International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-10 Issue-5, June 2021

# Real - Time Trajectory and Velocity Planning for Autonomous Vehicles

**Hrishikesh Dey, Rithika Ranadive, Abhishek Chaudhari**

***Abstract:*** ***Path planning algorithm integrated with a velocity profile generation-based navigation system is one of the most important aspects of an autonomous driving system. In this paper, a real-time path planning solution to obtain a feasible and collision-free trajectory is proposed for navigating an autonomous car on a virtual highway. This is achieved by designing the navigation algorithm to incorporate a path planner for finding the optimal path, and a velocity planning algorithm for ensuring a safe and comfortable motion along the obtained path. The navigation algorithm was validated on the Unity 3D Highway-Simulated Environment for practical driving while maintaining velocity and acceleration constraints. The autonomous vehicle drives at the maximum specified velocity until interrupted by vehicular traffic, whereas then, the path planner, based on the various constraints provided by the simulator using µWebSockets, decides to either decelerate the vehicle or shift to a more secure lane. Subsequently, a spline-based trajectory generation for this path results in continuous and smooth trajectories. The velocity planner employs an analytical method based on trapezoidal velocity profile to generate velocities for the vehicle traveling along the pre-computed path. To provide smooth control, an s-like trapezoidal profile is considered that uses a cubic spline for generating velocities for the ramp-up and ramp-down portions of the curve. The acceleration and velocity constraints, which are derived from road limitations and physical systems, are explicitly considered. Depending upon these constraints and higher module requirements (e.g., maintaining velocity, and stopping), an appropriate segment of the velocity profile is deployed. The motion profiles for all the use-cases are generated and verified graphically.***

***Keywords: Frenet Coordinate System, Path Planning, Spline Interpolation, Trapezoidal Velocity Curve, Velocity Profile***

## I. INTRODUCTION

Autonomous driving has been predicted to dramatically enhance driving safety, increase transportation efficiency and revolutionize the entire automobile industry, where particularly self-driving cars can offer tremendous benefits to both individuals and societies [1].

Manuscript received on June 20, 2021.
Revised Manuscript received on June 29, 2021.
Manuscript published on June 30, 2021.
***** Correspondence Author
**Hrishikesh Dey*,** Department of Electronics Engineering, VES Institute of Technology, Mumbai (Maharashtra), India. Email: 2017.hrishikesh.dey@ves.ac.in
**Rithika Ranadive,** Department of Electronics Engineering, VES Institute of Technology, Mumbai (Maharashtra), India. Email: 2017.rithika.ranadive@ves.ac.in
**Abhishek Chaudhari,** Department of Electronics Engineering, VES Institute of Technology, Mumbai (Maharashtra), India. Email: abhishek.chaudhari@ves.ac.in



With the help of the on-board technologies, self-driving cars can perform real-time driving tasks without any input from a human operator by creating an image of the surroundings to facilitate traffic navigation [1, 2]. Apart from basic navigation, self-driving vehicles offer increased road safety, and may unclog road traffic due to their ability to communicate with each other, thereby optimizing vehicular routes and thus, resulting in smooth and efficient traffic flow. This, in turn, results in decreased fuel consumption, hence, paving the way for a more sustainable future [3, 4]. For the plethora of advantages these vehicles offer, they have been subjected to intense research, and thus, have become a definite reality that may pave the way for future systems, where robots take over the art of driving [3, 4].

Autonomous driving cars rely on path planning as it is the "brain" of the entire system. Path planning and decision making in urban environments enable self-driving cars to discover the safest, most convenient, and most economical routes from source to destination [5, 6]. This sub-system can reproduce the human thought-process involved while driving-route planning based on the source and destination, real-time analysis of the surroundings and maneuvering any encountered obstacles, and at the same time adhering to traffic rules and maintaining a safe and optimal trajectory [7, 8].

Autonomous on-road driving requires a path planning algorithm that includes the search for a feasible path by taking into consideration the dimensions of the vehicle, the geometry of its surroundings, and the various kinematical path constraints [9, 10]. The vehicle's movement is calculated to fulfill the car's kinematic constraints based on its initial and final dynamic configurations. The vehicle must identify and bypass all the static and maneuverable obstacles to find a collision-free route. The generated real-time trajectories are extracted mathematically considering various path constraints such as restrictions on the size of the vehicle concerning the road, and restrictions on the vehicular drive. Each path generated is then associated with one or many velocity profiles.

Motion planning highlights various critical challenges in the development of autonomous driving systems, including velocity planning. Velocity planning, defined as the variation in velocity along a predefined path for the motion control of autonomous vehicles, becomes very complex due to continuously changing environments [11, 12].

Therefore, the velocity profile needs to be carefully assigned along the generated path. By considering both path planning and motion planning constraints, a velocity profile is employed that can optimize driving capability while adhering to the behavior and velocity limits of the vehicles in a driving scenario, and at the same time avoids any collisions [13]. Autonomous vehicles should be capable of assigning appropriate velocity profiles for a selected optimal path. Velocity planning is a versatile and effective tool for the motion control of autonomous vehicles. The planning optimization algorithm should offer an efficient method for the evaluation of minimum pure-jerk velocity functions. To ensure smooth driving, a velocity planner uses semantic information given by the path planner to establish a continuous velocity profile that considers acceleration and speed thresholds within the selected optimal path (trajectory).

In recent years, a significant amount of work has been dedicated to achieve fast and efficient path and motion planning algorithms, as it remains one of the most important aspect of autonomous driving. Zvi Shiller in [14] achieves a path planning algorithm for vehicles moving on a general terrain and accounts for obstacles, terrain topography, vehicle kinematics and dynamics. The mathematical derivation of the desired torque or velocity signals that act as an input to the respective torque or speed controllers is provided by Martin Adams in [15]. In addition to that, it also defined a trajectory generator which produced constant jerk profiles in acceleration, velocity and displacement, while maintaining vehicular constraints of acceleration and velocity. Tanishtha Nayak in [16], proposed a novel algorithm to find path between final and destination position for an intelligent system, which is considered to be a device/robot having an antenna connected with sensor-detector system. Cheng Chen in [17] devised a solution to counter the problem of trajectory generation by producing a continuous and bounded curvature profile to outline the trajectory, further optimized by the quartic B´ezier curve. In [18], A* path planning algorithm has been represented by Ali Khaleel Mahmood for a mobile robot to be able to follow a constructed path from its current position to a specified goal within its environment. An obstacle detection algorithm has been implemented as a final algorithm which will be used as a part of the whole system to give the robot the ability to move from its initial known position to a specific goal in an optimum way. The autonomous systems rely on several layers of sensor data. However, at the root is an A* search algorithm-based navigation system. P E Teleweck in [19] aims to present an introduction to these algorithms and use the cases where young roboticists can develop path finding/ path planning applications to fit their educational robotics requirements. Jianfang Lian in [20] proposed a cubic spline interpolation-based path planning method to maintain the smoothness of moving the robot's path.

Previous research has attempted to reduce jerks in autonomous vehicles by applying various smooth velocity profiles, such as the s-curve jerk-bounded profile. High-order polynomial motion profiles enable the vehicle to move smoothly. The increase in the order of a polynomial equation results in an equivalent increase in the number of coefficients for the function of time, resulting in the generation of smoother shapes of position and velocity profiles. An algorithm of computational complexity O(n) has been used by L. Consolini in [21], to provide a time-optimal velocity planning technique for autonomous vehicles. Keonyup Chu in [1] described a collision risk procedure to determine target speed by limiting the value of lateral acceleration as maintained by the curvature of the path. Jordi P´erez Talamino and Alberto Sanfeliu's method in [11] imposed initial and final acceleration restrictions to compute the profiles using third order velocity splines in four unknown variables, considering the fifth unknown to be the total time required to travel the given trajectory. Abhiram Rahatgaonkar's thesis on Velocity Planning Approach for Autonomous Vehicles [22] concentrates on velocity planning with familiar environment to avoid collision with moving obstacles. As demonstrated by Xiaohui Li in [23], the velocity profile generation is initiated by the determination of maximum permitted speed by the behavioural planner, the model constructs the trapezoidal speed curve, followed by the application of polynomial splines to ensure continuous acceleration.

This paper provides an overview to safely navigate a car around a virtual highway with other traffic, without going over the limit of 50 MPH. The car should be able to drive the entire 6946 meters length of the gently curved, closed-loop track maintaining speed limit while navigating traffic and without any "incidents" which include: driving over the speed limit, exceeding limits on acceleration and jerk (i.e., the change in acceleration over time, which can make for an uncomfortable and unsafe ride), driving outside the lane, and, of course, colliding with other cars. The simulator provides us with the telemetry data for the ego vehicle (position, heading and velocity) and sensor fusion data for the nearby traced vehicles on the highway (position, velocity). In return, the simulator expects a list of map coordinates from the server, each of which the ego vehicle will obediently follow at intervals of 20 milliseconds. The data obtained as inputs from the simulator, undergoes spline interpolation to generate smooth real-time trajectories. To ensure a smooth ride, the car should not experience total acceleration over 10 m/s2 and jerk that is greater than 10 m/s3. This paper also provides an overview to achieve generation of suitable velocity profiles for an optimal trajectory, explicitly considering various velocity, acceleration, path and time constraints. Considering these constraints help reduce the solution set of the velocity planning, resulting in the planner to focus on the solution space where the optimal solution is more likely to exist. The trapezoidal velocity profile is considered as a reference to generate a stable velocity profile fulfilling various constraints and mode of travel such as velocity-keeping, following, stopping etc. as suggested by higher order behavioral module. Finally, to obtain smooth and jerk free motion, the generated velocity profile is replaced with high order polynomial spline curve maintaining all constraints.



Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

Specifically, the main contributions of this paper are:

- Implementation of a path planner to safely navigate a car around a virtual highway with other traffic, considering various constraints provided by the simulator, maintaining speed limits, not exceeding limits on acceleration and jerk and, of course, bypassing all the static and maneuverable obstacles.
- An approach for accurately generating suitable velocity profiles by employing trapezoidal velocity curve for an optimal trajectory explicitly considering various velocity, acceleration, path and time constraints.
- Demonstration of different jerk minimization techniques involving generation of smooth s-curve motion profiles with low associated accelerations, incorporating high order polynomial spline equations with the suitably generated trapezoidal velocity profile sequences for the optimal trajectory.

The remaining paper is organized as follows. Section II presents the optimized and efficient approach for the development of the path planning algorithm. Section III specifies the techniques involved with formulating the velocity planning problem in detail. Section IV demonstrates the effectiveness of our approach on various simulation profiles and scenarios. Finally, Section V draws the conclusions and suggests future work.

## II. PATH PLANNER

This section on trajectory generation incorporates the design and application of a path planning decision algorithm that creates smooth and safe trajectories to navigate a car around a simulated highway scenario. The path planning algorithm considers various constraints such as the location coordinates, speed and deviation angles of ego vehicle along with the neighboring vehicles in the simulated environment. The program establishes a TCP connection using µWebSockets between the Visual Studio IDE (server) and the Unity 3D Highway-Simulated Environment (client).

The TCP server-client connection supports full-duplex communication thereby allowing the simulator and the server to transmit and receive dynamic information of the ego vehicle and the surrounding vehicles in the simulated environment. Initially, the ego vehicle is expected to follow the base frame provided by the simulator until it encounters an obstacle, wherein in that case, the path planner devises a new trajectory for the car to proceed further. The yaw angle, which is the deviation of the car from the central axis of the highway along with the estimated speed of the vehicles is provided as an input to the path planner. In addition to this, the current location of the ego vehicle and the nearby traced vehicles is also transmitted to the server. The localization data of the ego vehicle and the nearby traced vehicles are expressed in the Cartesian and Frenet coordinate systems.

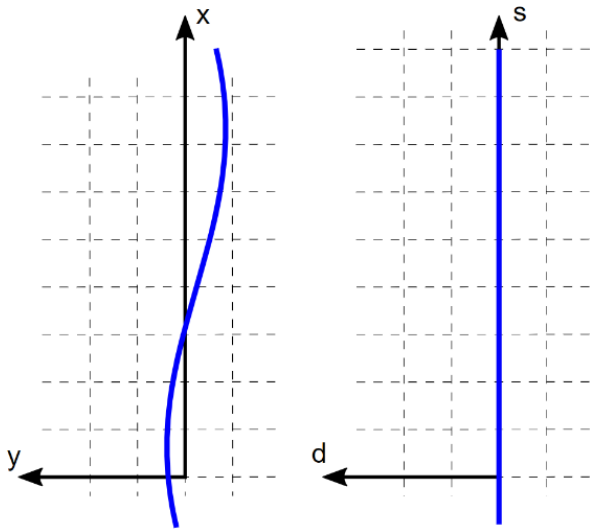



**Fig. 1: Representation of a reference path in Cartesian and Frenet coordinates**

Instead of referring to a fixed axis such as the x and y in the Cartesian Coordinate System, the Frenet system employs the road as a reference as shown in Figure 1, and hence, it continuously changes as per the curvatures of the road. Once the data has been received, spline calculation requires the application of various transforms to generate the trajectory. Therefore, the coordinates are converted from Frenet to the Cartesian system for ease of calculations.

Once the necessary localization and sensor fusion data has been received by the server, the path planner proceeds to analyze the surrounding environment to predict the next path for the vehicle. It initiates with a check for any previous path available. In case, it finds a previous path associated with the ego vehicle, it assigns endpoint coordinates of the previous trajectory as the starting reference for the succeeding one. The LiDAR sensors embedded in the ego vehicle scan up to 30 meters. Hence, 30 meters is assigned as the safe distance to be maintained between any two vehicles at every instance. The safe distance is considered as the limiting factor while scanning for any obstacle vehicles. By considering the safe distance and the predicted location of surrounding vehicles, the sensor fusion data alerts the path planner in anticipating a potential collision and influences in selecting the optimal path during the generation of the trajectory.

As shown in Algorithm 1, the path planner initially checks if there is traffic in the middle lane in which the car is currently

**Algorithm 1** Prediction and Decision-making stage of the Path Planner

```
for every 20 millisecond interval do
    if the current lane is a valid and safe lane then
        Boolean too_close = False
    else
        Boolean too_close = True
        if the right lane is a valid and safe lane then
            Boolean Right Available = True
        else
            Boolean Right Available = False
            if the left lane is a valid and safe lane then
                Boolean Left Available = True
            else
                Boolean Left Available = False
                if too_close = True then
                    if Right Available = True then
                        Change to Right Lane
                    else if Left Available = True then
                        Change to Left Lane
                    else
                        Decelerate by 5 m/s^2
                else
                    Keep Going Straight
```

Travelling, and if the safe distance has not been maintained. In such a situation, it then checks if the right lane is available with no obstacles within 30 meters. If the right lane provides a safe path to proceed, it decides to change to the right lane. Alternatively, if both the conditions are not being satisfied, it checks for the left lane availability with no probable collision with any proximate vehicles. Just like the previous lane checks, if left lane is suitable for travelling, it decides to change to the left lane. However, if the checks fail entirely with no switch possible in any alternate lane, either due to obstacles or the failure in maintaining a safe distance, the vehicle decelerates uniformly to avoid a collision with any other vehicle.

The decision made by the path planner is carefully calculated and executed in the trajectory generation stage. Like the path planning stage, it runs a check for any existing paths available and if found, assigns the end coordinates of the previous path as the starting reference for the next path. Once the start position has been assigned, it sets up a target point at a distance of 30 meters from the initial point. The path of the autonomous vehicle should be smooth to reduce the energy consumption and hence, cubic spline interpolation has been used to achieve this goal. Cubic spline interpolation is used to form a smooth curve through a series of shape points.

Take $(n + 1)$ nodes on the interval $[a, b]$:

$$\mathbf{a < x_0 < x_1 < \ldots < x_n = b}$$

A function $f(x)$ on $[a, b]$ becomes an interpolated cubic spline function if the following conditions are met. In each interval $[x_{i-1}, x_i]$, $f(x)$ is a cubic polynomial function.

$$\mathbf{f_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3}$$

$$\mathbf{f(x_0) = y_0, \ldots, f(x_n{+}1) = y_{n+1}}$$

where $f(x)$ is continuous in the interval $[a, b]$.

This spline polynomial equation consists of unknown variables which requires the substitution of location point coordinates to calculate the value of the variables. For this purpose, we utilize the initial point and the target point. In addition to this, we consider three path nodes situated at equal distances between the initial and the target point. Once these points are substituted in the equation, the Band Matrix Method is used to solve for the unknown variables, helping to generate a smooth and curved profile for the trajectory.

Path planning and trajectory generation are the most vital stages in the development of autonomous driving systems. It requires an accurate prediction of the behaviour of the ego vehicle as well as the surrounding obstacles. The server and the client handle the communication of dynamic data and alert the path planner of any approaching obstacles. Considering the safe distance constraint along with the lane availabilities, the planner devises a suitable solution to be followed by the car to avoid any collisions with its surroundings. Once the decision-making has been accomplished, the planner executes various mathematical formulations to provide a smooth and continuous trajectory to be followed by the ego vehicle. The entire process repeats throughout the course of the path thereby maintaining and adhering to all possible safety constraints.

## III. VELOCITY PROFILE FORMULATION

We designed and implemented an efficient real-time approach for velocity planning of a known optimal trajectory within specified constraints as shown in Algorithm 2. The required velocity and acceleration constraints can be modified such that the trapezoidal velocity profile adapts to a linear or a ramp velocity profile.

**Assumptions.** Considering the velocity (initial velocity $V_o$, maximum velocity $V_m$ and end velocity $V_e$) and acceleration (initial acceleration $A$ and deceleration $D$) constraints, a velocity profile can be designed for a selected optimal trajectory (Total path length $S$).

**Algorithm Phase 1.** The initial procedure involves the generation of a speed curve by employing the trapezoidal velocity profile. Instead of generating the velocity profile for the entire path length $S$, we divide the trajectory into several minuscule paths of length $s$ to obtain the best fitting curve as illustrated in Figure 2. The instantaneous changes in velocity demand the constraints to differ for every minuscule path.

**Algorithm Phase 2.** The initial velocity $V_o$ for the first segment is established in advance by the planner. The maximum velocity $V_m$ remains constant for every set of minuscule paths and restricts the vehicle from exceeding the range. We introduce a new constraint, the time limiting constant, $T_k$, which is the least time for which the vehicle is expected to travel at a constant maximum velocity to avoid collisions and maintain stability. The final velocity $V_e$ is dynamically dependent on the velocity of the leading vehicle to ensure collision free driving. The consecutive minuscule path segments obtain the final velocity $V_e$ of the previous set and set it as its initial velocity $V_o$, while the rest of the constraints remain the same.

*Retrieval Number: 100.1/ijeat.E28800610521*
*DOI:10.35940/ijeat.E2880.0610521*
*Journal Website: www.ijeat.org*



*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication*

**Algorithm 2** Generating Motion Profiles for an Optimal Trajectory

```
for each optimal trajectory do
    divide the trajectory into several minuscule paths
    for each minuscule path do
        consider various velocity, acceleration and path constraints
        consider time limiting constraint Tk for constant velocity region
        generate the trapezoidal velocity profile
        for each generated velocity profile do
            calculate the time required using newton-raphson method
            use the above estimated time for analysis of high-order polynomials
            generate smooth s-curve motion, position and acceleration profile
    Result: trapezoidal velocity profile, third-order s-curve motion profile
```

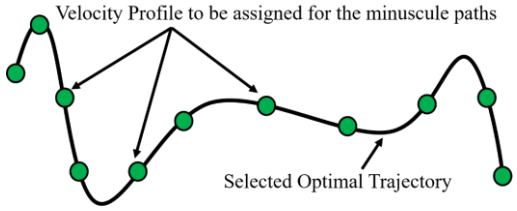
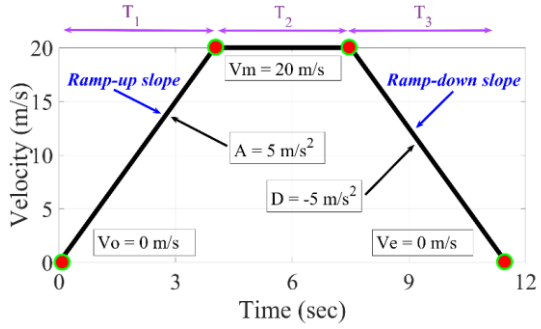
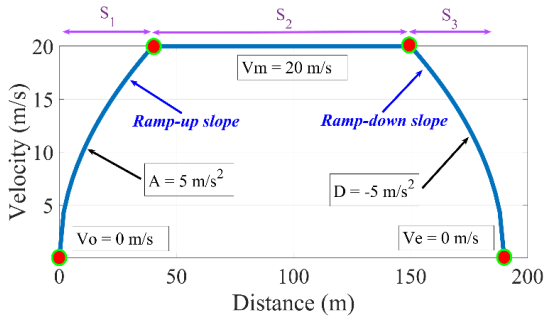



**Fig. 2: The division of selected optimal trajectory into minuscule paths for assigning velocity profile.**




**Velocity v/s Time**




**Velocity v/s Distance**

**Fig. 3: Trapezoidal velocity profiles showing different segments.**

As shown in Figure 3, the trapezoidal motion profile is applied for the velocity profile generation. We assume the ramp-up and ramp-down profiles with constant acceleration and deceleration values to attain a solution. The velocity profile consists of three sections:

1. The initial ramp-up slope $T_1$ / $S_1$ from the initial velocity $V_o$ till the maximum velocity $V_m$
2. Constant traversal segment $T_2$ / $S_2$ at the maximum velocity $V_m$
3. The ramp-down slope $T_3$ / $S_3$ from the maximum velocity $V_m$ till the final velocity $V_e$

The equations for distance covered by the three sections $S_1$, $S_2$ and $S_3$ have been formulated using Newton's Laws of Motion.

$$S_1 = \frac{Vm^2 - Vo^2}{2 \times A} \quad (1)$$

$$S_2 = V_m \times t_{s2} \quad (2)$$

$$S_3 = \frac{Ve^2 - Vm^2}{2 \times D} \quad (3)$$

$$s = S_1 + S_2 + S_3 \quad (4)$$

$$t_{S2} = \frac{s - (S_1 + S_3)}{V_m} \quad (5)$$

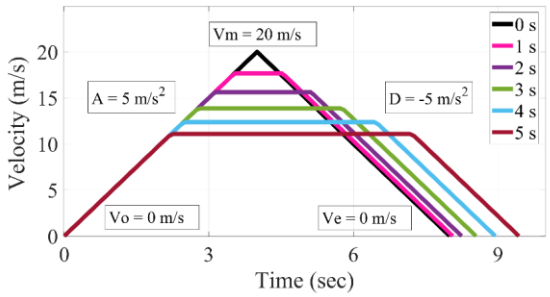



**Fig. 4: Trapezoidal Velocity Profile considering Tk to vary from 0 to 5 seconds.**

$$newV_m = \frac{-T_k + \sqrt{T_k^2 - 4 \times \left(\left(\frac{1}{2 \times A}\right) - \left(\frac{1}{2 \times D}\right)\right) \times \left(\left(\frac{Ve^2}{2 \times D}\right) - \left(\frac{Vo^2}{2 \times A}\right) - s\right)}}{2 \times \left(\left(\frac{1}{2 \times A}\right) - \left(\frac{1}{2 \times D}\right)\right)} \quad (6)$$

The total distance of the minuscule path s in Equation 4 for which the velocity profile has been generated, is equal to the sum of the distance covered by three segments of the profile as specified in Equations 1, 2 and 3. Considering the total path length to remain constant, and acceleration and deceleration to be A and D respectively, the time tS2 for traveling the segment S2 is estimated in Equation 5.

The velocity profile generated in Figure 4 is a special case as it does not trace zero acceleration or a constant velocity motion. The abrupt change from acceleration to deceleration results in a jerky motion. Thus, to ensure a smooth motion, a new value of maximum velocity, labeled newVm, specified in Equation 6, replaces Vm while plotting the velocity curve. For this purpose, we have defined a constant Tk, which serves as a time limiting constraint and fulfils the need to generate a non-jerky and smooth velocity profile for all sets of inputs. The constant, Tk, is the least time for which the vehicle is expected to travel at a constant velocity Vm to avoid collisions and maintain stability. Figure 4 shows that the value of Tk when set to zero, results in no alteration in the value of Vm as time tS2 evaluated for segment S2 is equal or greater than the value of Tk. However, when Tk is set to a higher value, then the value of the maximum velocity Vm alters itself to a lower value of newVm to adhere to the vehicle constraints.

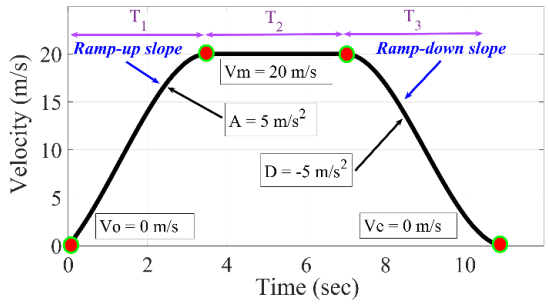
**Algorithm Phase 3.** The trapezoidal velocity profile is further smoothened to guarantee the continuity of the acceleration and optimize comfort by providing minimum jerk. Parameterization of the velocity profile is achieved by interpolating cubic polynomials.

$$v(t) = v_0 + at + bt^2 + ct^3 \quad (7)$$

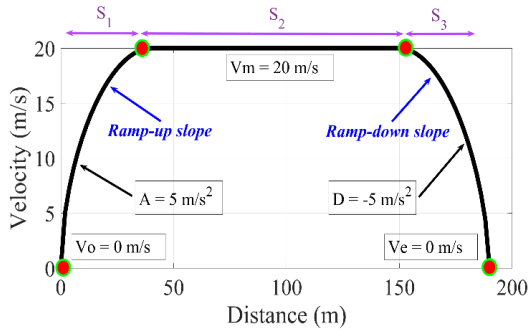
$$acc(0) = a = a_1 \quad (8)$$

$$v(t_f) = v_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 = v_f \quad (9)$$

$$acc(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 = a_f \quad (10)$$




**Velocity v/s Time**




**Velocity v/s Distance**

**Fig. 5: Cubic s-curve velocity profiles showing different segments.**

$$s(t_f) = v_0 t_f + \frac{a_1 tf^2}{2} + \frac{a_2 tf^3}{3} + \frac{a_3 tf^4}{4} = s_f \quad (11)$$

$$\left(\frac{a_0 - a_f}{12}\right) t_f^2 + \left(\frac{v_f + v_0}{2}\right) t_f - s_f = 0 \quad (12)$$

$$a_2 = 3\left(\frac{v_f - v_0}{tf^2}\right) - \left(\frac{2a_0 + a_f}{t_f}\right) \quad (13)$$

$$a_3 = \left(\frac{a_f - a_0}{3\,tf^2}\right) - \left(\frac{2\,a_2}{3\,t_f}\right) \quad (14)$$

The cubic expression for velocity $v(t)$ at any given time $t$ is defined in Equation 7. To smoothen these segments in the s-curve, the total time $t_f$ required by each segment $S_1$ and $S_3$, is calculated. The equation for path length at $t_f$, ie. $s\ (t_f)$ is derived in Equation 11, by integrating the velocity expression in Equation 9. Given the initial velocity $v_0$, initial acceleration $a_0$, terminal velocity $v_f$, final acceleration $a_f$, the unknown parameters $\{a_1, a_2, a_3, t_f\}$ in Equation 11 could be analytically solved with the help of Equations 7, 8, 9, 10 and 11. Newton–Raphson method, a root-finding algorithm that produces successively better approximations to the roots (or zeroes) of a real-valued function is used for calculating the value of $t_f$. The value of $t_f$ is evaluated at every point on the trajectory, and the segments, $S_1$ and $S_3$ are smoothened separately for all values of $t$ from $0 - t_f$ and $v(t)$ has to be calculated to obtain the velocity v/s time s-curve profile. As shown in Figure 5, the S-shaped ramp-up and ramp-down velocity profiles are solved individually, and trajectories are generated by incorporating the spatial path with the corresponding velocity profiles.

## IV. RESULTS AND DISCUSSION

### A. Experimental Setup

For simulating and verifying results and experimental evaluation, the entire programming code was developed using Integrated Development Environment (IDE) i.e., Visual Studio with the help of Object-Oriented Programming Methodology in C++ programming language.

A real-time path planner was developed in C++ to navigate a car around a simulated highway scenario, considering traffic, base frame waypoints, localization data of the car, and sensor fusion data as inputs, and generating the desired smooth and safe real-time trajectories as output. The developed path planner was implemented and validated for results using the Unity 3D Highway-Simulated Environment. A TCP (Transmission Control Protocol) server-client connection was established using µWebSockets for navigating the car around a simulated highway scenario.

In addition to this, a real-time velocity planning algorithm was developed in C++ to generate velocity profiles for a vehicle travelling along a predefined path. The plots of various motion profiles were generated and verified using MATLAB. To generate various motion profiles, the final deceleration i.e., acceleration at end velocity $V_e$ has been considered as 0 m/s$^2$. The limits of the initial acceleration $A$ and the initial deceleration $D$ are based on the vehicle, and traffic rules which are pre-decided and taken as 5 m/s$^2$ and – 5 m/s$^2$, respectively. The entire programming code has been developed and structured exclusively, taking velocity, acceleration, time, and path constraints as inputs, and generating the desired velocity profile as output.

### B. Path Planning Algorithm

The trajectory generated by the path planning algorithm is safe and smooth, assisting the car in avoiding collisions with other vehicles in the environment. It maintains lane constraints and drives according to the 50 MPH speed limit. The path planner controls the car perfectly by directing the ego vehicle through every map coordinate, at intervals of 20 milliseconds. The vector pointing from one coordinate towards the other, dictates the yaw angle of the car. Acceleration, both in the tangential and normal directions, is measured along with the jerk. The vehicle should not exceed a total acceleration of 10 m/s$^2$, and the jerk should not surpass 10 m/s$^3$. The Cartesian coordinates $(x, y)$ are expressed in meters and when calculated along with the specified time, help in determining the speed of the car.

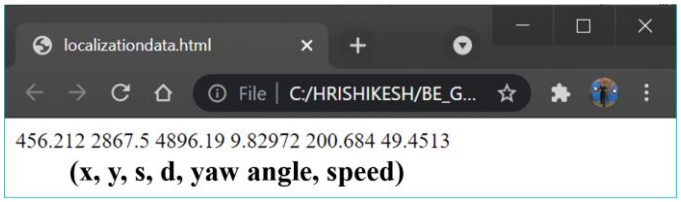



**Fig. 6: Localization data of the ego vehicle.**

*Retrieval Number: 100.1/ijeat.E28800610521*
*DOI:10.35940/ijeat.E2880.0610521*
*Journal Website: www.ijeat.org*



*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication*

Each map waypoint contains values such as [x,y,s,dx,dy]. x and y are the Cartesian coordinates, whereas s represents the distance along the road in Frenet coordinate system. The dx and dy values define the unit normal vector pointing outwards from the highway loop. The waypoints along the highway comprise of a loop, tracing values for the s parameter of the Frenet system from 0 to 6946 meters. As shown in Figure 6, the Localization data of the ego vehicle includes x and y coordinates in the Cartesian system, s and d coordinates in the Frenet system, yaw angle in degrees and speed in m/s. The localization data is responsible for precisely understanding the position of the ego vehicle on the map.

The simulator transmits telemetry information about surrounding cars in the highway, assisting the ego vehicle to avoid obstacles and drive maintaining speed limits. The ego vehicle is equipped with a range of sensors (Lidar, Radar and Camera) whose outputs are fused to produce more accurate measurements. The simulator provides Sensor Fusion data which involves the position and velocity of surrounding obstacle vehicles detected in the sensors' range. As shown in Figure 7, the Sensor Fusion Data is a 2D vector of the neighboring cars in the highway containing the information of vehicle's unique ID, x and y coordinates in the Cartesian system, s and d coordinates in the Frenet system and speed in m/s. The distance between the ego vehicle and other obstacle vehicles on the highway is computed by using localization data of the ego vehicle and sensor fusion data of the nearby traced vehicles.

In the absence of any traffic, the ego vehicle safely drives along the reference path at maximum velocity i.e., 50 MPH. If the LiDAR sensors embedded in the ego vehicle sense any obstacle within the safe distance of 30 meters, the path planner anticipates it as a potential collision, thereby influencing it in making a decision. The path planner checks for the availability and validity of alternate lanes on the highway for a safer path to proceed. However, if none of the other lanes is suitable and secure for travelling, the ego vehicle decelerates uniformly to avoid a collision with any other vehicle. The commands when the ego vehicle has changed to the right lane and a snapshot of the ego vehicle making a change to the right lane is shown in Figure 8 and 9, respectively.

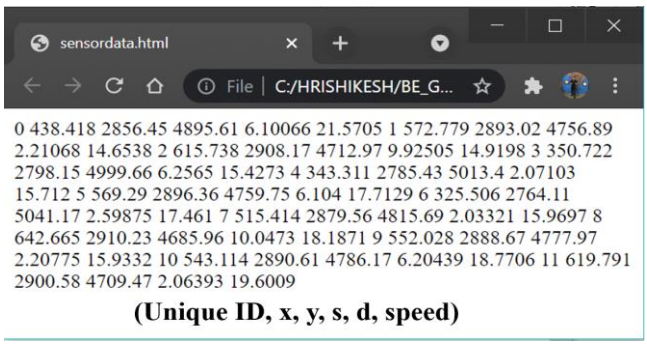



**Fig. 7: Sensor Fusion Data (Nearby Traced Vehicles Data)**

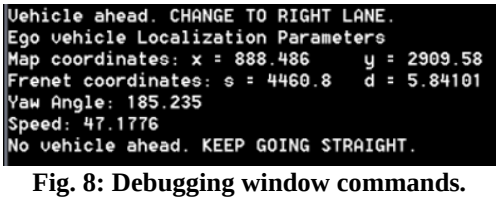



**Fig. 8: Debugging window commands.**

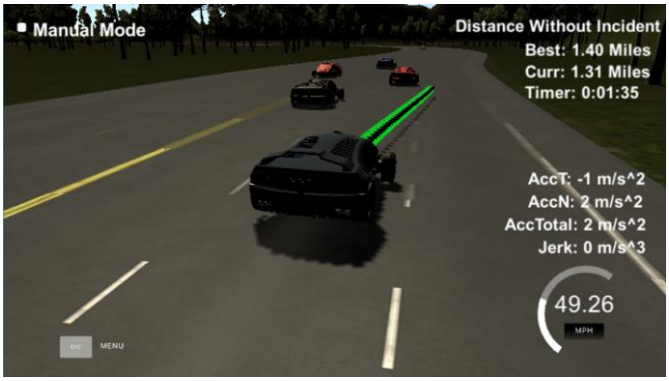



**Fig. 9: Unity 3D Highway-Simulated Environment.**

The autonomous vehicle is able to complete the 6946m highway loop without any collisions with neighboring obstacles by executing appropriate decisions. It also maintains the speed constraints of the road with gentle acceleration and minimum jerk. Also, the car is able to overtake the slower vehicles when there are suitable opportunities, for example, when the distance between two vehicles is greater than 30m and alternate lanes are available for permitting a lane change.

### C. Generating various Velocity Profiles

For optimizing and minimizing the travel time, velocity profile must be assigned carefully on each point along the trajectory. The total length of the given trajectory $S$ was divided into several minuscule paths each having a distance $s$ for which the velocity profiles were generated, considering various path constraints. The path constraints such as the maximum allowed speed $V_m$, the acceleration (acceleration $A$ and deceleration $D$), the velocity (initial velocity $V_o$ and end velocity $V_e$) and the total distance of the minuscule path $s$ dictated by the path were compulsorily fulfilled to obtain the desired velocity profile. Depending upon the applied constraints, the trapezoidal, linear, or ramp velocity profiles were generated.

To verify the different types of motion profiles, velocity v/s time profiles are obtained. In this profile, the velocity for every given point (time) $t$, lying on the total time taken to cover the minuscule path $T$, has been calculated. The path constraint $s$ is considered as *150 meters*. The distance, time, velocity, and acceleration equations required for the generation of various types of velocity profiles have been formulated using Newton's Laws of motion.

$$T_1 = \frac{V_m - V_0}{A} \quad (15)$$

$$T_2 = ts_2 \quad (16)$$

$$T_3 = \frac{V_e - V_m}{D} \quad (17)$$


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication

**Table- I: Velocity and Time constraints considered for generating various types of Velocity Profiles.**

| Velocity Profile | Velocity Constraints | | | Time Constraint |
|---|---|---|---|---|
| | $V_o$ m/s | $V_m$ m/s | $V_e$ m/s | $T_k$ s |
| Trapezoidal | 0 | 20 | 0 | 0 |
| Linear | 20 | 20 | 5 | 0 |
| Ramp | 0 | 20 | 20 | 0 |
| Trapezoidal (no $S_2$ Region) | 0 | 20 | 0 | 0 |
| Trapezoidal (new $V_{stable}$) | 0 | 20 | 0 | 6 |

$$V_1 = V_0 + (A \times t) \quad (18)$$

$$V_2 = V_m \quad (19)$$

$$V_3 = V_m + (D \times (t - (T_1 + T_2))) \quad (20)$$

Using Equations 15, 16 and 17, the time taken to cover each segment of the velocity v/s time profile can be obtained. The total time to travel the minuscule path $s$ is equal to the sum of the time taken by each individual segment of the profile. The velocity at any given point (time) $t$ lying between $0$ *to* $T_1$, $T_1$ *to* $T_2$ *and* $T_2$ *to* $T_3$ on various velocity curves in a velocity v/s time profile, can be calculated using Equations 18, 19 and 20, respectively. The generated trapezoidal, linear and ramp velocity profiles in Figures 10, 11 and 12, respectively, are obtained by considering the velocity and time constraints as specified in Table I. Table II presents the time taken by each segment of the various generated smooth and non- smooth velocity v/s time profiles to the considered velocity, acceleration, path, and time constraints.

Following the generation of various velocity profiles is the curve smoothening, which is necessary to provide smooth control and jerk minimization. The time duration required to cover the ramp-up and ramp-down segments in the s-curve profile differ from those obtained in the trapezoidal curve profile. The total time $t_f$, required to cover each segment in an s-curve profile, is calculated to produce a continuous s-curve. In this approach, the ramp-up and ramp-down segments are smoothened individually using cubic polynomials for all values

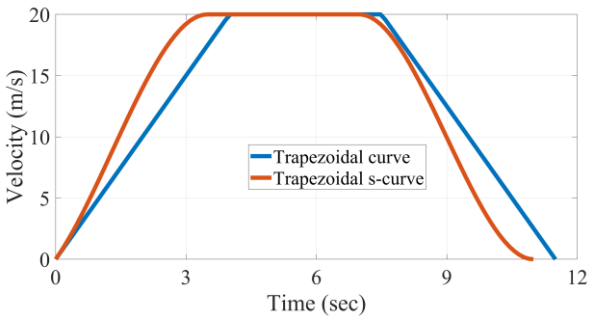



**Fig. 10: Non-smooth and smooth s-curve trapezoidal velocity profile.**

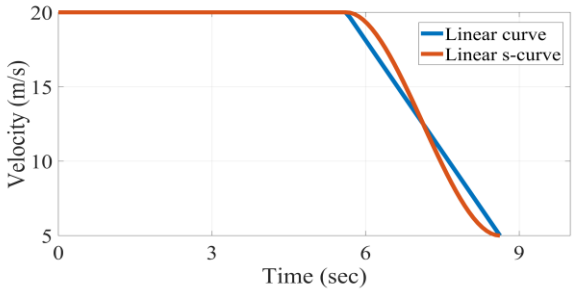



**Fig. 11: Non-smooth and smooth s-curve linear velocity profile.**

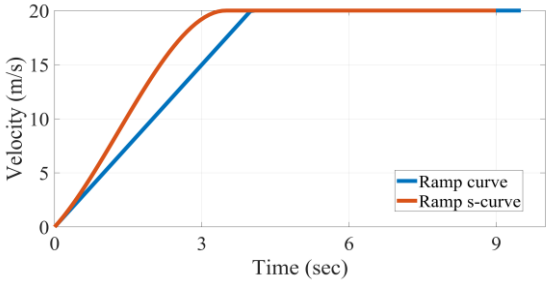



**Fig. 12: Non-smooth and smooth s-curve ramp velocity profile.**

of $t$ ranging from $0 - t_f$ in the velocity v/s time s-curve profile. The generated non-smooth trapezoidal, linear and ramp velocity profile in Figures 10, 11 and 12, respectively, are adopted to smoothen the curve using spline equations considering the velocity and time constraints listed in Table I.

**Table- II: Time taken to cover each segment of various Non-Smooth and Smooth S-curve Velocity Profiles.**

| Velocity Profile | Non-smooth curve | | | | Smooth s-curve | | | |
|---|---|---|---|---|---|---|---|---|
| | $T_1$ s | $T_2$ s | $T_3$ s | Total $T$ s | $T_1$ s | $T_2$ s | $T_3$ s | Total $T$ s |
| Trapezoidal | 4 | 3.5 | 4 | 11.5 | 3.745 | 3.5 | 3.745 | 10.99 |
| Linear | 0 | 5.625 | 3 | 8.625 | 0 | 5.625 | 3 | 8.625 |
| Ramp | 4 | 5.5 | 0 | 9.5 | 3.49 | 5.5 | 0 | 8.99 |
| Trapezoidal (no $S_2$ Region) | 4 | 0 | 4 | 8 | 3.745 | 0 | 3.745 | 7.49 |
| Trapezoidal (new $V_{stable}$) | 2 | 6 | 2 | 10 | 1.87 | 6 | 1.87 | 9.74 |

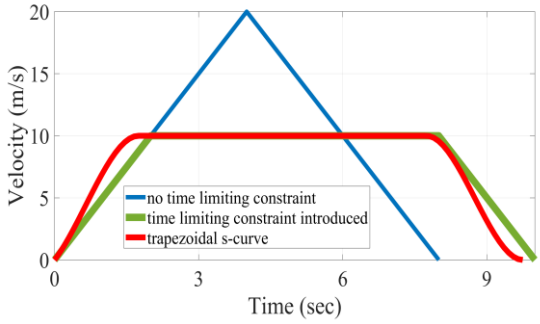



**Fig. 13: Generation of a Trapezoidal Velocity profile using $newV_m$ on introduction of a time limiting constant, $T_k$.**

In case the time required to travel segment $S_2$ i.e., $t_{S2}$ turns out to be lesser than the time limiting constraint $T_k$, a new value of maximum velocity $newV_m$ is calculated, considering the given velocity, acceleration, time, and path constraints. Figure 13 demonstrates the purpose of the time limiting constraint, $T_k$. The simulation of the profiles considers the path constraint, i.e., the total length of the minuscule path *s*, to be 80m. The profile generated by defining constraint, $T_k$, equal to 0 seconds, is observed to be extremely unstable due to an abrupt change from an accelerating to a decelerating region, resulting in a jerky motion. Hence, a trapezoidal curve is fitted for the given distance by calculating the new maximum velocity $newV_m$, lesser than the original, considering the time limiting constant, $T_k$. When the value of $T_k$ is considered to be 6 seconds, the value of $newV_m$ is estimated to be 10m/s for generating the trapezoidal profile. The trapezoidal curve is then smoothened using polynomial splines to generate the respective s-curve.

The above section deals with the plotting of smooth and continuous velocity profile curves within the applied constraints. The ability of jerk avoidance for the system is furnished by $T_k$ by altering the maximum velocity to a lower value for maintaining stability. This ensures that the vehicle travels at the maximum velocity for the longest time possible thereby reducing the duration of travel to a minimum. The generation of an acceleration continuous s-curve profile further contributes towards greater comfort by avoiding abrupt changes in the values of velocity and acceleration. The final acceleration, when assumed to be 0 $m/s^2$, ensures that the vehicle maintains a constant final velocity $V_e$. Amongst the three types of curves obtained, the trapezoidal velocity curve is the most applicable as it guarantees the employment of the segment $S_2$ which maintains the vehicle at constant maximum velocity. Consequently, the interpolation of cubic spline curve is suitable for the velocity profile algorithm as it acquires an edge over the non-smooth curve both in terms of minimum time travel and improved stability.

## V. CONCLUSION

The proposed method in this paper was able to achieve smooth and safe trajectory planning along with the incorporation of a velocity planning algorithm. The path planner programmed in the C++ language, successfully executed suitable decisions, and implemented them to generate efficient real-time trajectories to be followed by the autonomous vehicle while avoiding obstacles in a simulated highway scenario. Various constraints and specifications such as the speed limits, localization, and sensor fusion data were provided as inputs to the planner and significantly accurate results were observed. On detection of traffic, the self-driving car senses the surrounding environment and, on reviewing the behavior of neighboring vehicles on the highway, executes a decision to either decelerate or switch to a safer lane. The generated path is acceptably tolerable by the passenger as it maintains speed limits and minimizes the jerk.

The velocity profile algorithm was developed and implemented to generate an appropriate velocity profile for an optimal trajectory. After the selection of the optimal trajectory, the initial, maximum, and final velocities along with acceleration values assist to trace a perfect velocity curve along the path. The profiles were obtained by plotting a curve for the minimum time of travel for the vehicle through the specified path within given acceleration limits. The velocity profile ensures safety, efficiency thereby providing a non-jerky and comfortable motion to self-driving vehicles. Once tested on a broader range of urban traffic scenarios, the proposed methods promise to interact and fit well with the existing systems and evolve with the entire autonomous driving system.

## REFERENCES

1. K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1599–1616, 2012.
2. C. L. Bianco, A. Piazzi, and M. Romano, "Velocity planning for autonomous vehicles," in IEEE Intelligent Vehicles Symposium, 2004. IEEE, 2004, pp. 413–418.
3. S. Thrun, "Toward robotic cars," Communications of the ACM, vol. 53, no. 4, pp. 99–106, 2010.
4. L. D. Burns, "A vision of our transport future," Nature, vol. 497, no. 7448, pp. 181–182, 2013.
5. J. Kim, K. Jo, D. Kim, K. Chu, and M. Sunwoo, "Behavior and path planning algorithm of autonomous vehicle a1 in structured environments," IFAC Proceedings Volumes, vol. 46, no. 10, pp. 36–41, 2013.
6. V. T. Minh and J. Pumwa, "Feasible path planning for autonomous vehicles," Mathematical Problems in Engineering, vol. 2014, 2014.
7. J. Villagra, V. Milanes, J. P. Rastelli, J. Godoy, and E. Onieva, "Path and ´ speed planning for smooth autonomous navigation," in IV 2012 - IEEE Intelligent Vehicles Symposium, Jun 2012, Alcala de Henares, Madrid, Spain, 2012.
8. L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 5, pp. 1826–1848, 2019.
9. F. Altche, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2017, pp. 1–7.
10. T. Gu and J. M. Dolan, "On-road motion planning for autonomous vehicles," in International Conference on Intelligent Robotics and Applications. Springer, 2012, pp. 588–597.
11. J. P. Talamino and A. Sanfeliu, "Anticipatory kinodynamic motion planner for computing the best path and velocity trajectory in autonomous driving," Robotics and autonomous systems, vol. 114, pp. 93–105, 2019.
12. Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in Proceedings., IEEE International Conference on Robotics and Automation. IEEE, 1990, pp. 384–389.

13. R. Solea and U. Nunes, “Trajectory planning with velocity planner for fully-automated passenger vehicles,” in 2006 IEEE Intelligent Transportation Systems Conference. IEEE, 2006, pp. 474–480.
14. Y.-R. G. Zvi Shiller, “Dynamic motion planning of autonomous vehices,” IEEE Transactions on Robotics and Automation, vol. 7, pp. 241–249, 1991.
15. M. Adams and J. Ibanez-Guzman, “Limiting velocity & acceleration commands for dynamic control of a large vehicle,” in 7th International Conference on Control, Automation, Robotics and Vision, 2002. ICARCV 2002., vol. 3. IEEE, 2002, pp. 1475–1480.
16. Dash, Tirtharaj, Goutam Mishra, and Tanistha Nayak, “A novel approach for intelligent robot path planning,” arXiv preprint arXiv:1306.4672 (2013).
17. C. Chen, Y. He, C. Bu, J. Han, and X. Zhang, “Quartic bezier curve-based trajectory generation for autonomous vehicles with curvature and velocity constraints,” in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 6108–6113.
18. Mahmood, Ali Khaleel, and Robert Bicker, “Path Planning, Motion Control and Obstacle Detection of Indoor Mobile Robot,” American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS) 26, no. 1 (2016): 91-107.
19. Teleweck, P. E., and B. Chandrasekaran, “Path Planning Algorithms and Their Use in Robotic Navigation Systems,” in Journal of Physics: Conference Series, vol. 1207, no. 1, p. 012018. IOP Publishing, 2019.
20. Lian, Jianfang, Wentao Yu, Kui Xiao, and Weirong Liu, “Cubic Spline Interpolation-Based Robot Path Planning Using a Chaotic Adaptive Particle Swarm Optimization Algorithm,” Mathematical Problems in Engineering 2020.
21. L. Consolini, M. Locatelli, A. Minari, and A. Piazzi, “A linear-time algorithm for minimum-time velocity planning of autonomous vehicles,” in 2016 24th Mediterranean Conference on Control and Automation (MED). IEEE, 2016, pp. 490–495.
22. A. Rahatgaonkar, “Velocity planning approach for autonomous vehicles,” Master’s thesis, 2016.
23. X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, “Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications,” IEEE/ASME Transactions on mechatronics, vol. 21, no. 2, pp. 740–753, 2015.

## AUTHORS PROFILE

**Hrishikesh Dey** completed the Bachelor of Engineering (B.E.) degree in Electronics Engineering from the Vivekanand Education Society's Institute of Technology, Mumbai, India, in 2021. He is currently admitted to the North Carolina State University, Raleigh, NC, United States for a Master’s in Science (MS) degree. His research interests include navigation techniques of autonomous systems, control of robotic systems, interrelated with image analysis and signal processing of intelligent systems.

**Rithika Ranadive** completed the Bachelor of Engineering (B.E.) degree in Electronics Engineering from the Vivekanand Education Society's Institute of Technology, Mumbai, India, in 2021. She is currently admitted to the University of Minnesota, Minneapolis, MN, United States for a Master’s in Science (MS) degree. Her research interests include integrated circuits and VLSI, with a primary focus in digital design and computer-aided design (CAD) techniques.

**Abhishek Chaudhari** completed the Bachelor of Engineering (B.E.) degree in Electronics and Telecommunication from Prof. Ram Meghe Institute of Technology, Amravati, India in 2009. He completed the Masters of Technology (M.Tech) in Digital Image Processing from Shree Guru Gobind Singhji Institute of Engineering and Technology, Nanded, India, in 2011. He is currently an Assistant Professor in Vivekanand Education Society's Institute of Technology, Mumbai, India. His research interest includes Computer Vision, Image Processing and Machine Learning.