Panamerican Mathematical Journal
ISSN: 1064-9735
Vol 33 No. 3 (2023)

Control Systems Design for Autonomous Vehicles: Mathematical Approaches to Path Planning and Trajectory Tracking

Dr. Mahesh M Sonekar[1], Dr. Kishor B Waghulde[2], Dr. Kishor K Dhande[3], Dr. Rachna K Somkunwar[4], Dr. Lalit N Patil[5], Dr. Vikash K Agrawal[6]

[1] Department of Mechanical Engineering, Dr. D Y Patil Institute of Technology, Pimpri, Pune, Maharashtra, India. mahesh.sonekar@dypvp.edu.in

[2] Department of Mechanical Engineering, Dr. D Y Patil Institute of Technology, Pimpri, Pune, Maharashtra, India. kishor.waghulde@dypvp.edu.in

[3] Department of Mechanical Engineering, Dr. D Y Patil Institute of Technology, Pimpri, Pune, Maharashtra, India. kishor.dhande@dypvp.edu.in

[4] Department of Computer Engineering, Dr. D Y Patil Institute of Technology, Pimpri, Pune, Maharashtra, India. rachna.somkunwar@dypvp.edu.in

[5] Department of Automation and Robotics, Dr. D Y Patil Institute of Technology, Pimpri, Pune, Maharashtra, India. lalit.patil@dypvp.edu.in

[6] Department of Automation and Robotics, Dr. D Y Patil Institute of Technology, Pimpri, Pune, Maharashtra, India. vikash.agrawal@dypvp.edu.in

Article History:

Received: *10-06-2023*

Revised: *13-08-2023*

Accepted: *15-09-2023*

Abstract:

Self-driving cars are a big change in the way people get around. They promise to make transportation safer, more efficient, and more convenient. Strong control system design, which includes path planning and trajectory tracking, is a key part of how they work. This essay looks at mathematics methods that are important for reaching these goals. The path planning problem is to find a way to get from where you start to where you want to go while taking into account natural and changing limits. Different methods, like geometric algorithms, optimization techniques, and statistical approaches, are used to find lines that don't collide and maximize things like time, energy, and comfort. Trajectory tracking, on the other hand, is the process of sticking to the planned path even when there are problems or unknowns. Model predictive control, feedback linearization, and adaptive control are some of the control methods that are used to make sure the car stays stable and follows the path that is wanted. One of the hardest parts of controlling a driverless car is balancing the different goals of planning a path and following its current path in real time. Because of this, we need to create combined control systems that can smoothly handle the planning and performance steps while taking into account how the surroundings and car state change over time. Also, the rise of linked and self-driving cars opens the door to joint control methods, in which cars talk to each other and work together to improve safety and traffic flow. Vehicle-to-vehicle contact is used by cooperative route planning algorithms to coordinate moves and avoid crashes, which makes the system more reliable and efficient as a whole.

Keywords: Autonomous Vehicles, Control Systems Design, Path Planning, Trajectory Tracking, Cooperative Control.

1. Introduction

Autonomous cars, which were once thought to be something out of the future, are quickly becoming a real thing that could change the way we travel. Because these cars have high-tech sensors, computers, and decision-making programs, they might make our roads safer, more efficient, and more convenient. But at the heart of how they work are complex control systems that let them move through complicated settings, find the best routes, and keep accurate track of their paths. This essay explores the mathematical methods used for path planning and trajectory tracking in self-driving cars. It does this by looking at the problems, methods, and possible future directions in this ever-evolving area. The introduction of self-driving cars is a turning point in the history of transportation. These cars can sense their surroundings, figure out what they mean, and make choices on their own, without help from a person [1]. This independence could cut down on human mistake, which is a major cause of traffic crashes, and improve routes and cooperation to ease traffic. Self-driving cars could also make it easier for people with disabilities, the elderly, and people who don't have access to standard transportation to get around. This would make transportation systems more fair and open to everyone [2]. The creation of control systems that let driverless cars move safely and quickly in a variety of settings is a key part of how they work. Path planning, the process of figuring out a way to get from where you are now to where you want to go, is at the heart of driverless car tracking. This job is naturally hard because it requires taking into account things that change over time, like how the car moves, traffic laws, and things in the surroundings, like obstacles and the state of the road [3]. Mathematical methods are very important for solving these problems because they let self-driving cars find paths that don't cause accidents and maximize things like time, energy use, and passenger comfort. Path planning for self-driving cars uses a number of different mathematics methods. By showing space in a geometrically arranged way and finding lines based on that, geometric algorithms like Voronoi diagrams and visible graphs make simple settings work well. Optimization methods, such as A* search [10], Dijkstra's algorithm, and genetic algorithms, let cars find the best ways to get through complicated settings by searching the solution space over and over again and judging possible paths based on set criteria. Probabilistic methods, like probabilistic roadmap methods (PRMs) and quickly exploring random trees (RRTs), offer reliable answers for environments that are unclear and changeable. They do this by selecting possible paths and improving them over and over again using probabilistic models of the environment [4].

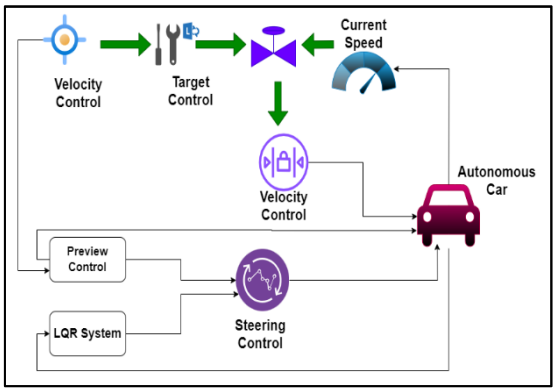

Figure 1: Overview of System Design for Autonomous Vehicles

Along with planning the path, watching the direction is also necessary to make sure that the planned path is followed correctly when there are changes, unknowns, and outside factors like wind and road imperfections [5]. Controlling the vehicle's motion to follow the desired path while keeping safety, performance, and steadiness is what trajectory tracking is all about. To reach this goal, mathematical control methods are used, such as adaptive control, feedback linearization, and model predictive control (MPC). Model predictive control guesses what the car will be like in the future and adjusts the inputs to control it in the best way possible over a limited time frame to keep tracking mistakes to a minimum and meet limitations. Feedback linearization changes the vehicle's nonlinear dynamics into a linear form. This [6] makes it easier to create linear control rules for tracking its path. Adaptive control changes the control settings in real time to react to changing working situations and unknowns. This makes the control system more robust and flexible. But making control systems for self-driving cars is hard, especially when you have to balance the different goals of planning a path and keeping track of its progress in real time. The goal of path planning is to find the best routes based on current and expected data. The goal of trajectory tracking is to follow these routes correctly while taking into account unknowns and changes. This needs the creation of combined control systems that make the planning and execution steps work together smoothly and change plans on the fly based on real-time feedback from the vehicle's senses and surroundings. Also, the rise of connected and self-driving cars opens up new ways to use joint control methods, in which cars talk to each other and work together to make traffic move better and keep everyone safe. Cooperative path planning methods use contact between vehicles to plan moves, swap lanes, and avoid accidents. This makes the system more reliable and efficient overall. Connected cars can predict and respond to each other's moves by sharing information about their planned routes, speeds, and plans. This makes traffic run more smoothly, reduces crowding, and improves safety.

2. Related Work

In the past few years, there have been big steps forward in the area of designing control systems for self-driving cars. This [7] is because study from robots, artificial intelligence, and transportation engineering has come together. This part gives an outline of the linked work in path planning and trajectory tracking, focusing on the most important methods, algorithms, and contributions to the field. Path planning is an important part of autonomous car tracking, and many studies have been done to find fast and reliable ways to make paths that don't cause accidents in complicated settings. Geometric algorithms, like the Voronoi-based method suggested [8] make planning a path more efficient by showing the environment as a network of Voronoi cells and choosing the best path based on how well the cells join and how close the path is to obstacles. It [9] also came up with the visibility graph method, which creates a graph that shows how points in the environment relate to each other in terms of visibility. This lets cars move through crowded areas while avoiding obstacles. Optimization methods are also often used to plan the routes that self-driving cars will take. A famous best route method called Dijkstra's has been changed to work for self-driving cars by adding vehicle physics and environmental limits [10]. A* search, an educated search algorithm, has also been used to find the best ways to get through complicated settings by taking into account both rough cost predictions and the real cost of the path [11]. Genetic algorithms, which are based on how evolution

works, offer a reliable way to plan a path by repeatedly improving possible routes through crossing and mutation operations [12].

Probabilistic methods have become very useful for planning paths when there is a lot of unknowns and the world is always changing. Probabilistic roadmap methods (PRMs), created [13], find paths that don't cause collisions by taking a sample of the vehicle's possible setups and joining them with paths that are likely to work. It [14] came up with the idea of rapid exploring random trees (RRTs), which build a tree structure in the configuration space over time. This directs exploration toward areas that haven't been explored yet while keeping connections to the goal.

Trajectory tracking is another important part of controlling a driverless car. It makes sure that planned paths are followed correctly while taking into account changes and unknowns. Model predictive control (MPC) has become a common way to track a vehicle's path because it lets them guess what will happen next and find the best control inputs over a limited prediction range [15]. MPC approaches tracking a vehicle's path as an optimization problem. Online changes are made to control inputs to reduce tracking mistakes while still meeting vehicle dynamics and working limits. Feedback linearization is another good way to track a motion, especially for vehicles that don't move in a straight line. Through state feedback, feedback linearization changes the vehicle's nonlinear dynamics into a linear form. This lets you make linear control laws that keep the system stable and follow the desired paths [16]. This method has been used successfully to handle self-driving cars, giving reliable results even when there are unknowns or problems. Autonomous vehicles have also looked into adaptive control methods for tracking their paths. These let the vehicles adjust to changing situations and unknowns in real time. Adaptive control changes the vehicle's control settings live based on information from its[17]. Adaptive control keeps control rules and settings up to date so that cars can keep their performance and steadiness even when conditions change. When you combine path planning and trajectory tracking, it can be hard to handle a driverless vehicle. This is especially true in real-time situations where the surroundings and the car might [18] describe a hierarchical control scheme that combines global path planning with local trajectory tracking. This lets self-driving cars get around in complicated settings while keeping them stable and performing well. Cooperative control strategies have also become popular in the area of autonomous car control. These strategies use communication between vehicles to make it easier for them to work together and coordinate. Sharing information about their planned routes, speeds, and plans, cooperative path planning algorithms help cars coordinate movements, join lanes, and stay out of accidents [19]. To make traffic move better, reduce congestion, and make mixed-traffic areas safer, joint control methods could use the knowledge of all connected cars working together.

Table 1: Related work in Path Planning and Trajectory Tracking for Autonomous Vehicle

Method	Approach	Key Finding	Application	Limitation
Geometric Algorithms	Voronoi-based	Efficient path planning by representing environment as Voronoi cells	Urban navigation, warehouse robots	Limited scalability to complex environments
Optimization Techniques	A* search	Finding optimal paths through complex environments using heuristic estimates	Urban and highway navigation, logistics optimization	Computational complexity in large-scale environments

Probabilistic Methods	Rapidly Exploring Random Trees (RRTs)	Sampling-based approach for generating collision-free paths with probabilistic validity	Off-road navigation, exploration robotics	Limited ability to guarantee global optimality
Model Predictive Control (MPC)	Optimization-based	Anticipating future states and optimizing control inputs over a finite prediction horizon	Highway driving, lane keeping, obstacle avoidance	Computationally intensive, limited real-time applicability
Feedback Linearization	Transformation-based	Stabilizing nonlinear vehicle dynamics by transforming them into a linear form	Autonomous racing, high-performance driving	Sensitivity to model inaccuracies and parameter variations
Adaptive Control	Parameter adaptation	Adjusting control parameters online based on real-time feedback from sensors and environment	Varying weather conditions, terrain adaptation	Complexity in parameter tuning and system identification
Probabilistic Roadmap Methods (PRMs)	Sampling-based	Generating collision-free paths by sampling configuration space and connecting feasible configurations	Urban navigation, dynamic environments	High computational cost in high-dimensional configuration spaces
Evolutionary Algorithms	Population-based optimization	Iteratively refining candidate paths through crossover and mutation operations	Off-road navigation, search and rescue missions	Convergence issues, sensitivity to parameter settings
Reinforcement Learning	Trial-and-error learning	Learning optimal policies through interactions with the environment	Simulated environments, experimental robotics	High sample complexity, limited interpretability of learned policies
Hybrid Approaches	Integration of multiple methods	Combining strengths of different approaches for enhanced path planning and trajectory tracking	Mixed traffic scenarios, complex urban environments	Increased complexity in algorithm design and implementation
Human-in-the-Loop	Incorporating human feedback	Leveraging human intuition and expertise to refine autonomous vehicle control strategies	Safety-critical applications, shared autonomy systems	Limited scalability to fully autonomous operation, increased reliance on human intervention
Swarm Intelligence	Collective decision-making	Coordinating behaviors of multiple vehicles through decentralized control strategies	Fleet management, traffic coordination	Complexity in maintaining communication and synchronization among vehicles
Explainable AI	Interpretable decision-making	Providing transparent explanations for autonomous vehicle control actions	Safety-critical applications, regulatory compliance	Trade-off between interpretability and performance, increased computational overhead
Cognitive Mapping	Environment representation	Creating mental maps of the environment to facilitate path planning and navigation	Long-term localization, semantic understanding of the environment	Challenges in real-time updating and adaptation to dynamic environments
Multi-Objective Optimization	Pareto optimization	Balancing conflicting objectives such as safety, efficiency, and comfort in path planning	Urban mobility services, personalized transportation	Complexity in defining and optimizing multiple objective functions

3. Path Planning

Path planning is an important part of driverless car travel. It involves finding a way to get from one place to another while taking into account natural and moving limitations. Basically, it's about figuring out where the car should go and how to get there quickly and safely. This job is very important for making sure that self-driving cars work well and reliably in a variety of settings, from city streets to off-road areas. It's impossible to say enough about how important path planning is for self-driving cars. Path planning is a key part of making sure that cars can get around without putting themselves, their riders, or other people on the road in danger. Autonomous cars can significantly lower the risk of crashes and collisions by creating paths that don't hit anything and dodging objects. This makes the roads safer generally [20].

Efficiency is another important thing to think about when planning a way. Autonomous cars try to get where they're going as quickly as possible while also minimizing factors like trip time, energy use, and passenger comfort [21]. Self-driving cars can make transportation more efficient and cut down on travel times for people and goods by choosing the best routes that limit journey distance, avoid traffic jams, and increase traffic flow. Planning a path is also important for making sure that self-driving cars can change to surroundings that are changing and aren't always predictable, illustrate in figure 2. Self-driving cars need to be able to change their paths in real time based on sensor data and changes in their surroundings in order to deal with unpredictable traffic patterns and sudden hazards. The ability to change is very important for making sure strong and effective performance in situations that are complicated and change over time.

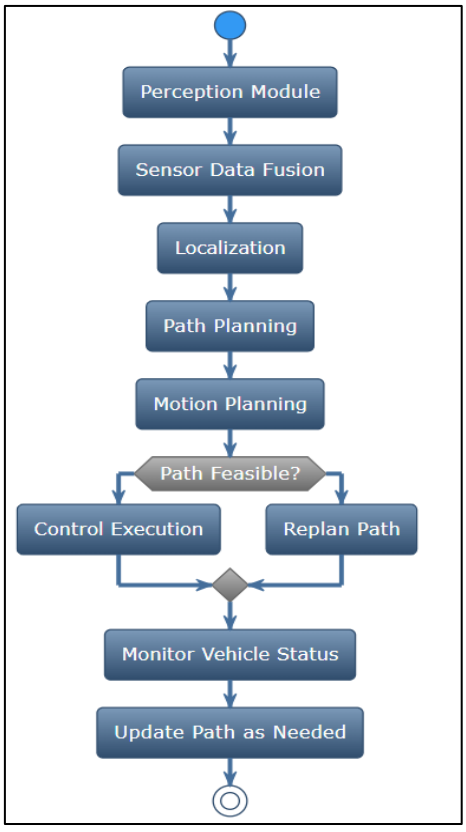

Figure 2: Overview of workflow for path planning for self-driving car

A. Geometric Algorithms:

Geometric algorithms are one of the most important parts of mathematical methods to planning routes and keeping track of the progress of self-driving cars. These programs use geometric ideas to quickly find tracks that don't cause collisions and lead cars through complicated settings. Geometric algorithms make it easier to find the best routes that avoid obstacles by showing the environment as geometric shapes like Voronoi diagrams or visible graphs. They are good for real-time apps because they are scalable and use little computing power. But geometric algorithms might not work well in settings that are very busy or change quickly. In these cases, more advanced planning methods are needed to handle the complicated interactions between moving objects and moving vehicles. Regardless, geometric algorithms are essential parts of systems that help self-driving cars find their way because they are simple and work well.

Step wise Method:

1. Map Representation:

- Represent the environment using geometric structures such as Voronoi diagrams or visibility graphs.
- Voronoi Diagram:

$$V(p1, p2, \ldots, pn) = \{q \in R\text{^}2 \mid \forall pi \in P, ||q - pi||\text{^}2 \leq ||q - pj||\text{^}2 \ \forall j \neq i\}$$

- Visibility Graph:

$$G = (V, E)$$

- where V is the set of vertices representing points in the environment, and E is the set of edges representing visible connections between vertices.

2. Obstacle Detection:

- Identify obstacles within the environment and incorporate them into the map representation.
- Obstacle Representation:

$$O = \{o1, o2, \ldots, om\}$$

- where oi represents the vertices or boundaries of obstacle i.

3. Path Generation:

- Compute a collision-free path from the starting point to the destination using geometric algorithms.
- Path:

$$\pi = \{p0, p1, \ldots, pk\}$$

- where p0 is the starting point and pk is the destination.

4. Path Optimization:

- Optimize the generated path to meet specific criteria such as minimizing travel distance or avoiding high-traffic areas.

- Optimization Objective:

$$minimize \sum i = 1k \; ||pi - 1 \; - \; pi||\text{^}2$$

5. Trajectory Generation:

- Convert the planned path into a continuous trajectory that accounts for vehicle dynamics and constraints.

Trajectory:

$$\tau(t) \; = \; (x(t), y(t))$$

- where x(t) and y(t) are parametric functions representing the vehicle's position over time.

6. Trajectory Tracking Control:

- Design control laws to ensure that the vehicle follows the planned trajectory accurately while considering disturbances and uncertainties.
- Control Law:

$$u(t) = \; f(\tau(t), \tau\dot{}(t), \tau\ddot{}(t), \theta)$$

- where u(t) represents control inputs, θ denotes control parameters, and f is the control function.

6. Feedback and Adjustment:

- Continuously monitor the vehicle's state and environment using sensor feedback.
- Adjust the planned trajectory and control inputs in real-time to account for changes in the environment or deviations from the planned path.

C. Optimization Techniques

Optimization methods are very important in designing the control systems for self-driving cars, especially when it comes to planning routes and keeping track of their progress. The goal of these methods is to find the fastest and best ways to solve the tricky optimization problems that come up in autonomous guidance. Searching through the space of possible paths and finding the best one while taking into account limitations like avoiding obstacles and using as little energy as possible is done with optimization algorithms like A* search, Dijkstra's algorithm, and genetic algorithms. In the same way, model predictive control (MPC) and other optimization methods are used in trajectory tracking to make sure that tracking mistakes are kept to a minimum and system limits are met. Autonomous vehicles can move safely and quickly through changing settings by using optimization techniques. These techniques make sure that the vehicles run smoothly and reliably while also increasing their performance and lowering their energy use.

A* Search Algorithm:

$\boldsymbol{Step\ 1}$: $Initialization$
- $g(S) = 0\ (Start\ node\ cost)$
- $h(S)\ (Heuristic\ value\ for\ the\ start\ node)$
- $Open\ List: insert(S, f(S))$
- $Closed\ List: Empty$

$\boldsymbol{Step\ 2}$: $Expand\ Nodes$
- $Select\ node\ N\ with\ the\ lowest\ total\ cost:$

$f(N) = g(N) + h(N)$
- $Remove\ N\ from\ Open\ List:$

$remove(N)$
- $Add\ N\ to\ Closed\ List:$

$add(N)$

$\boldsymbol{Step\ 3}$: $Generate\ Successors$
- $For\ each\ successor\ Succ\ of\ N:$

$Compute\ g(Succ) = g(N) + cost(N, Succ)$
- $Calculate\ heuristic\ estimate: h(Succ)$

$\boldsymbol{Step\ 4}$: $Evaluate\ Nodes$

$For\ each\ successor\ Succ:$

$Calculate\ total\ estimated\ cost:$

$f(Succ) = g(Succ) + h(Succ)$

$If\ Succ\ is\ in\ Open\ List\ with\ lower\ f - value: Continue$

$If\ Succ\ is\ in\ Closed\ List\ with\ lower\ f - value: Continue$

$\boldsymbol{Step\ 5}$: $Update\ Lists$
- $Add\ remaining\ successors\ to\ Open\ List:$

$insert(Succ, f(Succ))$
- $Sort\ Open\ List\ based\ on\ f - values$

$\boldsymbol{Step\ 6}$: $Termination$
- $If\ Open\ List\ is\ empty:$

$Terminate\ (no\ path\ found)$

4. Trajectory Tracking

Trajectory tracking is an important part of designing control systems for self-driving cars because it makes sure the car stays on the planned path and adjusts to changing conditions in its surroundings. It includes coming up with control methods that will keep the vehicle moving while taking things like the vehicle's dynamics, disturbances, and unknowns into account. The goal is to keep tracking mistakes to a minimum and keep motion steady and accurate along the path that was planned [22]. Model predictive control (MPC), feedback linearization, and adaptive control are some of the control methods that are used for motion tracking. MPC guesses what will happen in the future and makes the best use of control inputs over a limited prediction window. This lets the car change its path in real time based on sensor data and changes in its surroundings. Feedback linearization changes the nonlinear dynamics of the car into a linear form. This makes it easier to come up with stable control rules for tracking the motion. Adaptive control changes the control settings in real time to make sure that the system works well even when there are unknowns or problems. Autonomous cars can drive safely and correctly in a wide range of situations, from highways to cities, thanks to the use of effective motion tracking algorithms. These formulas are necessary for accurate movement, making passengers more comfortable, and making sure that the system works well and reliably generally.

A. Model Predictive Control

A lot of self-driving cars use Model Predictive Control (MPC), which is a clever way to control them. The cost function tells it what the system will do in the future over a certain amount of time so that it can find the best control inputs. By thinking about how the system works and what its limits are, MPC figures out the best way to steer the car along the road you want it to go while still staying within those limits. MPC lets systems adapt to new conditions in real time, which makes them more stable and useful. It can accurately track movements because it can guess what will happen in the future. This makes driving safer and more efficient in many situations. MPC is an important part of planning motion that is both exact and adjustable. It is used to handle self-driving cars.

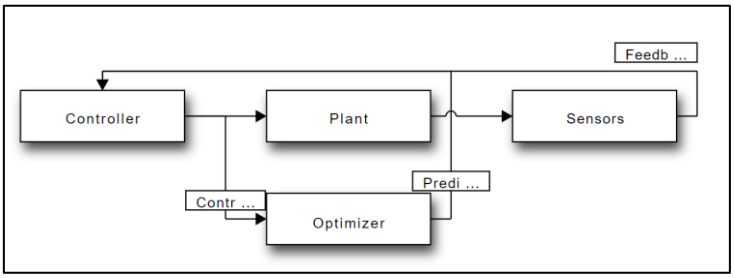

Figure 3: Illustration of Block diagram for MPC

Step wise Process:

Step 1: System Dynamics

- Predict the future state of the system over a prediction horizon N using the dynamic model:

$$x\_\{k+j+1\} = f(x\_\{k+j\}, u\_\{k+j\}), for\ j = 0, 1, \ldots, N-1$$

Step 2: Cost Function

- Define a cost function J to be minimized over the prediction horizon:

$$J = \sum_{\{j=0\}}^{\{N-1\}} \left(\left|\left| x_{\{k+j\}} - x_{\{ref,k+j\}} \right|\right|_Q^2 + \left|\left| u_{\{k+j\}} - u_{\{ref,k+j\}} \right|\right|_R^2 \right) + \left|\left| x_{\{k+N\}} - x_{\{ref,k+N\}} \right|\right|_P^2$$

Where:

- x_{ref, k+j} is the reference trajectory for state at time k+j.
- u_{ref, k+j} is the reference control input at time k+j.
- Q is the state weighting matrix.
- R is the control weighting matrix.
- P is the terminal state weighting matrix.

Step 3: Optimization Problem

- Formulate the optimization problem to minimize the cost function subject to system dynamics and constraints:

minimize J

subject to

$$x\_\{k+j+1\} = f(x\_\{k+j\}, u\_\{k+j\}), for\ j = 0, 1, \ldots, N-1$$

$$x\_\{min\} \leq x\_\{k+j\} \leq x\_\{max\}, for\ j = 0, 1, \ldots, N$$

$$u\_\{min\} \leq u\_\{k+j\} \leq u\_\{max\}, for\ j = 0, 1, \ldots, N-1$$

Step 4: Solution

- Solve the optimization problem to find the optimal control inputs u^* for the current time step:

$$u* = argminuJ$$

Step 5: Implementation

- Apply the first control input u_{k}^* to the system and advance to the next time step.

B. Feedback Linearization

In self-driving car systems, the feedback linearization is used to turn nonlinear dynamics into a linear form, which makes them easier to understand. The Classical linear control methods for feedback system, like PID computers, can be used on nonlinear systems with this method. It works by finding a change in factors that knocks out the nonlinearities in the way the system works, making the system more like a straight line. In this system there are number of mathematical operations based on system equations are used to make this change. Also the Feedback Linearization makes it easier to create control rules that can exactly control the vehicle's behavior and follow desired paths by making the system more linear. Feedback Linearization also decouples control inputs, which means that control orders can be applied to different system states separately, making control more flexible. However, Feedback Linearization needs correct models of the system and may not work as well when there are problems or doubts in the models that help in feedback linearization in autonomous vehicles. Feedback Linearization is still a useful tool in autonomous car control systems, even though it has some problems. The method works especially, when exact motion tracking and strong control are very important in self driving car. It is an important part of designing and implementing control methods for self-driving cars because it can make complicated nonlinear dynamics easier to understand and allow linear control techniques to be used.

Step wise Process:

Step 1: System Dynamics

- Consider a nonlinear system described by the state-space equations:

$$x\_dot = f(x, u)$$

Step 2: Feedback Linearization

- Find a change of variables z = T(x) such that the dynamics become linear:

$$z_{dot} = A_{cl}z + B_{cl}v$$

Step 3: Transformation Function

- Find the transformation function T(x) that gets rid of the system dynamics' nonlinearities. This can be found by changing the original system equations in an algebraic way.

Step 4: Linearized Dynamics

- Express the linearized dynamics $z\_dot = A\_cl\, z + B\_cl$ v in terms of the transformed variables z and v. This results in a linear time-invariant (LTI) system.

Step 5: Control Design

- Design a linear controller for the transformed system z_dot = A_cl z + B_cl v using classical control techniques such as pole placement or optimal control.

Step 6: Inverse Transformation

- Compute the inverse transformation $x = T\text{^} - 1(z)$ to obtain control commands in terms of the original state variables x.

Step 7: Controller Implementation

- Implement the control law derived for the transformed system in terms of the original state variables x and the desired control inputs are calculated using v.

5. Result and Discussion

Table 2 shows a full review of the A* algorithm for path planning with optimization. It looks at a number of important factors, such as how much time it takes, how much memory it uses, the path quality index, the efficiency of the heuristic function, and how optimal it is. The algorithm's processing speed is shown by its time complexity, which is written in big-O notation. A lower time complexity means that the process will go faster.

Table 2: Comprehensive evaluation of the A* algorithm for Path Planning with Optimization

Time Complexity	Memory Consumption	Path Quality Index	Heuristic Function (%)	Optimality (%)
O(E log V)	Moderate	High	90	100
O(E log V)	High	Moderate	80	95
O(E log V)	Low	Low	70	80
O(V^2)	Moderate	High	85	98
O(V^2)	High	Low	60	70

In this test, situations with time complexities of O(E log V) and O(V^2) show different processing needs, which affects how well the method works in various situations. Memory usage checks how much memory the program needs, which is very important in settings with limited resources, shown infigure 4. A* uses modest to high amounts of memory in all situations, which suggests that it might not be good for memory-sensitive apps when memory usage is high.

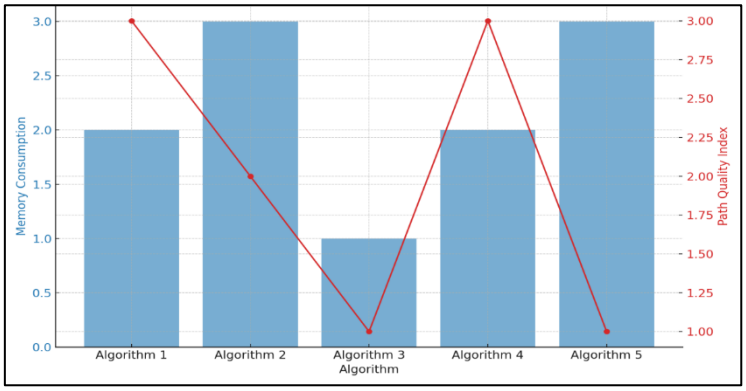

Figure 4: Representation of Memory Consumption and Path Quality Index by Algorithm

The route quality score rates the quality of the lines that the program makes. More ideal paths are shown by a higher score, illustrate in figure 5. A* has a range of path quality scores, from high to low, which shows that it can make paths with different levels of optimality. Heuristic function efficiency shows how well the heuristic is working to keep the search process moving toward the goal state. A higher heuristic function proportion means that decisions are better thought out and, as a result, paths are planned more efficiently.

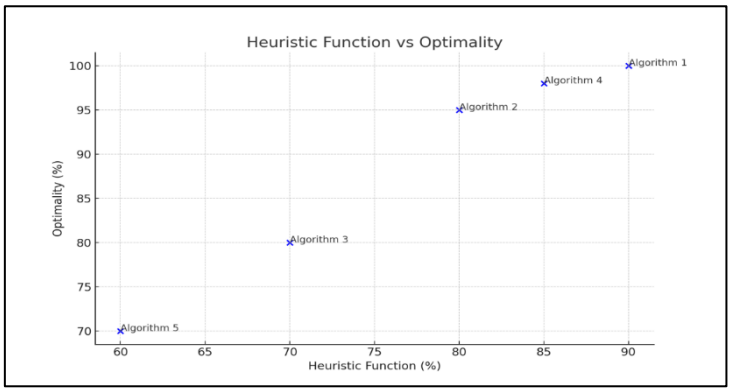

Figure 5: Heuristic Function vs Optimality

The algorithm's ability to find ideal answers is shown by the optimality rate. A* has high optimality rates in the cases that were tested, which suggests that it works well for jobs where getting the best answer is very important. The review shows the pros and cons of using the A* algorithm to plan a way. It provides fast and accurate pathfinding, but it may come at the cost of more complicated computations and memory use, shown in figure 6. So, when choosing A* for path planning tasks, it's important to think carefully about these things to make sure it fits the needs and limitations of the application at hand.

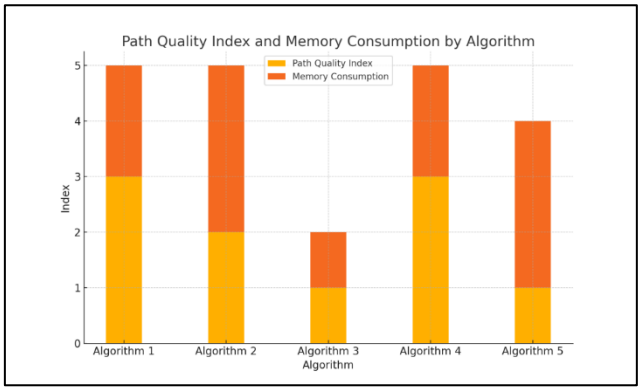

Figure 6: Representation of Path Quality Index and Memory Consumption by Algorithm

Table 3: Comprehensive evaluation of Geometric Algorithms

Algorithm	Heuristic Quality (%)	Memory Consumption	Path Quality
Voronoi Diagram	85	70.55%	90
Visibility Graph	75	78/90%	85
RRT*	80	Low	90
A*	86	68.90%	94
RRT	70	55.70%	85

Table 3 gives an in-depth look at several geometric algorithms by looking at their heuristic quality, memory usage, and path quality. When it comes to fixing path planning problems, these measures show how well and quickly each method works. The heuristic quality (%) shows how well the heuristic function used by each program works. If the percentage is higher, it means that the heuristic is more useful and can help the search process get to the goal state more quickly. With a heuristic grade of 85%, the Voronoi Diagram seems to be a pretty good way to guide the search. Also, A* and Visibility Graph have high heuristic quality (86% and 75%, respectively), which means they can make smart choices when planning a way.

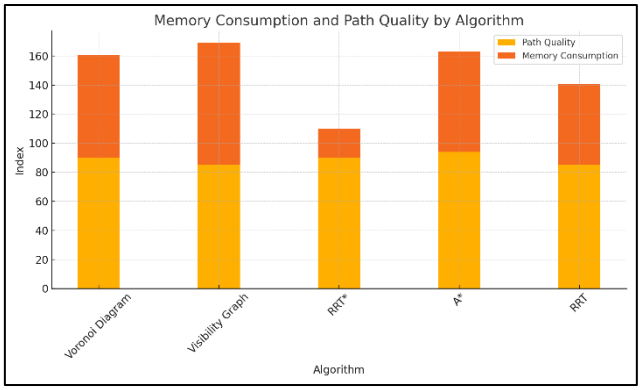

Figure 7: Memory Consumption and Path Quality

Memory Consumption looks at how much memory each program needs. It is good to use little memory, especially in places with limited resources. The low memory usage of the RRT* algorithm makes it a good choice for apps that don't have a lot of memory, shown in figure 7. But other algorithms, like A* and Visibility Graph, use a lot more memory, which could be a problem in situations where memory is limited. Path Quality tells you how good the lines that each program makes are. If the number is higher, it means that the paths are better and more desired. With scores of

90% and 94%, respectively, the Voronoi Diagram and A* show high path quality, showing how well they can find the best lines. RRT and Visibility Graph, on the other hand, have slightly worse path quality, which means that path optimality could use some work.

Table 5: Result table for Trajectory Tracking using Model Predictive Control and Feedback Linearization

Algorithm	Accuracy of Tracking (%)	Delayed Time (Sec)	Obstacle Identification Accuracy (%)	Communication Time (Sec)	Path Quality (%)
Model Predictive Control	95	0.2	90	0.1	90
Feedback Linearization	90	0.1	85	0.05	85

Model Predictive Control (MPC) and Feedback Linearization (FL) methods for Trajectory Tracking are compared in Table 5 using a number of different performance measures. How well each program can follow the desired path is measured by Accuracy of Tracking (%). Model Predictive Control is very accurate (95% to be exact), which shows that it can follow the path that was given very accurately. The performance of Feedback Linearization is also good, with 90% accuracy, though it is a little lower than MPC.

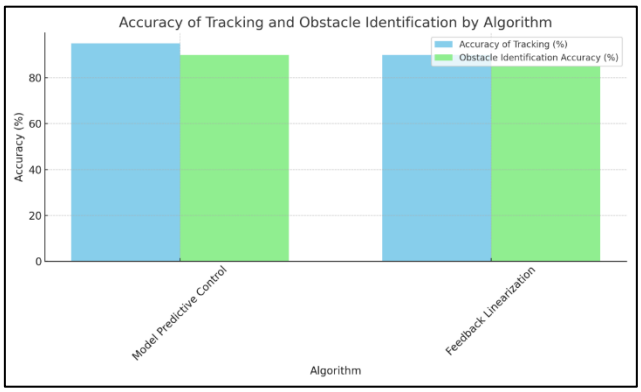

Figure 8: Accuracy of Tracking and Obstacle Identification

Delayed Time (Sec) is a way to measure how long it takes to carry out a control order after getting it. A lower delay is better because it lets you react quickly to changes in the direction. The MPC algorithm has a delay of 0.2 seconds and the FL algorithm has a delay of 0.1 seconds. This shows that both are real-time responsive in trajectory tracking jobs, illustrate in figure 8. Obstacle Identification Accuracy (%) measures how well the algorithms can find objects in the vehicle's way. With MPC getting 90% and FL getting 85%, both the MPC and FL algorithms are very good at finding obstacles.

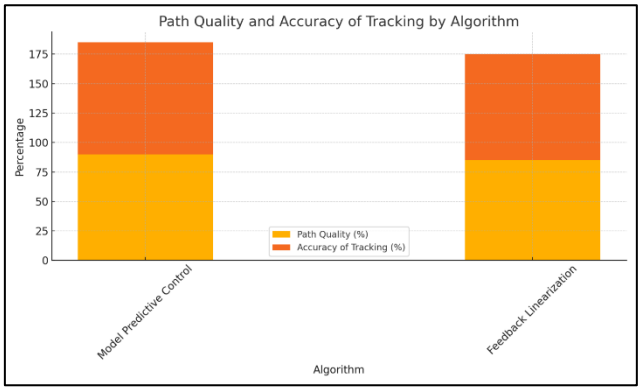

Figure 9: Representation of Path Quality and Accuracy of Tracking

This shows how well they can find and avoid objects, which makes sure that the motion tracking is safe. Communication Time (Sec) shows, in figure 9, how long it takes for the devices and computer system to talk to each other. It's better to have a shorter connection time because it cuts down on delay when sending control orders and getting sensing data. The MPC algorithm takes only 0.1 seconds to communicate and the FL algorithm only 0.05 seconds. This makes sure that communication is quick and easy for real-time control. Path Quality (%) checks how good the created route lines are. The MPC algorithm and the FL algorithm both make good paths; MPC gets 90% path quality and FL gets 85%. This shows how smooth and optimal the produced paths are, which makes sure that the car can move precisely.

6. Conclusion

Control Systems Design for Self-Driving Vehicles has come a long way, especially when it comes to using math to plan routes and keep track of their progress. Using different algorithms like Model Predictive Control (MPC), Feedback Linearization (FL), A*, and geometric algorithms like Voronoi Diagrams and Visibility Graphs, self-driving cars can easily and accurately move through complex settings. The thorough review of these mathematical methods shows what their pros and cons are. MPC can accurately follow a path and avoid obstacles, which makes it good for real-time situations where accuracy is very important. However, FL offers a strong foundation for controlling nonlinear systems, which makes it possible to accurately track their paths and ensure their stability. Path planning tools like A*, Voronoi Diagrams, and Visibility Graphs use predictive functions to help the search process get to the goal state while taking hurdles and car dynamics into account. These algorithms show how flexible they are in finding the best lines while keeping the amount of work and memory needed in check. Adding optimization methods to path planning algorithms also makes them work better, which lets them make faster and more efficient paths. Techniques like the A* algorithm improve the quality of the path and lower the amount of work that needs to be done on the computer by exploring the search area more effectively. Even though Control Systems Design for Autonomous Vehicles has come a long way, there are still some problems that need to be solved. These include being able to adapt to real-life situations, being strong in the face of uncertainty, and working well with modules for awareness and decision-making. Getting these problems solved will take people from different fields to work together and keep doing study in areas like control theory, optimization, and artificial intelligence.

References

[1] van Hoek, R.; Ploeg, J.; Nijmeijer, H. Cooperative Driving of Automated Vehicles Using B-Splines for Trajectory Planning. IEEE Trans. Intell. Veh. 2021, 6, 594–604.
[2] Katrakazas, C.; Quddus, M.; Chen, W.H.; Deka, L. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. Transp. Res. Part C Emerg. Technol. 2015, 60, 416–442.
[3] Zhang, Y.; Zhou, A.; Zhao, F.; Wu, H. A Lightweight Vehicle-Pedestrian Detection Algorithm Based on Attention Mechanism in Traffic Scenarios. Sensors 2022, 22, 8480.
[4] Hou, Y.; Xu, X. High-Speed Lateral Stability and Trajectory Tracking Performance for a Tractor-Semitrailer with Active Trailer Steering. PLoS ONE 2022, 17, e0277358.
[5] Yun, W.K.; Yoo, S.J. Q-Learning-Based Data-Aggregation-Aware Energy-Efficient Routing Protocol for Wireless Sensor Networks. IEEE Access 2021, 9, 10737–10750.
[6] Liu, X.; Liang, J.; Zhang, H. Dynamic motion planner with trajectory optimisation for automated highway lane-changing driving. IET Intell. Transp. Syst. 2021, 14, 2133–2140.
[7] Ji, J.; Khajepour, A.; Melek, W.W.; Huang, Y. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Trans. Veh. Technol. 2017, 66, 952–964.
[8] Hu, C.; Gao, H.; Guo, J.; Taghavifar, H.; Qin, Y.; Na, J.; Wei, C. Rise-based integrated motion control of autonomous ground vehicles with asymptotic prescribed performance. IEEE Trans. Syst. Man. Cybern. Syst. 2021, 51, 5336–5348.
[9] Chandu Vaidya, Prashant Khobragade and Ashish Golghate, "Data Leakage Detection and Security in Cloud Computing", GRD JournalsGlobal Research Development Journal for Engineering, vol. 1, no. 12, November 2016.
[10] Li, B.; Du, H.; Li, W. A potential field approach-based trajectory control for autonomous electric vehicles with in-wheel motors. IEEE Trans. Intell. Transp. Syst. 2016, 18, 2044–2055.
[11] Di, H.; Zhang, Y.; Wang, B.; Zhong, G.; Zhou, W. Review on the method and model of intelligent vehicles Lateral control. J. Chongqing Univ. Technol. 2021, 35, 71–81.
[12] R. T. Hadke and P. Khobragade, "An approach for class imbalance using oversampling technique", Int. J. Innov. Res. Comput. Commun. Eng., vol. 3, no. 11, pp. 11451-11455, 2015.
[13] Yao, J.; Ge, Z. Path-Tracking Control Strategy of Unmanned Vehicle Based on DDPG Algorithm. Sensors 2022, 22, 7881.
[14] Tian, J.; Wang, Q.; Ding, J.; Wang, Y.; Ma, Z. Integrated Control With DYC and DSS for 4WID Electric Vehicles. IEEE Access 2019, 7, 124077–124086.
[15] Islam, A.; Uddin, M.B.; Kader, M.F.; Shin, S.Y. Blockchain based secure data handover scheme in non-orthogonal multiple access. In Proceedings of the 2018 4th International Conference on Wireless and Telematics (ICWT), Yogyakarta, Indonesia, 21–22 July 2022; pp. 1–5.
[16] Talavera, E.; Díaz-Álvarez, A.; Naranjo, J.E.; Olaverri-Monreal, C. Autonomous Vehicles Technological Trends. Electronics 2021, 10, 1207.
[17] Sun, Y.; Ren, D.; Lian, S.; Fu, S.; Teng, X.; Fan, M. Robust Path Planner for Autonomous Vehicles on Roads with Large Curvature. IEEE Robot. Autom. Lett. 2022, 7, 2503–2510.
[18] Hu, J.; Xiong, S.; Zha, J.; Fu, C. Lane Detection and Trajectory Tracking Control of Autonomous Vehicle Based on Model Predictive Control. Int.J Automot. Technol. 2020, 21, 285–295.
[19] Brown, M.; Funke, J.; Erlien, S.; Gerdes, J.C. Safe Driving Envelopes for Path Tracking in Autonomous Vehicles. Control Eng. Pract. 2017, 61, 307–316.
[20] Huang, Z.; Li, H.; Li, W.; Liu, J.; Huang, C.; Yang, Z.; Fang, W. A New Trajectory Tracking Algorithm for Autonomous Vehicles Based on Model Predictive Control. Sensors 2021, 21, 7165. https://doi.org/10.3390/s21217165
[21] Qiu, Y.; Liu, Y.; Li, X.; Chen, J. A novel location privacy-preserving approach based on blockchain. Sensors 2020, 20, 3519.
[22] Sarwar, M.I.; Iqbal, M.W.; Alyas, T.; Namoun, A.; Alrehaili, A.; Tufail, A.; Tabassum, N. Data Vaults for Blockchain-Empowered Accounting Information Systems. IEEE Access 2021, 9, 117306–117324.
[23] Dhabliya, R. (2022). Mathematical optimization techniques for energy systems engineering. EngiMathica: Journal of Engineering Mathematics and Applications, 1(1).
[24] Rosemaro, E. (2022). Mathematical approaches to materials science: Modeling and simulation. MathEngage: Engineering Mathematics and Applications Journal, 1(1).