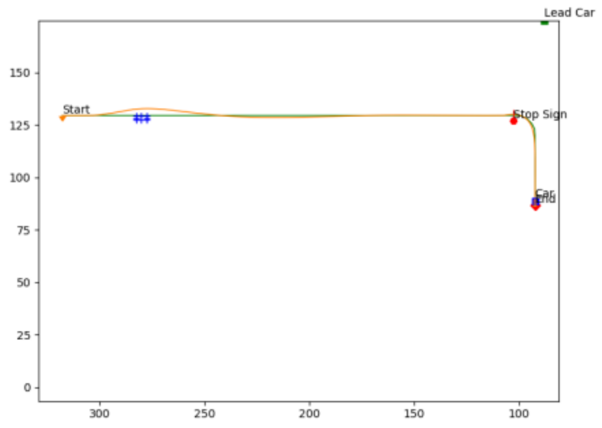


Figure 10. Vehicle Trajectory

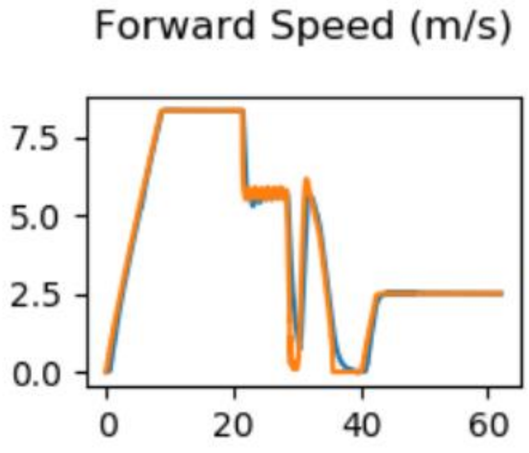
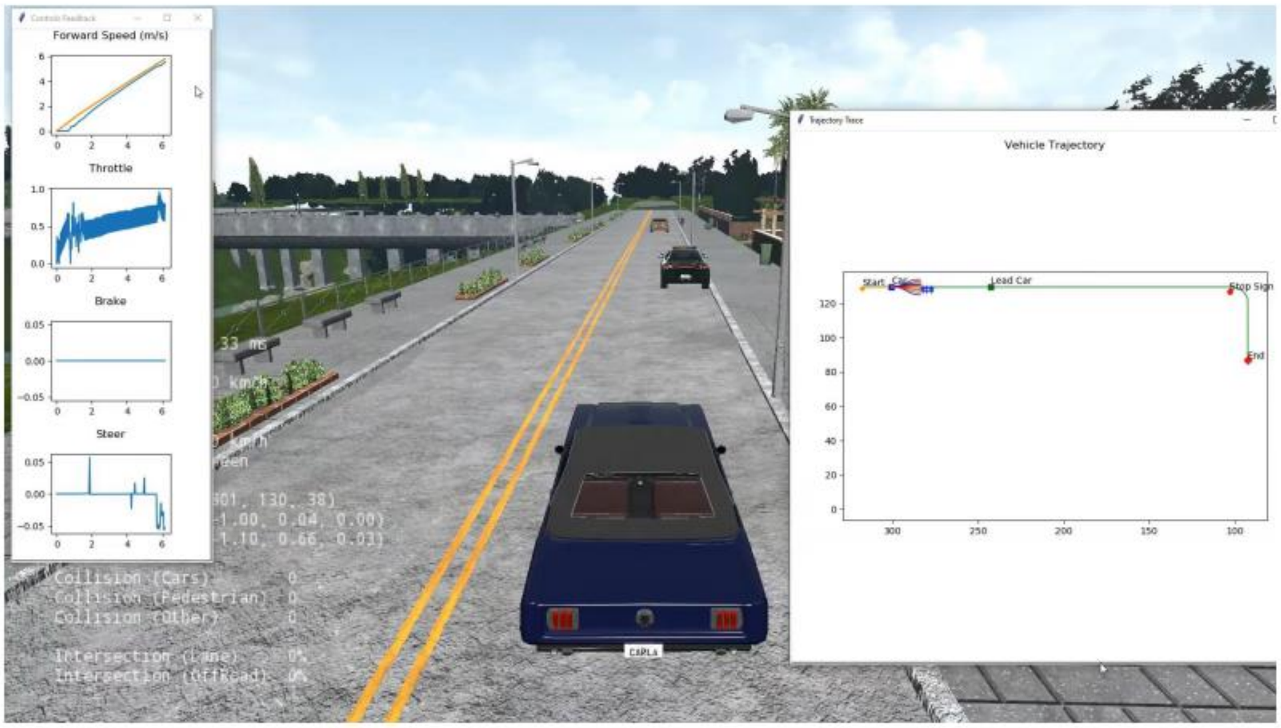


Figure 11. Vehicle Speed



Collision (Cars) 0
Collision (Pedestrian) 0
Collision (Other) 0
Intersection (Lane) 0%
Intersection (OffRoad) 0%

Figure 12. CARLA Simulator

VIII. CONCLUSION

The implementation of a motion planner for Autonomous Vehicles using the tools involved for the project indicated that the Carla Simulator as well as using a high-level programming language such as Python can be used by many automotive manufacturers all over the world to develop their own simulation models for their products. This results in enormous advantages during the testing phase of the production because it gives on point results in a variety of real-world environments. Thus, this step can be an additional step in the production cycle for automotive vehicle design, which can provide increase in efficiency and make the cycle more robust. Furthermore, the project involved usage of various mathematical models for each level of motion planning, and we concluded that the usage of appropriate methods becomes crucial in the design of Autonomous Vehicles. We found that for efficient collision checking, the implementation of Circle-based motion planner is by far a better method than the Swath-based method due to its ease of design and reliability in terms of practicality. Moreover, out of the many methods available to predict the trajectory during Local Planning, Spiral-based planner was found to be fruitful. Hence, it becomes of a crucial importance to select the right models and algorithms to develop an efficient AV. Lastly, determining the velocity profile of AVs is still a branch of study that requires deeper research for the development of robust systems. We concluded that the velocity profile generator using the Carla Simulator gives errors at various points in the mission lane, which can be improved with better software upgrades in the future.

IX. REFERENCES

[1]. B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," in IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33-55, March 2016, doi: 10.1109/TIV.2016.2578706.

[2]. David Silver, "How Path Planning Works for Self-Driving Cars" Available: https://www.linkedin.com/pulse/how-path-planning-works-self-driving-cars-david-silver/

[3]. J. Wei, J. M. Snider, T. Gu, J. M. Dolan and B. Litkouhi, "A behavioral planning framework for autonomous driving," 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 458-464, doi: 10.1109/IVS.2014.6856582.

[4]. S. -H. Bae, S. -H. Joo, J. -W. Pyo, J. -S. Yoon, K. Lee and T. -Y. Kuc, "Finite State Machine based Vehicle System for Autonomous Driving in Urban Environments," 2020 20th International Conference on Control, Automation and Systems (ICCAS), 2020, pp. 1181-1186, doi: 10.23919/ICCAS50221.2020.9268341.

[5]. https://github.com/qiaoxu123/Self-Driving-Cars/blob/master/Part4-Motion_Planning_for_Self-Driving_Cars/Module6-Reactive_Planning_in_Static_Environments/Module6-Reactive_Planning_in_Static_Environments.md

[6]. P. Polack, F. Altché, B. d'Andréa-Novel and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 812-818, doi: 10.1109/IVS.2017.7995816.

[7]. M. McNaughton, C. Urmson, J. M. Dolan and J. -W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 4889-4895, doi: 10.1109/ICRA.2011.5980223.