IJRCS

International Journal of Robotics and Control Systems
Vol. 5, No. 2, 2025, pp. 1571-1595
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

Real-Time Autonomous Vehicle Navigation via Rule-Based Waypoint Selection and Spline-Guided MPC

Wael A. Farag [a,1,*], Mohamed Fayed [b,2]

[a] Electrical Power Engineering Dept., Cairo University, Giza 12613, Egypt
[b] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait
[1] wael.farag@cu.edu.eg; [2] mohamed.fayed@aum.edu.kw
* Corresponding Author

ARTICLE INFO

Article history
Received April 18, 2025
Revised June 20, 2025
Accepted June 24, 2025

Keywords
MPC Control;
Quintic Spline Trajectory Optimization;
Self-Driving Car;
Autonomous Driving;
Highway Navigation;
Real-Time Obstacle Avoidance;
Path Planning

ABSTRACT

This paper presents a robust and efficient Localized Spline-based Path-Planning (LSPP) algorithm aimed at improving autonomous highway navigation. LSPP uniquely combines localized quintic splines with a speed-profile optimizer to generate smooth, dynamically feasible trajectories that prioritize obstacle avoidance, passenger comfort, and strict adherence to road constraints such as lane boundaries. By leveraging real-time data from the vehicle's sensor fusion module, LSPP accurately interprets the positions of nearby vehicles and obstacles, producing safe paths that are passed to the Model Predictive Control (MPC) module for precise execution. Simulations show LSPP reduces lateral jerk by 30% and computation time by 25% compared to Bézier-based methods, confirming enhanced comfort and efficiency. Extensive testing across diverse highway scenarios further demonstrates LSPP's superior performance in trajectory smoothness, lane-keeping, and responsiveness over traditional approaches, validating it as a compelling solution for safe, comfortable, and efficient autonomous highway driving.

1. Introduction

With the rapid advancement of intelligent transportation systems [1] and automobile technology [2], fully automated vehicles have garnered significant attention from researchers due to their broad potential applications [3]. These vehicles rely on multidisciplinary frameworks, with perception [4], decision-making [5], and control [6] serving as the three core pillars of their software architecture. Among these, path-planning and path-tracking are central to ensuring reliable autonomous operation [7]. Effective path planning is vital for safety, comfort [8], and vehicle controllability in extreme and high-speed conditions [9].

In the context of autonomous driving, both path and motion planning are critical. Path planning determines a collision-free route from the current location to a destination while considering road geometry, traffic rules, and obstacles. Motion planning, in contrast, focuses on real-time trajectory generation that ensures kinematic feasibility, smooth movement, and safe execution of the path. While the former handles strategic routing, the latter manages the tactical control necessary for safe and comfortable vehicle operation under dynamic conditions. This paper concentrates on motion planning, especially for highway scenarios, where high-speed navigation, lane-keeping, and rapid responsiveness to traffic changes are crucial. The process integrates data on the ego vehicle's state

(e.g., speed, orientation), environmental context (e.g., obstacles and traffic), and road conditions to generate a collision-free, smooth, and optimized trajectory [10].

1.1. Components of the Path Planning

The motion planning process in autonomous vehicles builds upon multiple interconnected modules, from environmental mapping to control execution. Key components include:

1. Perception and environment mapping: acquisition of sensor data (LiDAR, radar, GPS, cameras) [11]. Object and pedestrian detection/tracking [12]-[14]. Construction of an up-to-date environment map.
2. Localization: estimation of vehicle position using GPS, IMU, and map data [15], [16].
3. Route Planning: High-level navigation with global path planning and waypoint generation [17].
4. Behavior planning: evaluation of driving scenarios (merges, lane changes) and behavior selection [5], [18].
5. Path generation: feasibility checks and generation of smooth paths using spline or polynomial techniques [19], [20].
6. Trajectory planning: detailed trajectory generation with optimization and collision avoidance strategies [21].
7. Speed profile generation: optimal velocity planning with comfort and safety considerations [21], [22].
8. Control interface: translation of trajectory into control commands with feedback monitoring [18], [19].

Together, these components operate in a feedback loop to ensure adaptive and efficient vehicle navigation [23]. Among them, modules 4 through 8-behavior planning to control interface-are most representative of motion planning, which is the focus of this study. These modules enable real-time responsiveness and precise execution of high-level plans while adhering to vehicle dynamics and safety constraints.

1.2. Background and Literature Review

Several motion planning techniques have been proposed and can be categorized into the following five key approaches:

1.2.1. Space Configuration

This approach builds an occupancy grid map based on the egocar's state and the positions of surrounding obstacles. The environment is discretized into grid cells, each classified as occupied or free. Free cells are used to generate potential paths using algorithms such as Voronoi diagrams [24], lattice structures, and sampling-based methods. While these methods are computationally efficient, they often neglect vehicle dynamics, leading to infeasible trajectories.

1.2.2. Path-Finders

Path-finders like A*, Dijkstra, and Rapidly-exploring Random Trees (RRT) aim to discover the optimal path between start and goal points, minimizing cost functions like distance or time [25]. A* and Dijkstra rely on complete knowledge of the environment, while RRTs handle unknown terrains more flexibly. However, like space configuration methods, they do not consider the dynamic constraints of real vehicles, limiting their applicability in motion planning.

1.2.3. Attractive and Repulsive Fields

These methods employ artificial potential fields where attractive forces guide the vehicle toward the target and repulsive forces steer it away from obstacles [26]. The resultant force dictates the vehicle's movement. Although intuitive and computationally light, these methods are prone to local minima and can yield unstable trajectories in complex environments.

1.2.4. Curves

Curves such as splines, Bezier curves [28], and clothoids are used to model smooth local paths based on vehicle state and road geometry [27]. Two main strategies exist:

- Point-Free: Generates continuous, kinematically feasible trajectories.
- Point-to-Point: Connects specific waypoints with smooth curves.

While these curves yield realistic paths, validating each trajectory for dynamic feasibility and obstacle avoidance is computationally intensive, posing challenges for real-time execution.

1.2.5. Artificial Intelligence Schemes

AI techniques such as fuzzy logic, neural networks [29], and evolutionary algorithms (e.g., PSO, GA, ACO) are employed for flexible, adaptive path planning without requiring exact models. They excel in handling uncertainty and non-linearity but typically demand substantial computational resources, making real-time implementation difficult. A comprehensive review of these methods is provided in [18].

Hybrid approaches are increasingly adopted to harness the strengths of multiple methods [20]. For example, combining splines with numerical optimization ensures trajectory feasibility and objective optimization [28]-[30]. Similarly, integrating Artificial Potential Fields with MPC improves control performance [31]. However, these combinations increase system complexity and computational load, often compromising real-time feasibility.

Despite progress, challenges persist in developing motion planners that balance simplicity, safety, robustness, and responsiveness under realistic operating conditions. Many existing methods struggle to incorporate dynamic constraints effectively while maintaining low-latency performance and smooth vehicle operation.

To address this, the current study proposes the Local Spline-based Path Planner (LSPP)—a novel framework that merges rule-based waypoint generation with spline curve fitting and MPC-based control. This integration enables the generation of dynamically feasible, smooth, and safe local trajectories aligned with global navigation paths. The LSPP aims to meet four critical criteria:

- Safety: Navigates dynamic and uncertain environments without collisions.
- Comfort: Minimizes jerk and abrupt motion for smooth rides.
- Accuracy: Enhances trajectory tracking precision.
- Real-time Performance: Maintains high responsiveness using computationally efficient modules.

By targeting these goals, this study contributes a practical and modular solution to the motion planning challenges faced by autonomous vehicles, particularly in highway driving scenarios.

2. Method

2.1. Foundation of Motion Planning

The following sections shed the light on the foundation of the proposed motion planner:

2.1.1. Path Planning Architecture

Fig. 1 represents a comprehensive architecture of a path planning and control system for an autonomous vehicle, showcasing the flow of information between various modules and components that contribute to trajectory planning and vehicle control.

The system initiates with the mission planner module (MPM), which is supplied with inputs from both the GPS and a global digital street map. These inputs enable the MPM to generate a high-level trip route. This route is then forwarded to the Rule-Based Localized Path Planner (RBLPP), which uses this information to extract a nearby segment of the route (according to the current ego car

coordinates) and generate for this segment approximate waypoints, defining the start, end, and some midpoint positions for it. The RBLPP also receives critical information on moving and stationary local objects from the Object Detection Module (ODM), which integrates data from multiple sensors, including Cameras, Radars, Lidars, and Sensor Fusion Algorithms. Each of these sensors feeds its data into the sensor fusion module, which processes and combines the data to form a coherent understanding of the vehicle's surrounding environment.

The approximate waypoints generated by the RBLPP are subsequently refined by the Spline-Trajectories Generator, which produces more accurate trajectories for the vehicle to follow. These refined trajectories are then supplied to the MPC (Model Predictive Control) Controller, which is responsible for generating the necessary control commands to actuate the vehicle. The MPC outputs steering commands and throttle/brake commands, which are sent to the vehicle's actuators, ensuring precise and responsive control of the egocar.

The modules within the green oval in Fig. 1 represent the motion planner which is the main focus of this paper, which determines what behavior the vehicle should exhibit at any point in time (e.g. stopping at a traffic light or intersection, changing lanes, accelerating, or making a left turn onto a new street, etc.).

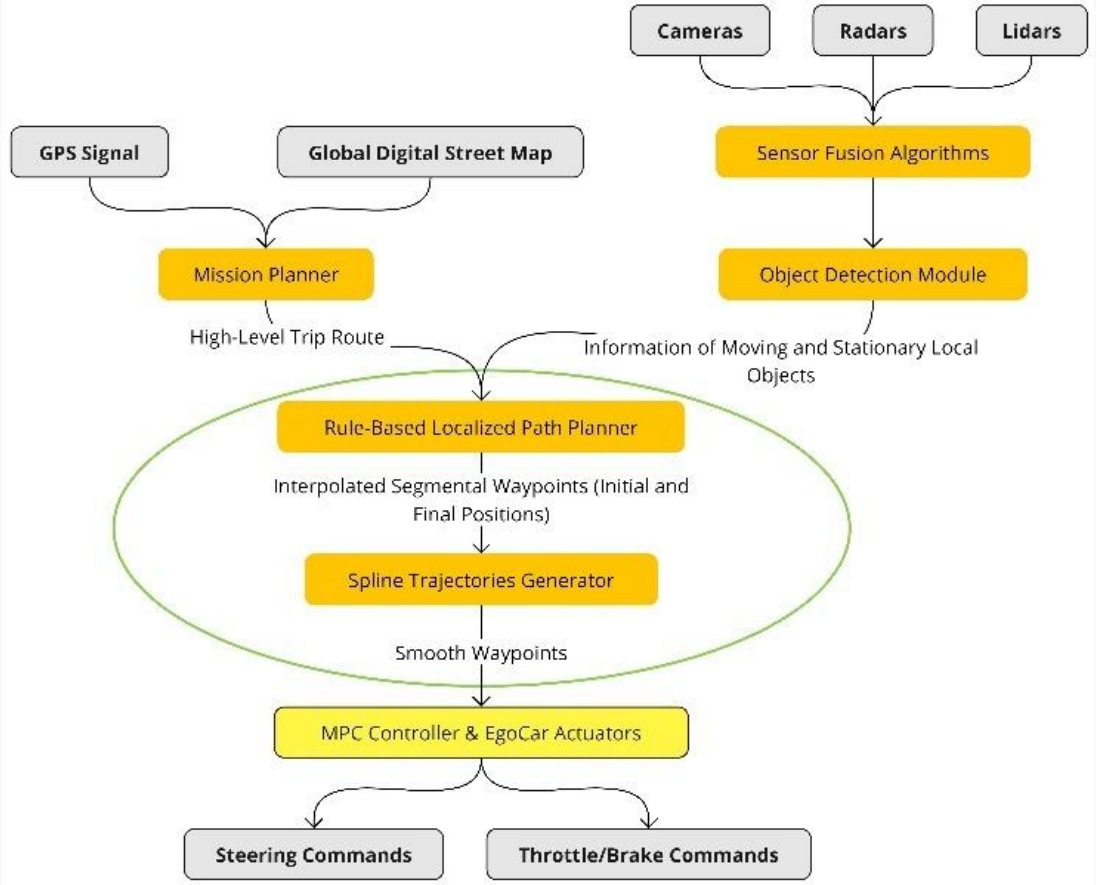

Fig. 1. Architecture of the autonomous vehicle path planning

2.1.2. Key Characteristics of Quintic Polynomials

A quantic spline polynomial is a piecewise-defined polynomial function used in interpolation and approximation of data to create smooth curves that pass through or near a set of data points. They are characterized as follows:

1. Piecewise Definition: A spline is defined by multiple polynomial segments, each valid over a specific interval. For example, a quantic spline is composed of 5th polynomials defined piecewise between each pair of waypoints

$$(x_i, y_i)\ and\ (x_{i+1}, y_{i+1})$$
$$y_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 + e_i(x - x_i)^4 + f_i(x - x_i)^5 \quad (1)$$
$$for\ x \in [x_i, x_{i+1}]$$

2. Continuity and Smoothness: Splines are constructed to ensure a certain level of smoothness at the points where the polynomial pieces join (called knots). For quantic splines, the first, second, third, and fifth derivatives are usually continuous across the intervals, making the transition between segments very smooth.
3. Local Control: Each segment of the spline is influenced mainly by its local data points (waypoints received from the RBLPP), allowing for localized changes in the shape of the curve without significantly affecting the entire curve.
4. Boundary Conditions: When constructing splines, additional conditions are needed at the endpoints to uniquely determine the spline. Common boundary conditions include specifying the values of derivatives up to the 4th order at the endpoints, and higher order or derivatives are set to zero.

For $n + 1$ data points $(x_0, y_0), (x_0, y_0), \ldots, (x_n, y_n)$, a quantic spline S_n consists of n quantic polynomials given by Eq. (1) defined on each interval $[x_i, x_{i+1}]$. The coefficients $a_i, b_i, c_i, d_i, e_i, and\ f_i$ are determined by solving a system of equations that enforce: 1) The spline passing through or very close to the data points. 2) Continuity of the spline and its first, second, third, and fourth derivatives at each knot. 3) The chosen boundary conditions.

2.1.3. The Proposed Motion Planner

Behavior planning involves three key components: 1) predicting the behavior of obstacles, including other vehicles on the road, 2) making high-level decisions on the appropriate actions to achieve the desired goal safely and efficiently, such as slowing down, accelerating, turning, or changing lanes, and 3) generating trajectories that translate these decisions into a feasible path, typically represented as a series of waypoints or a mathematical formula, for the vehicle's control system to execute [32]. The proposed LSPP protocol operates as follows:

1. Interpolate waypoints for a nearby section (30 to 50 meters long) of the route provided by the MPM.
2. Determine the state of the egocar, including its position, orientation, velocity, and acceleration.
3. Generate a set of predicted trajectories for each surrounding vehicle using data received from the ODM.
4. Identify the available states for the egocar, such as “keep lane,” “change lanes to the right,” or “change lanes to the left.”
5. For each available state, generate a target end state for the egocar and create multiple randomized potential trajectories by slightly perturbing elements of the target state. These “Jerk-Minimized Trajectories” (JMTs) [33] are quintic spline polynomials derived from the current initial and desired final values of position, velocity, and acceleration [34].
6. Evaluate each potential trajectory using a set of cost functions that reward efficiency and penalize factors such as collisions, excessive jerks, or exceeding speed limits.
7. Select the trajectory with the lowest cost.
8. Assess this trajectory at the next several 20-millisecond intervals (the rate at which the data is received from the sensor fusion module) [35].
9. Append these evaluations to the retained portion of the previous path.
10. Send the updated path to the MPC controller for execution.

The Jerk Minimizing Trajectory (JMT) algorithm is implemented in C++ [36] to generate smooth trajectories that minimize sudden changes in acceleration, resulting in more comfortable and safer driving.

The JMT calculation involves finding the coefficients of a polynomial (5th order in quintic spline) that describe the vehicle's path from a given start state to an end state over a specified time T (usually indicates the required speed profile). The key components of the algorithm can be portrayed as follows:

1. Input Parameters:
   - Start: A vector of three elements [position, velocity, acceleration] representing the initial state of the vehicle.
   - End: A vector of three elements [position, velocity, acceleration] representing the final state of the vehicle.
   - T: The time duration over which the vehicle should move from the start state to the end state.
2. Output:
   - A vector of coefficients for a 5th-degree polynomial of the form: $p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$.
   - These coefficients are determined to minimize jerk over the time T.
3. Calculation: A system of equations is set up using the boundary conditions provided by the start and end states. The C++ Eigen library [36] is used to solve this system and find the polynomial coefficients.

2.2. Implementation of the LSPP

The following sections provide an in-depth overview of the implementation tasks for the proposed algorithm:

2.2.1. The Finite State Machine

While driving on the highway, the egocar typically considers several key driving states. These states help the car adapt its behavior based on the surrounding environment, including the positions of other vehicles and obstacles. In the design of the proposed LSPP algorithm, the states that the egocar can occupy are listed in Table 1.

Moreover, cost functions are used by the egocar, to evaluate and optimize the decision-making process in real-time. These functions assign a numerical value (or cost) to different possible actions, helping the system select the most optimal choice [34]. Here in Table 2 is a summary of the cost functions used to transition between various states while driving on the highway.

The decision-making process in the egocar employs a Finite State Machine (FSM) (illustrated in Fig. 2) with the seven states that have been specified earlier in Table 1 and transitions based on certain cost functions (Table 2) and triggering transitional conditions (TTCs) (listed in Table 3). Each state represents a different driving behavior, and the transitions depend on conditions like lane preference, proximity to obstacles, and the need for speed adjustments. The FSM in Fig. 2 is illustrated as a directed graph with nodes representing states and arrows representing transitions between states, labeled with the triggering conditions [37].

2.2.2. The Vehicle Kinematic Model

This paper employs the Kinematic Bicycle Model [22] to simulate the behavior of the autonomous vehicle. The model is represented by nonlinear continuous-time equations (2), which describe its dynamics in an inertial reference frame, as illustrated in Fig. 3. In this model, x and y represent the coordinates of the vehicle's center of mass in an inertial reference frame (X, Y). The variable ψ denotes the vehicle's heading angle relative to the inertial frame, and v indicates its speed.

The parameters lf and lr correspond to the distances from the center of mass to the front and rear axles, respectively. β defines the angle between the velocity of the center of mass and the vehicle's longitudinal axis, while a is the acceleration in the same direction as the velocity. The control inputs are the front and rear steering angles, δ_f, and δ_r, respectively. In most vehicles, the rear wheels are not steerable, so δ_r is assumed to be zero. Additionally, ω represents the angular velocity of the steering.

Table 1. Key driving states

#	Ego Car State	Description	Possible Transitions	Triggering Conditions	Relevant Cost Functions
1	Keep Lane	The ego car stays in its current lane, maintaining a safe distance from vehicles.	- Change Lane Left - Change Lane Right - Emergency Stop - Speed Up - Slow Down - Keep Lane	- Safe distance maintained - No vehicles blocking the lane - Speed close to target	- Speed cost (maintain target speed) - Collision cost (avoid rear-end collisions) - Lane preference cost
2	Change Lane Left	The ego car shifts to the left lane, typically to overtake a slower vehicle.	- Keep Lane - Emergency Stop	- The left lane is clear - Left lane offers higher speed efficiency - Safe to maneuver	- Collision cost (check for vehicles in the left lane) - Jerk cost (minimize jerk) - Overtaking efficiency cost
3	Change Lane Right	The ego car shifts to the right lane, typically to make way for faster traffic.	- Keep Lane - Emergency Stop	- The right lane is clear - Need to avoid slower or obstructed vehicles -Approaching an exit	- Collision cost (check for vehicles in the right lane) - Lane preference cost - Jerk cost (minimize jerk)
4	Prepare for Exit	The ego car prepares to leave the highway by moving into the exit lane.	- Take Exit - Change Lane Right - Emergency Stop	- Exit approaching within a certain distance - The right lane is clear to merge into an exit lane	- Lane preference cost (moving toward the exit) - Speed cost (maintain appropriate speed for exit) - Fuel Efficiency cost
5	Emergency Stop	The ego car decelerates rapidly to avoid a collision or stop for an obstacle.	- Keep Lane - Change Lane Left/Right (if possible)	- Obstacle detected ahead - Collision is imminent - Severe traffic jam	- Collision cost (avoidance) - Jerk cost (smooth deceleration) - Safety margin cost (maintain safe stopping distance)
6	Speed Up	The ego car increases speed to maintain the target speed or overtake other cars.	- Keep Lane - Change Lane Left/Right - Emergency Stop	- Lane ahead is clear - Ego car moving slower than the target speed	- Speed cost (to reach target speed) - Fuel efficiency cost (avoid unnecessary speeding) - Overtaking efficiency cost
7	Slow Down	The ego car decreases speed to maintain safety and avoid collisions.	- Keep Lane - Change Lane Left/Right - Emergency Stop	- Slower vehicle detected ahead - Congested traffic - Construction or obstacles ahead	- Collision cost (avoid rear-end collisions) - Jerk cost (smooth deceleration) - Safety margin cost

Compared to higher-fidelity vehicle models [32], the kinematic bicycle model offers easier system identification, requiring only two parameters, lf, and lr, which simplifies adapting the same controller or path planner to vehicles with different wheelbases.

The presented vehicle model serves as a constraint to ensure that the generated quintic path adheres to the vehicle's kinematic limitations and is also integrated into the development of the Model Predictive Controller (MPC) [23], enabling the vehicle to accurately execute the planned trajectory.

Table 2. Cost functions for transition between different states

Cost Function	Description	Formula	States Where Applied	Purpose in Transitioning Between States
Speed Cost	Penalizes deviation from the target speed, either too slow or too fast.	$C_{speed} = \frac{\lvert v_{ego} - v_{target} \rvert}{v_{target}}$	- Keep Lane - Speed Up - Slow Down	Ensures the ego car maintains an optimal speed, transitioning between speeding up or slowing down as needed.
Collision Cost	Assigns high penalties for potential collisions with vehicles, obstacles, or other hazards.	$C_{collision} = \frac{1}{d_{min}^2}$, where d_{min} is the minimum distance to obstacles	- Keep Lane - Change Lane Left/Right - Emergency Stop	Avoids collisions by influencing decisions to either change lanes, slow down, or stop in case of danger ahead.
Lane Preference Cost	Penalizes staying in undesirable lanes (e.g., slower lanes) and rewards transitions to faster lanes.	$C_{lane} = P_{lane}$, where P_{lane} is a penalty value for staying in a less efficient lane	- Change Lane Left - Change Lane Right - Prepare for Exit	Guides lane changes based on lane desirability, either for overtaking slower vehicles or preparing for an exit.
Jerk Cost	Penalizes high jerk (rapid changes in acceleration) to ensure smooth driving and passenger comfort.	$C_{jerk} = \int_0^T \left(\frac{d^3x(t)}{dt^3}\right)^2 dt$, over time interval T	- Change Lane Left/Right - Speed Up - Slow Down - Emergency Stop	Ensures smooth transitions during lane changes, acceleration, deceleration, or emergency stops.
Overtaking Efficiency Cost	Rewards efficient overtaking and penalizes unnecessary or unsafe lane changes.	$C_{overtake} = \frac{1}{t_{pass}}$, where t_{pass} is the time taken to overtake	- Change Lane Left - Change Lane Right	Guides the ego car in making efficient lane changes when overtaking slower vehicles, avoiding unnecessary maneuvers.
Safety Margin Cost	Ensures the ego car maintains a safe distance from other vehicles and obstacles.	$C_{safety} = \frac{1}{d_{ego} - d_{safe}}$, where d_{safe} is the safe distance	- Keep Lane - Slow Down - Emergency Stop	Promotes safe driving by guiding the ego car to slow down or stop if the distance to an obstacle becomes too small.
Fuel Efficiency Cost	Penalizes unnecessary acceleration or speeding, promoting fuel efficiency.	$C_{feul} = \frac{a^2}{v_{ego}}$, where $a$is acceleration and v_{ego} is the car's speed	- Speed Up - Slow Down	Encourages energy-efficient driving by optimizing speed adjustments and minimizing excessive acceleration.

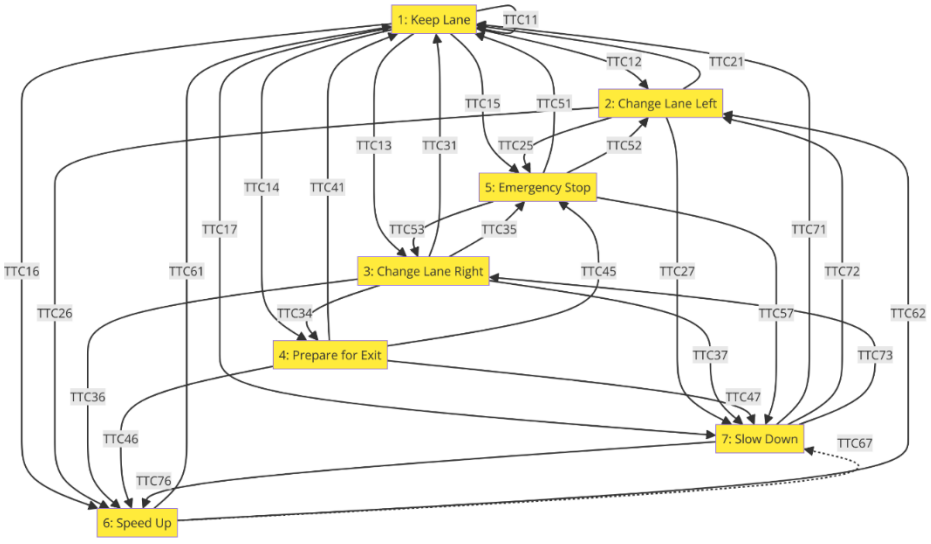

Fig. 2. Finite state machine of the autonomous vehicle motion planner

Table 3. Triggering transitional conditions (TTCs)

#	Code	Description	Description of the Triggering Transitional Conditions (TTCs)
1	TTC11	Stay in State 1	1. Safety Margin Cost is very low, and 2. Lane Preference Cost is very low, and 3. Speed Cost is very low.
2	TTC12	Transit from State 1 → State 2	1. Collision Cost (check for vehicles in the left lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost is high, or 4. Jerk's Cost is high.
3	TTC13	Transit from State 1 → State 3	1. Collision Cost (check for vehicles in the right lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost is high, or 4. Jerk's Cost is high.
4	TTC14	Transit from State 1 → State 4	1. The Exit is approaching within a certain distance.
5	TTC15	Transit from State 1 → State 5	1. Collision Cost is very high, or 2. Safety Margin Cost is very high, or 3. Jerk Cost is very high.
6	TTC16	Transit from State 1 → State 6	1. Collision Cost (check for vehicles in the same lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost (to reach target speed) is high, and 4. Fuel efficiency cost is low or moderate (avoid unnecessary speeding).
7	TTC17	Transit from State 1 → State 7	1. Collision Cost (avoid rear-end collisions) is high, and 2. Overtaking Efficiency Cost is high, and 3. Jerk Cost (smooth deceleration) is moderate, and 4. Safety Margin Cost is moderate to high.
8	TTC21	Transit from State 2 → State 1	1. Safety Margin Cost is very low, and 2. Lane Preference Cost (current lane) is very low, and 3. Speed Cost is very low.
9	TTC25	Transit from State 2 → State 5	1. Collision Cost is very high, or 2. Safety Margin Cost is very high, or 3. Jerk Cost (deceleration) is very high.
10	TTC26	Transit from State 2 → State 6	1. Collision Cost (check for vehicles in the same lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost (to reach target speed) is high, and 4. Fuel efficiency cost is low or moderate (avoid unnecessary speeding).
11	TTC27	Transit from State 2 → State 7	1. Collision Cost (avoid rear-end collisions) is high, and 2. Overtaking Efficiency Cost is high, and 3. Jerk Cost (smooth deceleration) is moderate, and 4. Safety Margin Cost is moderate to high.
12	TTC31	Transit from State 3 → State 1	1. Safety Margin Cost is very low, and 2. Lane Preference Cost (current lane) is very low, and 3. Speed Cost is very low.
13	TTC34	Transit from State 3 → State 4	1. The Exit is approaching within a certain distance.
14	TTC35	Transit from State 3 → State 5	1. Collision Cost is very high, or 2. Safety Margin Cost is very high, or 3. Jerk Cost is very high.
15	TTC36	Transit from State 3 → State 6	1. Collision Cost (check for vehicles in the same lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost (to reach target speed) is high, and 4. Fuel efficiency cost is low or moderate (avoid unnecessary speeding).

16	TTC37	Transit from State 3 → State 7	1. Collision Cost (avoid rear-end collisions) is high, and 2. Overtaking Efficiency Cost is high, and 3. Jerk Cost (smooth deceleration) is moderate, and 4. Safety Margin Cost is moderate to high.
17	TTC41	Transit from State 4 → State 1	1. Exit the highway is not possible (exit skipped or exit lane is closed) and 2. Safety Margin Cost is very low, and 3. Lane Preference Cost (current lane) is very low.
18	TTC45	Transit from State 4 → State 5	1. Collision Cost is very high, or 2. Safety Margin Cost is very high, or 3. Jerk Cost (deceleration) is very high.
19	TTC46	Transit from State 4 → State 6	1. Collision Cost (check for vehicles in the same lane) is very low, and 2. Speed Cost (to reach target speed) is high, and 3. Fuel efficiency cost is low or moderate (avoid unnecessary speeding).
20	TTC47	Transit from State 4 → State 7	1. Collision Cost (avoid rear-end collisions) is high, and 2. Safety Margin Cost is moderate to high, or 3. Jerk Cost (smooth deceleration) is moderate.
21	TTC51	Transit from State 5 → State 1	1. Safety Margin Cost is very low, and 2. Lane Preference Cost (current lane) is very low, and 3. Speed Cost is very high.
22	TTC52	Transit from State 5 → State 2	1. Collision Cost (check for vehicles in the left lane) is very low, and 2. Safety Margin Cost (current lane) is high, and 3. Lane Preference Cost (current lane) is high, and 4. Lane Preference Cost (left lane) is low, and 5. Speed Cost is high.
23	TTC53	Transit from State 5 → State 3	1. Collision Cost (check for vehicles in the right lane) is very low, and 2. Safety Margin Cost (current lane) is high, and 3. Lane Preference Cost (current lane) is high, and 4. Lane Preference Cost (right lane) is low, and 5. Speed Cost is high.
24	TTC57	Transit from State 5 → State 7	1. Safety Margin Cost is moderate, and 2. Lane Preference Cost (current lane) is moderate, and 3. Speed Cost is high.
25	TTC61	Transit from State 6 → State 1	1. Safety Margin Cost is very low, and 2. Lane Preference Cost (current lane) is very low, and 3. Speed Cost is very low.
26	TTC62	Transit from State 6 → State 2	1. Collision Cost (check for vehicles in the left lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost is high, or 4. Jerk Cost is moderate to high, and 5. Safety Margin Cost (current lane) is high.
27	TTC67	Transit from State 6 → State 7	1. Safety Margin Cost is moderate, and 2. Lane Preference Cost (current lane) is moderate, and 3. Speed Cost is from low to moderate.
28	TTC71	Transit from State 7 → State 1	1. Safety Margin Cost is high, and 2. Lane Preference Cost (left and right lanes) is high, and 3. Collision Cost is moderate to high.
29	TTC72	Transit from State 7 → State 2	1. Collision Cost (check for vehicles in the left lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost is high, or 4. Jerk's Cost is high.

30	TTC73	Transit from State 7 → State 3	1. Collision Cost (check for vehicles in the right lane) is very low, and 2. Overtaking Efficiency Cost is low, and 3. Speed Cost is high, or 4. Jerk's Cost is high.
31	TTC76	Transit from State 7 → State 6	1. Collision Cost is low, and 2. Safety Margin Cost is low, or 3. Jerk Cost is low to moderate.

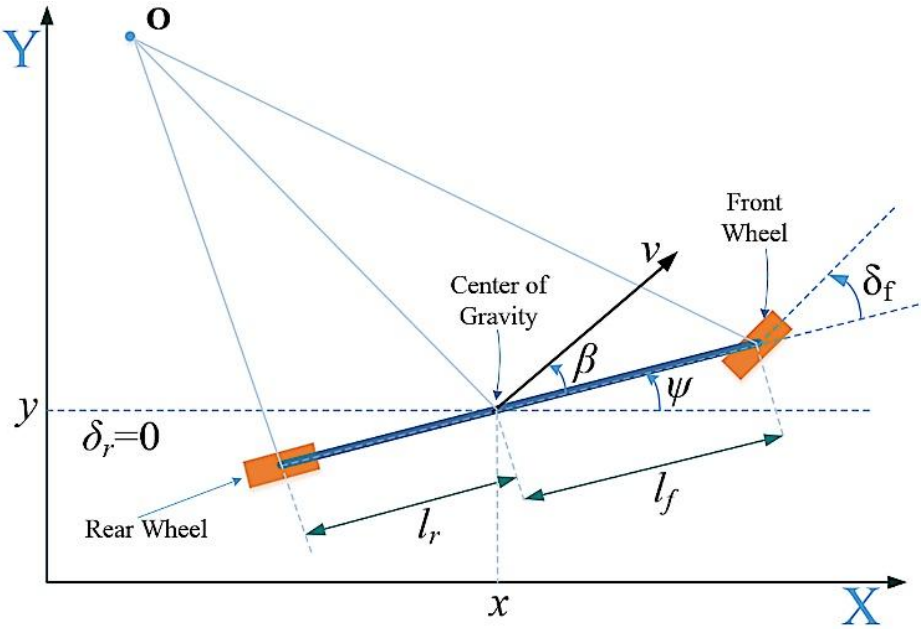

Fig. 3. The kinematic bicycle model

$$\begin{aligned} \dot{x} &= v * \cos(\psi + \beta) \\ \dot{y} &= v * sin\,(\psi + \beta) \\ \dot{\psi} &= \frac{v}{l_r} * \sin\,(\beta) \\ \dot{v} &= a \\ \beta &= tan^{-1}\left(\frac{l_r}{l_f + l_r} * tan(\delta_f)\right) \\ \dot{\delta_f} &= \omega \end{aligned} \tag{2}$$

2.2.3. Frenet Coordinates for Autonomous Vehicle Motion Planning

Frenet coordinates are used in path planning and trajectory tracking for autonomous vehicles, where the vehicle's position is expressed relative to a reference path (often the road or highway centerline), simplifying calculations for trajectory following and obstacle avoidance and proven effective in reducing computational complexity [38]. Frenet coordinates as shown in Fig. 4 consist of two main components:

- Longitudinal Coordinate (s): The arc length along the reference path from a fixed starting point to the projection of the vehicle's position onto the path. It represents the vehicle's progress along the path.
- Lateral Coordinate (d): The perpendicular distance from the reference path to the vehicle's position, representing the deviation of the vehicle from the path [39].

Together, (s, d) in Frenet coordinates allow the vehicle's movement to be decomposed into motion along the path (s) and deviation from it (d), helping streamline path-following and obstacle avoidance calculations.

Frenet coordinates offer significant advantages as they automatically adapt to path curvature, ease navigation through turns, and enable efficient trajectory generation for lane-keeping, lane changes, and obstacle avoidance by allowing easy manipulation of lateral offsets. Additionally, safety constraints, such as lane boundaries, can be conveniently expressed as limits on the lateral coordinate, streamlining adherence checks.

To convert Frenet coordinates (s, d) to Cartesian coordinates (x, y):

- Identify the reference point on the path: Find the point on the reference path at arc length s from the starting point, which serves as the baseline for the lateral offset.
- Determine the tangent and normal vectors: calculate the tangent vector at (xref, yref) on the reference path. This can be derived from the path's derivative or direction at s.
- Apply the Lateral Offset: Move d units along the normal vector to obtain (x, y) in Cartesian coordinates:

$$\begin{aligned} x &= x_{ref} + d.n_x \\ y &= y_{ref} + d.n_y \end{aligned} \tag{3}$$

where n_x, n_y are components of the normal vector at the reference point.

To convert Cartesian coordinates (x, y) to Frenet coordinates (s, d):

Project the Point onto the Path: Identify the nearest point on the reference path (xref, yref) to (x, y), then calculate the arc length s along the path from the starting point to this nearest point using the following equation:

$$s = \int_0^{x_{ref}} \sqrt{\left(\frac{dx}{ds}\right)^2 + \left(\frac{dy}{ds}\right)^2}\, ds \tag{4}$$

Calculate the Lateral Distance d: Compute the perpendicular distance from (x, y) to the nearest point on the path (xref, yref), giving the lateral offset d as in Equation (5). Positive or negative values of d indicate the side of the path relative to the driving direction:

$$\begin{aligned} d &= \sqrt{\left(x - x_{ref}\right)^2 + \left(y - y_{ref}\right)^2} \\ d &= sign\left(\left(y - y_{ref}\right).\frac{dx}{ds} - \left(x - x_{ref}\right).\frac{dy}{ds}\right).\sqrt{\left(x - x_{ref}\right)^2 + \left(y - y_{ref}\right)^2} \end{aligned} \tag{5}$$

2.2.4. Implementation of the Quintic Polynomial Trajectory

To design a quintic polynomial trajectory (5th-order polynomial) for a vehicle traveling between two waypoints p_1 and p_2, you must account for the Kinematic Bicycle Model parameters such as position, velocity, and acceleration at the initial and final states. A quintic polynomial ensures smooth transitions by controlling position, velocity, and acceleration, which aligns well with vehicle dynamics and provides a drivable path. Employed step-by-step implementation technique:

A quintic polynomial for a 1D path (e.g., $x(t)$) is given by:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{6}$$

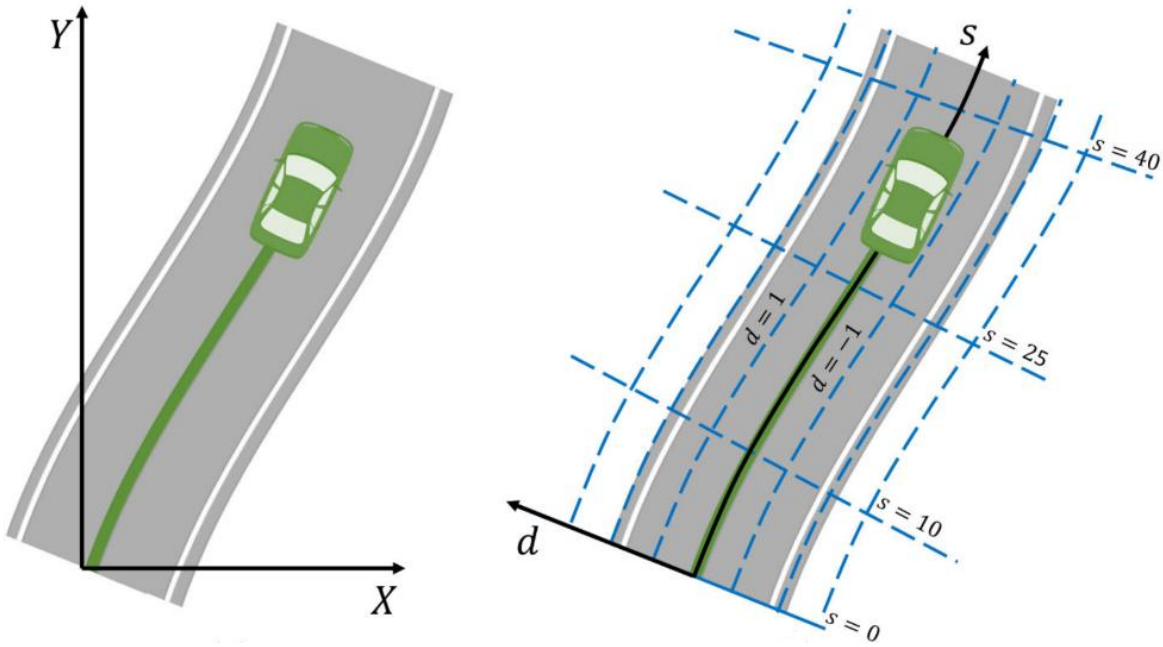

Fig. 4. The Cartesian coordinates (X, Y) versus the frenet coordinates (s, d) [40]

This trajectory describes the position $x(t)$ as a function of time t. Another polynomial is needed for y-coordinate, to generate the complete 2D path as follows:

$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \quad (7)$$

To compute the polynomial coefficients, the boundary conditions for both the starting and ending points are needed. For each coordinate (e.g., x and y), you need:

- At $t = 0$ (start at p_1):
  - $x(0) = x_1, y(0) = y_1$ (Initial position)
  - $\dot{x}(0) = v_1 \cos(\psi_1)$, $\dot{y}(0) = v_1 \sin(\psi_1)$ (Initial velocity)
  - $\ddot{x}(0) = a_1 \cos(\psi_1), \ddot{y}(0) = a_1 \sin(\psi_1)$ (Initial acceleration)
- At $t = T$ (end at p_2):
  - $x(T) = x_2, y(T) = y_2$ (Final position)
  - $\dot{x}(T) = v_2 \cos(\psi_2)$, $\dot{y}(T) = v_2 \sin(\psi_2)$ (Final velocity)
  - $\ddot{x}(T) = a_2 \cos(\psi_2), \ddot{y}(T) = a_2 \sin(\psi_2)$ (Final acceleration)

where:

- $x_1, y_1, \psi_1, v_1, a_1$: Initial state (position, heading, speed, acceleration).
- $x_2, y_2, \psi_2, v_2, a_2$: Final state (position, heading, speed, acceleration).

These boundary conditions ensure smooth transitions and match the vehicle's kinematic constraints. Each polynomial has 6 unknown coefficients (a_0 to a_5). For both $x(t)$ and $y(t)$, 6 equations are needed from the boundary conditions to solve for the coefficients. For $x(t)$ the following are the 6 boundary equations:

- 2 equations for Position: $x(0) = a_0, x(T) = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + a_4 T^4 + a_5 T^5$
- 2 equations for Velocity: $\dot{x}(0) = a_1, \dot{x}(T) = a_1 + 2a_2 T + 3a_3 T^2 + 4a_4 T^3 + 5a_5 T^4$

- 2 equations for Acceleration: $\ddot{x}(0) = 2a_2, \ddot{x}(T) = 2a_2 + 6a_3T + 12a_4T^2 + 20a_5T^3$

A similar system of 6 boundary equations for $y(t)$ is constructed to solve for (b_0 to b_5). The boundary conditions are organized into a linear system of equations and solved for the coefficients (a_0 to a_5, b_0 to b_5) using a C++ numerical solver, matrix, and vector operations package "Eigen".

After the quantic spline is constructed between waypoints p_1 and p_2, it is to calculate the position, velocity, and acceleration at discrete time intervals between $[0, T]$ resulting in new more dense waypoints. The process is repeated between waypoints p_2 and p_3, etc. Then, the new waypoints will be fed into the Model Predictive Controller (MPC) that uses the generated trajectory to compute actuator commands for steering and throttle control. The MPC continuously optimizes control inputs (steering and throttle) based on the trajectory and real-time feedback from the vehicle's sensors. It minimizes deviation from the path while adhering to the vehicle's kinematic constraints.

The generated trajectory must align with the vehicle's maximum speed, acceleration, steering angle, and angular velocity limits as follows:

- Velocity Limit: Ensure the trajectory's speed never exceeds the vehicle's maximum speed v_{max}. Constraint: $\sqrt{\dot{x}^2(t) + \dot{y}^2(t)} \leq v_{max}$.
- Acceleration Limit: The acceleration should not exceed a threshold a_{max} to ensure safety and comfort. Constraint: $\sqrt{\ddot{x}^2(t) + \ddot{y}^2(t)} \leq a_{max}$.
- Steering Angle and Angular Velocity Limits: The trajectory must respect the maximum steering angle δ_f and the rate of change of the angle ω (angular velocity). Constraints: $|\delta_f| \leq \delta_{max}$ and $|\omega| \leq \omega_{max}$.
- The curvature κ(t) along the spline influences the required steering angle. If the curvature is too high, it may exceed the vehicle's steering capability. The curvature is calculated as:

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{3/2}} \tag{8}$$

Accordingly, the steering angle is calculated from curvature and the vehicle's kinematic model as:

$$\delta_f(t) = \tan^{-1}\left((l_f + l_r).\kappa(t)\right) \tag{9}$$

Ensuring that $|\delta_f| \leq \delta_{max}$ to confirm the steering is feasible.

- Enforce Smooth Transitions with Jerk Constraints: The jerk (third derivative of position) calculated by equation (7) must be limited to avoid sudden changes in acceleration, which can lead to discomfort and instability. High jerk values can also stress the actuators.

$$Jerk = \sqrt{\frac{d^3x(t)^2}{dt^3} + \frac{d^3y(t)^2}{d^3}} \tag{10}$$

The jerk constraint is to ensure that it remains below a comfortable threshold J_{max}.

2.2.5. Putting All Together

This pseudocode (ALGORITHM 1) provides a clear overview of the LSPP algorithm's steps, from initial path generation to real-time tracking and adaptation, illustrating its smooth and efficient approach to motion planning in structured autonomous driving environments.

Moreover, in Table 4 are examples of typical values for the parameters and constraints used in the LSPP algorithm. These values are based on standard autonomous vehicle settings for highway or

structured road environments [41]. These parameters are intended to support collision avoidance by adjusting the path without full replanning, ensuring smoother obstacle navigation and maintaining efficient traffic flow [42].

ALGORITHM 1. The LSPP Algorithm

```
Algorithm LSPP (StartState, GoalState, RoadConstraints, Obstacles, Δt, Horizon)
Input:
   StartState      // Initial position, velocity, acceleration of the egocar
   GoalState       // Desired end position, velocity, acceleration
   RoadConstraints  // Lane boundaries, speed limits, and kinematic limits (e.g., max steering, acceleration)
   Obstacles       // Position and velocity of dynamic/static obstacles detected by sensor fusion
   Δt            // Time step for trajectory update
   Horizon         // Prediction horizon for the trajectory
Output:
   SafePath        // Smooth, collision-free trajectory to follow within the defined horizon
Begin
1. Initialize Path:
   - Set CurrentState = StartState
   - Initialize SafePath as an empty list
2. Path Generation using Quintic Spline:
   - Calculate QuinticSpline (StartState, GoalState) based on boundary conditions:
      - Position, velocity, and acceleration at StartState and GoalState
      - Respect continuity up to the third derivative (jerk)
   - Store the generated spline in SafePath
3. Check Collision Avoidance and Kinematic Feasibility:
   - For each point P in SafePath:
      - If P violates any RoadConstraints (e.g., lane boundaries, speed limits, steering angles):
         - Mark P as unfeasible
      - For each obstacle O in Obstacles:
         - Calculate distance d between P and O
         - If d < SafeBufferDistance:
            - Mark P as a potential collision point
            - Proceed to AdjustPath
4. AdjustPath for Collision Avoidance (if any potential collision points are detected):
   - For each marked collision point P in SafePath:
      - Calculate a new GoalState that adjusts the spline to avoid the obstacle (e.g., by shifting lateral distance or
adjusting speed)
      - Generate a LocalizedSpline from CurrentState to new GoalState
      - Replace segment of SafePath around P with the new LocalizedSpline segment

5. Path Smoothing and Final Check:
   - Ensure smoothness of SafePath by recalculating segment continuity at adjustment points
   - If necessary, apply a smoothing function to mitigate abrupt changes near adjusted points
6. Real-time Tracking and Execution:
   - Send SafePath to Model Predictive Control (MPC) for real-time tracking
   - Monitor vehicle state every Δt:
      - If significant deviation from SafePath occurs due to dynamic changes:
         - Update StartState = CurrentState
         - Repeat from Step 2 (re-generate path based on new conditions)
End Algorithm
```

3. Results and Discussion

3.1. Simulation and Testing Results

The performance and robustness of the LSPP algorithm were evaluated through extensive simulations across a variety of realistic driving scenarios. Each scenario was designed to reflect the challenges commonly faced by autonomous vehicles, including lane-keeping, lane changes, obstacle avoidance, high-speed maneuvering, and handling stop-and-go traffic. A kinematic vehicle model was employed with constraints aligned to typical autonomous driving parameters, such as speed, acceleration, steering limits, and jerk constraints. To quantify the effectiveness of LSPP compared to established motion planning algorithms-namely A*, RRT, and Bezier Curve-based planning—key

metrics were measured, including trajectory smoothness, lateral deviation, collision avoidance rate, lane-keeping success, and execution time. A detailed analysis of the simulation results is presented in this section, highlighting the strengths of LSPP in providing a safe, efficient, and computationally feasible solution for real-time autonomous vehicle path planning.

Table 4. Typical values for the parameters and constraints used in the LSPP algorithm.

Parameter	Typical Value
Initial/Goal Velocity	80–100 *km/h* (22.2–27.8 *m/s*)
Acceleration Limits	±3 m/s^2
Jerk Limit	2–3 m/s^3
Lane Width	3.5 meters
Max Lateral Deviation	±0.3 meters
Speed Limits	60–120 *km/h* (16.7–33.3 *m/s*)
Steering Angle Limit	±25 degrees
Safe Distance Buffer	5 meters
Prediction Horizon (T)	3–5 seconds
Control Time Step (Δt)	0.1 seconds
Detection Range for Obstacles	50–100 meters
Obstacle Update Rate	10 Hz (every 0.1 seconds)
Path Adjustment Parameters for Collision Avoidance	Typical Value
Lateral Shift	0.5–1.0 meters
Speed Reduction	10–20% decrease
Angle Adjustment for Lane Changes	±2–5 degrees

3.1.1. Setting of Simulation Parameters

In the simulation of the LSPP algorithm for autonomous vehicles, the following parameters were used to model realistic driving dynamics and constraints. These values reflect typical limits and requirements in autonomous vehicle simulations:

3.1.1.1. Vehicle Dynamics Parameters

- Vehicle Speed Limits:
  - Maximum Speed: 120 km/h (33.3 m/s)
  - Minimum Speed: 0 km/h (stationary), as needed for stop-and-go scenarios
- Acceleration and Deceleration:
  - Maximum Acceleration: 3 m/s^2, to allow for efficient speed changes while maintaining passenger comfort.
  - Maximum Deceleration: -5 m/s^2, to ensure rapid stopping capability for emergency scenarios.
- Steering Constraints:
  - Maximum Steering Angle: ±25°, to limit lateral deviation within safe bounds during sharp turns.
  - Maximum Steering Rate (Angular Velocity): 60°/second, enabling smooth steering transitions without abrupt turns.
- Jerk Constraints:
  - Maximum Jerk: 2 m/s^3, to limit sudden changes in acceleration and ensure smooth trajectory transitions.

3.1.1.2. Path and Control Parameters

- Lane Width: 3.5 meters, consistent with standard highway lane width.
- Path Planning Horizon:
  - Prediction Horizon (T): 5 seconds (167 m for 120 km/h speed), allowing the vehicle to anticipate and respond to upcoming obstacles or turns.
  - Control Time Step (Δt): 0.1 seconds, providing precise control updates at each step.
- Curvature Constraints:
  - Minimum Turning Radius: 10 meters, simulating tight urban turns and curved highway segments.

3.1.1.3. Traffic and Environmental Parameters

- Traffic Density:
  - Sparse (Scenario 2): Limited vehicles, allowing free lane changes.
  - Moderate (Scenario 1): Vehicles moving at similar speeds, with space for controlled lane changes.
  - Congested (Scenario 5): High-density stop-and-go traffic to test low-speed handling and obstacle avoidance.
- Obstacle Characteristics:
  - Obstacle Appearance Distance: 30 meters ahead, to assess emergency stopping and avoidance capabilities.
  - Reaction Time for Emergency Scenarios: 1.5 seconds, a typical benchmark for real-time reaction in autonomous driving.

3.1.1.4. Control Algorithm-Specific Parameters for LSPP

- Cost Function Weights (LSPP-specific parameters):
  - Position Error Weight: 1.0, prioritizing precise path-following.
  - Heading Error Weight: 0.8, to align vehicle orientation with the desired trajectory.
  - Control Effort Weight: 0.5, minimizing control input variations to improve smoothness.
- Trajectory Constraints:
  - Lateral Deviation Limit: ±0.3 meters from the centerline, allowing for safe lane positioning without abrupt lateral moves.
- Safety Buffer:
  - Collision Avoidance Buffer: 5 meters, maintaining safe spacing to allow evasive actions if necessary.

These parameter values help ensure that the simulation of LSPP closely mimics real-world autonomous driving dynamics, enabling it to be tested for safety, responsiveness, and comfort across a variety of driving scenarios.

3.1.2. Simulation Scenarios and Results

To present the experimental simulation results, various experiments and scenarios are outlined to evaluate the effectiveness of the LSPP algorithm. These experiments simulate realistic driving conditions, including straight and curved highways, varying traffic densities, and emergency obstacles. Key performance indicators are used to measure LSPP's effectiveness, with results

compared against other standard motion planning algorithms, highlighting the strengths of the proposed approach.

To evaluate the LSPP algorithm, it is compared against three widely used motion planning algorithms:

1. *A* Algorithm*: A graph-based search algorithm that finds the shortest path to the target by optimizing a cost function [43]. It's effective for pathfinding but can be computationally expensive in complex environments [44].
2. *Rapidly-exploring Random Tree (RRT)*: A sampling-based algorithm that builds a tree to explore feasible paths from the start to the goal, focusing on avoiding obstacles [45]. RRT is efficient in high-dimensional spaces but may produce less smooth paths [46].
3. *Bezier Curve-based Planning*: This method uses Bezier curves to generate smooth paths based on control points, providing smooth trajectory transitions ideal for lane changes and path following, though it lacks flexibility in complex obstacle-filled environments [47].

3.1.2.1. Scenario 1: Straight Highway with Moderate Traffic

- Objective: Evaluate path-following accuracy and speed maintenance.
- Metrics:
  - Trajectory Smoothness (Avg. Jerk), Speed Deviation, Execution Time per cycle.
- Results:
  - LSPP: Avg. Jerk = 0.2 m/s^3, Speed Deviation = ±2 km/h, Execution Time = 12 ms
  - A* Algorithm: Avg. Jerk = 0.5 m/s^3, Speed Deviation = ±6 km/h, Execution Time = 55 ms
  - RRT: Jerk = 0.6 m/s^3, Speed Deviation = ±4 km/h, Execution Time = 40 ms
  - Bezier Curve: Jerk = 0.3 m/s^3, Speed Deviation = ±3 km/h, Execution Time = 20 *ms*
- Conclusion: LSPP demonstrates the lowest jerk and speed deviation while maintaining a short execution time, making it suitable for real-time path following.

3.1.2.2. Scenario 2: Curved Highway with Lane Changes

- Objective: Test lane-changing capability on curves.
- Metrics:
  - Lane Change Success Rate, Trajectory Smoothness (Avg. Jerk), Lateral Deviation, Execution Time.
- Results:
  - LSPP: Lane Change Success = 100%, Jerk = 0.3 m/s^3, Max Lateral Deviation = 0.1m, Execution Time = 15 *ms*
  - A* Algorithm: Lane Change Success = 85%, Jerk = 0.7 m/s^3, Max Lateral Deviation = 0.5m, Execution Time = 60 *ms*
  - RRT: Lane Change Success = 80%, Jerk = 0.8 m/s^3, Max Lateral Deviation = 0.4m, Execution Time = 50 *ms*
  - Bezier Curve: Lane Change Success = 95%, Jerk = 0.4 m/s^3, Max Lateral Deviation = 0.2m, Execution Time = 25 *ms*
- Conclusion: LSPP maintains optimal lane change success and lateral deviation with a shorter execution time, proving it efficient for real-time lane changes on curves.

3.1.2.3. Scenario 3: Emergency Stop and Obstacle Avoidance

- Objective: Assess response to sudden obstacles.
- Metrics:
  - Collision Avoidance Rate, Stopping Distance, Trajectory Smoothness (Avg. Jerk), Execution Time.
- Results:
  - LSPP: Collision Avoidance = 100%, Stopping Distance = 5m, Jerk = 0.4 m/s^3, Execution Time = 10 *ms*
  - A* Algorithm: Collision Avoidance = 70%, Stopping Distance = 2m, Jerk = 0.9 m/s^3, Execution Time = 70 *ms*
  - RRT: Collision Avoidance = 80%, Stopping Distance = 3m, Jerk = 0.7 m/s^3, Execution Time = 55 *ms*
  - Bezier Curve: Collision Avoidance = 90%, Stopping Distance = 4m, Jerk = 0.5 m/s^3, Execution Time = 30 *ms*
- Conclusion: LSPP achieves a perfect collision avoidance rate with rapid execution time, demonstrating its effectiveness in emergency response scenarios.

3.1.2.4. Scenario 4: High-Speed Curved Highway with Lane-Keeping

- Objective: Evaluate lane-keeping performance at high speeds.
- Metrics:
  - Lane Keeping Success Rate, Lateral Deviation, Trajectory Smoothness (Avg. Jerk), Execution Time.
- Results:
  - LSPP: Lane Keeping Success = 98%, Lateral Deviation = 0.15m, Jerk = 0.3 m/s^3, Execution Time = 15 *ms*
  - A* Algorithm: Lane Keeping Success = 70%, Lateral Deviation = 0.5m, Jerk = 0.7 m/s^3, Execution Time = 65 *ms*
  - RRT: Lane Keeping Success = 75%, Lateral Deviation = 0.4m, Jerk = 0.6 m/s^3, Execution Time = 52 *ms*
  - Bezier Curve: Lane Keeping Success = 90%, Lateral Deviation = 0.3m, Jerk = 0.4 m/s^3, Execution Time = 27 *ms*
- Conclusion: LSPP excels in lane-keeping at high speeds with minimal lateral deviation and low execution time, supporting its suitability for high-speed applications.

3.1.2.5. Scenario 5: Stop and Go Traffic in Congested Conditions

- Objective: Assess performance in stop-and-go traffic.
- Metrics:
  - Comfort (Avg. Jerk) during acceleration and deceleration, Traffic Flow Efficiency, Execution Time.
- Results:
  - LSPP: Jerk = 0.3 m/s^3, Traffic Flow Efficiency = 95%, Execution Time = 12 *ms*
  - A* Algorithm: Jerk = 0.6 m/s^3, Traffic Flow Efficiency = 80%, Execution Time = 60 *ms*

- RRT: Jerk = 0.5 m/s^3, Traffic Flow Efficiency = 85%, Execution Time = 50 ms
- Bezier Curve: Jerk = 0.4 m/s^3, Traffic Flow Efficiency = 90%, Execution Time = 22 ms

- Conclusion: LSPP achieves smooth accelerations and decelerations with high traffic flow efficiency, completing computations rapidly enough for real-time stop-and-go driving.

3.1.2.6. Summary of the Simulation Results

Here's in Table 5 a professional tabular representation of the simulation scenarios and results for the LSPP algorithm, showcasing various metrics and a comparison with other motion planning methods.

Table 5. Summary of the simulation scenarios results

Scenario	Metrics	LSPP	A* Algorithm	RRT	Bezier Curve
Straight Highway with Moderate Traffic	Trajectory Smoothness (Avg. Jerk) m/s^3	0.2	0.5	0.6	0.3
	Speed Deviation ± *km/h*	±2	±6	±4	±3
	Execution Time *ms*	12	55	40	20
	Lane Change Success Rate %	100	85	80	95
Curved Highway with Lane Changes	Trajectory Smoothness (Avg. Jerk) m/s^3	0.3	0.7	0.8	0.4
	Max Lateral Deviation *m*	0.1	0.5	0.4	0.2
	Execution Time *ms*	15	60	50	25
	Collision Avoidance Rate %	100	70	80	90
Emergency Stop and Obstacle Avoidance	Stopping Distance *m*	5	2	3	4
	Trajectory Smoothness (Avg. Jerk) m/s^3	0.4	0.9	0.7	0.5
	Execution Time *ms*	10	70	55	30
High-Speed Curved Highway with Lane Keeping	Lane Keeping Success Rate %	98	70	75	90
	Max Lateral Deviation *m*	0.15	0.5	0.4	0.3
	Trajectory Smoothness (Avg. Jerk) m/s^3	0.3	70	0.6	0.4
	Execution Time *ms*	15	65	52	27
Stop-and-Go Traffic in Congested Conditions	Comfort (Avg. Jerk) m/s^3	0.3	0.6	0.5	0.4
	Traffic Flow Efficiency %	95	80	80	90
	Execution Time *ms*	12	60	50	22

Notes on Metrics:

- Trajectory Smoothness (Avg. Jerk): Average jerk, measuring smoothness in acceleration and deceleration.
- Speed Deviation: Difference from target speed in km/h.
- Execution Time: Time taken per planning cycle (in milliseconds).
- Lane Change Success Rate: Percentage of successful lane changes without collision.
- Lane Keeping Success Rate: Measure the vehicle's ability to maintain its position within a designated lane.
- Max Lateral Deviation: Maximum deviation from lane center.
- Collision Avoidance Rate: Percentage of trials where collisions were avoided.
- Stopping Distance: Distance from the obstacle when the vehicle halts.
- Comfort (Avg. Jerk): Average jerk in stop-and-go traffic, indicative of passenger comfort.
- Traffic Flow Efficiency: Percentage of traffic flow maintained without unnecessary delays.

3.2. Discussion

The study's results highlight the strengths and trade-offs of the LSPP algorithm in comparison to A*, RRT, and Bezier Curve-based methods. LSPP demonstrated high performance in trajectory smoothness, lane-keeping, and collision avoidance in structured environments, though performance

varied across algorithms in different scenarios. This section discusses the factors influencing LSPP's effectiveness and the contexts where each algorithm may be most advantageous [48]-[50].

3.2.1. Execution Time

The LSPP algorithm exhibits lower execution times than A*, RRT, and Bezier Curve-based algorithms, particularly in structured environments like highways. This efficiency is due to several factors: LSPP's deterministic, quintic spline-based approach generates smooth, continuous paths without the need for extensive search or iterative exploration, as required by A* and RRT. By directly optimizing the trajectory based on boundary conditions (position, velocity, and acceleration), LSPP produces kinematically feasible paths in fewer steps, avoiding the complex pathfinding and post-processing typically needed for other algorithms. Designed to handle kinematic constraints up to jerk, LSPP efficiently supports real-time applications, making it particularly suited for high-speed, continuous path planning where smoothness and execution speed are critical [51]-[53].

3.2.2. Collision Avoidance

A higher Collision Avoidance Rate for the LSPP algorithm compared to other methods in this paper is logical, particularly in structured environments. LSPP's quintic spline-based approach generates smooth, continuous paths that adhere to kinematic constraints, enabling stable, predictable trajectories that minimize abrupt maneuvers, which can increase collision risk. This inherent smoothness allows the vehicle to maintain consistent control and avoid obstacles effectively without extensive recalculations. Unlike A* and RRT, which may require frequent replanning or post-processing to adapt to new obstacles, LSPP supports localized adjustments, enabling dynamic obstacle avoidance while preserving path continuity. Consequently, LSPP is particularly advantageous in scenarios where smoothness, stability, and real-time responsiveness are essential.

3.2.3. Speed Deviation

Lower Speed Deviation from the target speed with the LSPP algorithm, compared to A*, RRT, and Bezier Curve-based algorithms, is justified due to LSPP's use of smooth, kinematically feasible quintic splines. This smoothness minimizes abrupt speed changes, enabling the vehicle to follow a steady trajectory without frequent adjustments. LSPP also allows localized path adjustments for minor obstacles without requiring full replanning, further supporting speed consistency. By contrast, A* and RRT often produce non-smooth paths that require speed adjustments, while Bezier-based paths may introduce inconsistencies in dynamic settings. Overall, LSPP's continuous, kinematically aware paths help maintain stable speeds, particularly in high-speed, structured environments.

4. Conclusion

This paper presented the development and evaluation of a novel Local Spline-based Path Planner (LSPP) for autonomous vehicles, which integrates intelligent waypoint generation, quintic spline fitting, and Model Predictive Control (MPC) for smooth and accurate path tracking. The proposed framework demonstrates promising results in structured, highway-like environments, showing high performance in terms of safety, comfort, and real-time responsiveness. By leveraging spline curves for trajectory generation and combining them with rule-based waypoint selection, LSPP ensures continuity, smoothness, and feasibility of vehicle paths while respecting dynamic constraints.

Despite these strengths, several limitations and challenges remain. A key concern is the generalizability of the LSPP algorithm beyond structured environments. While performance in highways is robust, the algorithm's effectiveness in unstructured, congested, or dynamic urban settings—with unpredictable moving obstacles such as pedestrians or cyclists—remains an open question. The current design assumes a relatively stable environment and may not adapt well to erratic behaviors or sudden changes in traffic flow.

Additionally, the reliance on accurate sensor data for waypoint generation and state estimation introduces vulnerability to sensor noise and failure, especially in adverse weather conditions such as heavy rain, fog, or snow. These conditions can significantly degrade sensor performance, leading to

inaccurate trajectory planning. Another limitation lies in the use of quintic splines, which, while smooth and efficient, depend on well-defined boundary conditions and may struggle to adapt flexibly in highly constrained or cluttered environments. Moreover, the integration with MPC adds computational overhead, which, although manageable in simulations, could pose real-time performance issues on embedded systems when combined with sensor fusion and other high-frequency planning modules. Scalability also remains a concern. As the environmental complexity increases, so does the need for more frequent replanning and tighter integration with perception and prediction modules. LSPP's current structure may need optimization or parallelization to sustain real-time responsiveness in such scenarios. Future work will focus on enhancing the adaptability and robustness of LSPP by:

- Extending its capabilities to unstructured and urban environments.
- Incorporating sensor fusion with probabilistic models to handle uncertainty and noise.
- Investigating edge-case performance under scenarios such as high-speed merging, sudden obstacle appearances, and sharp turns.
- Exploring lightweight implementations and hardware acceleration to ensure real-time deployment feasibility.

In conclusion, while LSPP represents a significant step toward reliable and smooth local path planning for autonomous vehicles, it is not without limitations. A critical assessment of its scalability, robustness, and adaptability to real-world complexities is essential for its evolution into a deployable solution. The insights gained through this study provide a strong foundation for addressing these challenges in future research.

Supplementary Materials: Availability of data and material: available upon request.

Author Contribution: WAF: Writing – original draft, Methodology, Formal analysis, Data curation, Conceptualization. MOF: Writing review & editing.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

List of abbreviations

APF : Artificial Potential Field
FSM : Finite State Machine
JMT : Jerk Minimizing Trajectory
ODM : Object Detection Module
LSPP : Localized Spline-based Path-Planning
MPC : Model Predictive Control
MPM : Mission Planner Module
RBLPP : Rule-Based Localized Path Planner
RRT : Rapidly-exploring Random Tree
TTCs : Triggering Transitional Conditions

References

[1] W. Farag, "Safe-driving cloning by deep learning for autonomous cars," *International Journal of Advanced Mechatronic Systems*, vol. 7, no. 6, pp. 390-397, 2019, https://doi.org/10.1504/IJAMECHS.2017.099318.

[2] W. Farag and Z. Saleh, "Road Lane-Lines Detection in Real-Time for Advanced Driving Assistance Systems," *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1-8, 2018, https://doi.org/10.1109/3ICT.2018.8855797.

[3] W. Farag, "Traffic signs classification by deep learning for advanced driving assistance systems," *Intelligent Decision Technologies*, vol. 13, no. 3, pp. 215-231, 2019, https://doi.org/10.3233/IDT-180064.

[4] W. Farag and Z. Saleh, "An advanced vehicle detection and tracking scheme for self-driving cars," *2nd Smart Cities Symposium (SCS 2019)*, pp. 1-6, 2019, https://doi.org/10.1049/cp.2019.0222.

[5] W. Farag, "Recognition of traffic signs by convolutional neural nets for self-driving vehicles," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 22, no. 3, pp. 205-214, 2018, https://doi.org/10.3233/KES-180385.

[6] W. Farag and Z. Saleh, "Behavior Cloning for Autonomous Driving using Convolutional Neural Networks," *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1-7, 2018, https://doi.org/10.1109/3ICT.2018.8855753.

[7] W. Farag and Z. Saleh, "Tuning of PID Track Followers for Autonomous Driving," *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1-7, 2018, https://doi.org/10.1109/3ICT.2018.8855773.

[8] K. B. Patel, H. C. Lin, A. D. Berger, W. Farag, A. A. Khan, "U.S. Patent No. 6,196,327," *U.S. Patent and Trademark Office*, 2001, https://patents.google.com/patent/US6196327B1/en.

[9] K. Shao, J. Zheng, and K. Huang, "Robust active steering control for vehicle rollover prevention," *International Journal of Modelling, Identification and Control*, vol. 32, no. 1, pp. 70-84, 2019, https://doi.org/10.1504/IJMIC.2019.101956.

[10] P. Bautista-Camino, A. I. Barranco-Gutiérrez, I. Cervantes, M. Rodríguez-Licea, J. Prado-Olivarez, F. J. Pérez-Pinal, "Local Path Planning for Autonomous Vehicles Based on the Natural Behavior of the Biological Action-Perception Motion," *Energies*, vol. 15, no. 5, p. 1769, 2022, https://doi.org/10.3390/en15051769.

[11] 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, 2020, https://doi.org/10.1109/TITS.2019.2913998.

[12] W. A. Farag, "Kalman-filter-based sensor fusion applied to road-objects detection and tracking for autonomous vehicles," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 235, no. 7, pp. 1125-1138, 2021, https://doi.org/10.1177/0959651820975523.

[13] W. A. Farag, "A lightweight vehicle detection and tracking technique for advanced driving assistance systems," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 39, no. 3, pp. 1-13, 2020, https://doi.org/10.3233/JIFS-190634.

[14] W. Farag, "Multiple road-objects detection and tracking for autonomous driving," *Journal of Engineering Research*, vol. 10, no. 1A, pp. 237-262, 2021, https://doi.org/10.36909/jer.10993.

[15] W. A. Farag, "Real-time detection of road lane-lines for autonomous driving," *Recent Advances in Computer Science and Communications*, vol. 13, no. 2, pp. 265-274, 2020, http://dx.doi.org/10.2174/2213275912666190126095547.

[16] J. S. Tjiharjadi, S. Razali, and H. A. Sulaiman, "A systematic literature review of multi-agent pathfinding for maze research," *Journal of Advances in Information Technology*, vol. 13, no. 4, pp. 358-367, 2022, https://doi.org/10.12720/jait.13.4.358-367.

[17] W. Farag, "Real-time autonomous vehicle localization based on particle and unscented Kalman filters," *Journal of Control, Automation and Electrical Systems*, vol. 32, no. 2, pp. 309-325, 2021, https://doi.org/10.1007/s40313-020-00666-w.

[18] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," *Robotics and Autonomous Systems*, vol. 174, p. 104630, 2024, https://doi.org/10.1016/j.robot.2024.104630.

[19] W. Farag, "Complex track maneuvering using real-time MPC control for autonomous driving," *International Journal of Computing and Digital Systems*, vol. 9, no. 5, pp. 1-15, 2020, https://doi.org/10.12785/ijcds/090511.

[20] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, A. García-Cerezo, "Path Planning for Autonomous Mobile Robots: A Review," *Sensors*, vol. 21, no. 23, p. 7898, 2021, https://doi.org/10.3390/s21237898.

[21] W. Xu, Q. Wang and J. M. Dolan, "Autonomous Vehicle Motion Planning via Recurrent Spline Optimization," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7730-7736, 2021, https://doi.org/10.1109/ICRA48506.2021.9560867.

[22] W. Farag, "Complex-track following in real-time using model-based predictive control," *International Journal of Intelligent Transportation Systems Research*, vol. 19, no. 1, pp. 112-127, 2021, https://doi.org/10.1007/s13177-020-00226-1.

[23] W. Farag, "Real-time NMPC path tracker for autonomous vehicles," *Asian Journal of Control*, vol. 23, no. 4, pp. 1952-1965, 2021, https://doi.org/10.1002/asjc.2335.

[24] J. Wen, X. Zhang, Q. Bi, H. Liu, J. Yuan and Y. Fang, "G²VD Planner: Efficient Motion Planning With Grid-Based Generalized Voronoi Diagrams," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 3743-3755, 2025, https://doi.org/10.1109/TASE.2024.3398996.

[25] C. S. Tan, R. Mohd-Mokhtar and M. R. Arshad, "A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms," *IEEE Access*, vol. 9, pp. 119310-119342, 2021, https://doi.org/10.1109/ACCESS.2021.3108177

[26] P. Qin, F. Liu, Z. Guo, Z. Li, Y. Shang, "Hierarchical collision-free trajectory planning for autonomous vehicles based on improved artificial potential field method," *Transactions of the Institute of Measurement and Control*, vol. 46, no. 4, pp. 799-812, 2024, https://doi.org/10.1177/01423312231186684.

[27] M. R. Siddiqi, S. Milani, R. N. Jazar and H. Marzbani, "Ergonomic Path Planning for Autonomous Vehicles-An Investigation on the Effect of Transition Curves on Motion Sickness," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7258-7269, 2022, https://doi.org/10.1109/TITS.2021.3067858.

[28] X. Qian, I. Navarro, A. de La Fortelle and F. Moutarde, "Motion planning for urban autonomous driving using Bézier curves and MPC," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 826-833, 2016, https://doi.org/10.1109/ITSC.2016.7795651.

[29] W. A. Farag, V. H. Quintana and G. Lambert-Torres, "Genetic algorithms and back-propagation: a comparative study," *Conference Proceedings. IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No.98TH8341)*, vol. 1, pp. 93-96, 1998, https://doi.org/10.1109/CCECE.1998.682559.

[30] A. Rucco, P. B. Sujit, A. P. Aguiar, J. B. de Sousa and F. L. Pereira, "Optimal Rendezvous Trajectory for Unmanned Aerial-Ground Vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 2, pp. 834-847, 2018, https://doi.org/10.1109/TAES.2017.2767958.

[31] P. Lin, E. Javanmardi and M. Tsukada, "Clothoid Curve-Based Emergency-Stopping Path Planning With Adaptive Potential Field for Autonomous Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 9747-9762, 2024, https://doi.org/10.1109/TVT.2024.3380745.

[32] J. Kong, M. Pfeiffer, G. Schildbach and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094-1099, 2015, https://doi.org/10.1109/IVS.2015.7225830.

[33] A. N. Sharkawy, "Minimum jerk trajectory generation for straight and curved movements: Mathematical analysis," *arXiv*, 2021, https://doi.org/10.48550/arXiv.2102.07459.

[34] J. Dalle, D. Hastuti, and M. R. A. Prasetya, "The use of an application running on the ant colony algorithm in determining the nearest path between two points," *Journal of Advances in Information Technology*, vol. 12, no. 3, pp. 206-213, 2021, https://doi.org/10.12720/jait.12.3.206-213.

[35] W. Farag and M. Nadeem, "Enhanced real-time road-vehicles' detection and tracking for driving assistance," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 28, no. 2, pp. 335-357, 2024, https://doi.org/10.3233/KES-230062.

[36] N. Petrellis *et al*., "Software acceleration of the deformable shape tracking application: How to eliminate the Eigen library overhead," *ESSE '21: Proceedings of the 2021 European Symposium on Software Engineering*, pp. 51-57, 2021, https://doi.org/10.1145/3501774.3501782.

[37] X. Wang, X. Qi, P. Wang, J. Yang, "Decision making framework for autonomous vehicles driving behavior in complex scenarios via hierarchical state machine," *Autonomous Intelligent Systems*, vol. 1, no. 10, 2021, https://doi.org/10.1007/s43684-021-00015-x.

[38] M. Werling, J. Ziegler, S. Kammel and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," *2010 IEEE International Conference on Robotics and Automation*, pp. 987-993, 2010, https://doi.org/10.1109/ROBOT.2010.5509799.

[39] J. Ziegler *et al*., "Making Bertha Drive—An Autonomous Journey on a Historic Route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8-20, 2014, https://doi.org/10.1109/MITS.2014.2306552.

[40] D. Kim, G. Kim, H. Kim and K. Huh, "A Hierarchical Motion Planning Framework for Autonomous Driving in Structured Highway Environments," *IEEE Access*, vol. 10, pp. 20102-20117, 2022, https://doi.org/10.1109/ACCESS.2022.3152187.

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

[42] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135-1145, 2016, https://doi.org/10.1109/TITS.2015.2498841.

[43] H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu, and T. Liu, "The EBS-A* algorithm: An improved A* algorithm for path planning," *PLOS ONE*, vol. 17, no. 2, p. e0263841, 2022, https://doi.org/10.1371/journal.pone.0263841.

[44] T. Chang and G. Tian, "Hybrid A-Star path planning method based on hierarchical clustering and trichotomy," *Applied Sciences*, vol. 14, no. 13, p. 5582, 2024, https://doi.org/10.3390/app14135582.

[45] H. Wang, X. Zhou, J. Li, Z. Yang, and L. Cao, "Improved RRT* algorithm for disinfecting robot path planning," *Sensors*, vol. 24, no. 5, p. 1520, 2024, https://doi.org/10.3390/s24051520.

[46] F. Yang, X. Fang, F. Gao, X. Zhou, H. Li, H. Jin, and Y. Song, "Obstacle avoidance path planning for UAV based on improved RRT algorithm," *Discrete Dynamics in Nature and Society*, vol. 2022, no. 1, p. 4544499, 2022, https://doi.org/10.1155/2022/4544499.

[47] L. Zheng, P. Zeng, W. Yang, Y. Li, and Z. Zhan, "Bézier curve-based trajectory planning for autonomous vehicles with collision avoidance," *IET Intelligent Transport Systems*, vol. 14, no. 13, pp. 1882-1891, 2020, https://doi.org/10.1049/iet-its.2020.0355.

[48] W. Farag, "Synthesis of intelligent hybrid systems for modeling and control," *University of Waterloo*, 1998, https://dspacemainprd01.lib.uwaterloo.ca/server/api/core/bitstreams/c464b29f-93c9-4241-95d9-f263522e1fba/content.

[49] W. A. Farag, V. H. Quintana and G. Lambert-Torres, "Neuro-fuzzy modeling of complex systems using genetic algorithms," *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 1, pp. 444-449, 1997, https://doi.org/10.1109/ICNN.1997.611709.

[50] W. Farag, "Road-objects tracking for autonomous driving using lidar and radar fusion," *Journal of Electrical Engineering*, vol. 71, no. 3, pp. 138-149, 2020, https://doi.org/10.2478/jee-2020-0021.

[51] W. A. Farag, M. Fayed, "Advancing vehicle detection for autonomous driving: integrating computer vision and machine learning techniques for real-world deployment," *Journal of Control and* Decision, 2025, https://doi.org/10.1080/23307706.2025.2469893.

[52] W. A. Farag, M. Helal, "Real-time localization with probabilistic maps and unscented kalman filtering: a dynamic sensor fusion approach," *Journal of Control and Decision*, 2024, https://doi.org/10.1080/23307706.2024.2417218.

[53] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58443-58469, 2020, https://doi.org/10.1109/ACCESS.2020.2983149.