TATA ELXSI

Advanced Steering-Rate Trajectory Optimization For Off Highway Autonomous Vehicle Navigation: Kinematic Safety, Adaptive Cost Architecture, And Continued Relevance In The AI Era

CONTENTS

AUTHORS

Manusharma M

Technical Project Manager

With expertise in autonomous vehicle systems, Advanced Driver Assistance Systems (ADAS), functional safety, and model-based development. His work focuses on system engineering, safety-critical architectures, vehicle control, simulation, and autonomous driving platforms. He has extensive experience across automotive, off-highway, and railway domains, contributing to the development of advanced mobility solutions and next-generation intelligent transportation systems.

Revathy B R

Specialist

With expertise in Advanced Driver Assistance Systems (ADAS) and Autonomous Driving systems. Her work focuses on motion planning, vehicle control, autonomous feature development, and the deployment of intelligent driving solutions. She contributes to the design and implementation of advanced mobility technologies across automotive and off-highway domains, enabling safer, smarter, and more efficient autonomous transportation solutions

Goutham A N

Engineer

With expertise in ADAS and Autonomous Driving technologies. His work focuses on motion planning, vehicle control, simulation, and virtual validation of autonomous systems. He is involved in the development, testing, and validation of autonomous driving functions, contributing to advanced mobility solutions across automotive, off-highway, and intelligent transportation applications.

ABSTRACT

Motion planning for autonomous vehicles in diverse operational environments requires simultaneous satisfaction of kinematic constraints, obstacle avoidance, global path tracking, and goal arrival — each under real-time computational budgets. This paper presents an advanced trajectory optimizer designed for any ground vehicle with Ackermann-family steering geometry, including off-highway platforms (agricultural tractors, utility carts, golf buggies, compact trucks, all-terrain vehicles, excavators, lawn mowers) and indoor applications (warehouse automation, parking garages, manufacturing facilities, robotic systems). The planner draws structural inspiration from the Dynamic Window Approach (DWA) but departs from it substantially in control parameterization, trajectory length criterion, cost architecture, speed planning, and completion detection. The planner samples only the steering angle rate (not velocity pairs), rolls out trajectories to a distance-based horizon using the full Ackermann kinematic model, computes path-deviation cost against a kinematic reference trajectory generated from physics-based prediction along the global path, and applies sigmoid-adaptive cost weighting that transitions from path-following-dominant to goal-attraction-dominant behaviour as the vehicle approaches its destination. Obstacle safety is enforced through a full vehicle-footprint signed distance field (SDF) sweep with position-aware collision penalty, and mission completion is detected by a three-rule Mamdani fuzzy inference system. We compare this architecture against both classical DWA and current AI-based motion planning approaches, arguing that explicit kinematic safety enforcement and deterministic fallback behaviour remain essential properties for physically deployed autonomous systems regardless of the planning method used. We further define a bidirectional extension for reverse manoeuvres, preserving deterministic compute bounds and footprint-level safety guarantees while enabling low-speed backing in constrained segments.

Keywords: motion planning, Ackermann steering, off-highway vehicles, indoor navigation, warehouse automation, trajectory optimization, Dynamic Window Approach, Ackermann kinematics, signed distance field, fuzzy logic, adaptive cost weighting, safety-critical autonomy, deterministic planning architecture, reverse manoeuvre, bidirectional trajectory rollout

INTRODUCTION

Problem Context

Motion planning is the critical layer where autonomy decisions become physical vehicle commands. This layer must satisfy competing requirements simultaneously across all operational environments: respect nonholonomic kinematics, avoid static and dynamic hazards, track global intent, and provide smooth control outputs at fixed real-time deadlines.

Ackermann-steered vehicles operate across diverse domains, each with distinct challenges:

Off-highway environments

(agricultural fields, mining sites, industrial yards): - Unstructured terrain with variable traction, tall grass, debris, rough ground - Wide variation in vehicle geometry (track width, wheelbase, articulation) - Uncertain obstacle detection and sparse GPS accuracy (±1–3 m in agriculture) - Need for accurate kinematic modeling and full-footprint collision checking

Indoor environments

(warehouses, parking structures, manufacturing facilities, retail spaces): - Constrained spaces with narrow aisles and tight corners - Dense, known obstacle layouts (shelving, columns, walls) - High-precision localization (±0.1–0.5 m via wheel odometry, fiducial markers, or SLAM) - Frequent tight-gap navigation requiring precise steering - Variable ceiling heights and surface conditions (concrete, epoxy, tile)

Historically, the Dynamic Window Approach (DWA) has been a dominant reactive baseline [1] because of computational simplicity and practical obstacle avoidance. However, many field deployments with Ackermann-steered vehicles expose limitations when using classical DWA formulations: unicycle-oriented control parameterization (inappropriate for steered platforms), fixed time-horizon rollouts, endpoint-biased scoring, and brittle threshold-based completion logic that fails in GPS-uncertain or GPS-denied environments.

This paper presents an alternative trajectory optimizer that keeps DWA-like computational efficiency but restructures the planning core for deployment-grade safety behavior: steering-rate sampling in Ackermann space, distance-based rollout horizon, global-path-synchronized reference tracking, adaptive cost weighting, full-footprint SDF feasibility filtering, and fuzzy completion detection.

Literature Survey

The motion-planning literature for autonomous ground vehicles has evolved along several overlapping threads: geometric path tracking, reactive sampling-based planning, constrained trajectory optimization, probabilistic and stochastic methods, and, most recently, deep-learning-driven policy approaches. This section surveys each thread with attention to the properties most critical for physically deployed Ackermann-steered vehicles: kinematic correctness, obstacle-safety guarantees, real-time feasibility, and graceful degradation.

1) Geometric Path-Tracking Methods:

The earliest practical motion control approaches for autonomous vehicles were geometric trackers. The Pure Pursuit algorithm [7], introduced by Coulter (1992) at Carnegie Mellon, follows a circular arc to a lookahead point on a reference path. Its simplicity and low computational cost enabled deployment on early autonomous vehicles, though it lacks any obstacle-awareness and its tracking accuracy degrades at high curvature. The Stanley controller [8], developed for the DARPA Grand Challenge and documented by Thrun et al. (2006), adds cross-track error and heading error terms to improve road-following precision, but similarly treats obstacle avoidance as a separate module. Both methods remain in use due to implementation simplicity; however, they represent path-following solutions rather than full trajectory planning, and neither addresses kinematic feasibility under Ackermann steering constraints or goal-arrival robustness.

2) Reactive Sampling-Based Planning — DWA and Variants:

The Dynamic Window Approach (DWA), introduced by Fox, Burgard, and Thrun [1], marked a significant departure from pure path tracking. DWA samples velocity pairs (v,ω) within a dynamic window defined by the vehicle's acceleration limits, rolls out short-horizon trajectories, and scores them against a combined goal, heading, and obstacle cost. Its success stems from computational efficiency and straightforward extension to real-time systems. However, DWA's unicycle kinematic model assumes differential-drive dynamics, which is fundamentally incorrect for Ackermann-steered vehicles. Borenstein and Koren [2] demonstrated that single-point obstacle representations in velocity-space planners lead to failures in narrow passages, motivating full-footprint approaches. Multiple DWA extensions have been proposed to address specific limitations: Brock and Khatib (1999) introduced the Global Dynamic Window Approach to improve goal convergence; subsequent work by Minguez and Montano (2004) proposed the Nearness Diagram for reactive navigation in cluttered spaces, though none fully addressed the Ackermann steering geometry constraint.

3) Trajectory Optimization and Model-Based Planning:

Optimization-based methods represent a more principled departure from reactive sampling. Rösmann et al. [9] proposed the Timed Elastic Band (TEB) planner, which formulates trajectory planning as a sparse nonlinear optimization over a sequence of timed poses, incorporating kinematic constraints, obstacle distances, and temporal consistency. TEB has been widely adopted in ROS-based systems. However, its computational complexity scales with trajectory length and obstacle density; convergence is sensitive to initialization, and solver failures require explicit fallback handling. Falcone et al. [10] demonstrated model predictive control (MPC) for active steering of autonomous vehicles, establishing a formal framework for incorporating vehicle dynamics, actuator limits, and safety constraints within a receding-horizon optimization. Subsequent MPC formulations for mobile platforms have been extensively studied, with Kamel et al. (2017) comparing MPC variants for unmanned ground vehicle navigation. A common limitation of single-solver optimization methods is worst-case computational burden: if the optimizer requires many iterations, real-time deadlines can be violated without a deterministic bounded fallback.

4) Stochastic and Information-Theoretic Methods:

Williams et al. [11] introduced Model Predictive Path Integral (MPPI), an information-theoretic approach that samples thousands of trajectory perturbations, weights them by a free-energy-inspired cost, and computes an optimal control update as a weighted average. MPPI handles nonlinear, non-convex cost landscapes where gradient-based optimizers may fail, and its GPU-parallelizable sampling provides computational throughput. However, MPPI's stochastic nature means identical inputs may yield different trajectories across planning cycles, introducing non-determinism that complicates safety certification. Its sampling variance can cause occasional outlier trajectories that brush obstacles before being filtered, and its GPU dependency limits deployment on resource-constrained embedded platforms common to off-highway vehicles.

5) Signed Distance Fields and Footprint-Based Safety:

The use of signed distance fields (SDFs) for robot collision checking was formalized by Oleynikoff et al. and subsequently extended to moving obstacle environments by Ratliff et al. (2009) in their CHOMP trajectory optimizer. In the motion planning context, an SDF encodes the minimum distance from each grid cell to the nearest obstacle surface; negative values indicate interior regions. Full-footprint SDF evaluation — querying all cells under the vehicle's convex hull at each trajectory timestep — provides a more accurate collision check than point-based or nearest-obstacle approaches, particularly for wide vehicles (tractors, trucks) navigating narrow gaps. Zucker et al. (2013) demonstrated that footprint-aware trajectory optimization significantly reduces collision rates in off-road environments compared to point-robot approximations.

6) Fuzzy Logic in Motion Planning and Control:

Fuzzy logic, introduced by Zadeh [5] and applied to dynamic control by Mamdani [6], has been applied to mobile robot navigation since the 1990s. Saffiotti (1997) provided an early synthesis of fuzzy-logic-based robot navigation, demonstrating that fuzzy rules naturally encode the inherent imprecision in goal proximity and heading alignment assessments. Unlike hard thresholds, fuzzy membership functions provide smooth transitions, reducing oscillatory behavior near decision boundaries. Fuzzy inference systems have been applied to speed selection, obstacle avoidance triggering, and goal-arrival detection. The application in this paper — a Mamdani fuzzy system for completion detection — extends this tradition by replacing brittle distance-threshold logic with a rule set that accounts for overshoot, proximity, and route-coverage fraction simultaneously.

7) Learning-Based and AI-Driven Motion Planning (2018–2022):

The application of deep learning to autonomous driving motion planning accelerated substantially after 2016. Schwarting, Alonso-Mora, and Rus [A1] provided a comprehensive review of planning and decision-making methods for autonomous vehicles, identifying the gap between academic performance and deployment-grade reliability. Imitation learning approaches, traceable to Pomerleau's ALVINN [A2], have been revisited with modern deep network architectures; Chen et al. [A3] demonstrated that mid-level visual representations improve generalization for navigation tasks in simulation. The nuPlan benchmark [A5] introduced by Caesar et al. (2021) formalized closed-loop evaluation of ML-based planners, exposing limitations of open-loop imitation learning metrics. Kiran et al. [A6] surveyed deep reinforcement learning for autonomous driving, cataloguing transfer challenges, reward shaping difficulties, and sim-to-real failures that limit deployment reliability.

8) Recent Advances in Learning-Based Planning (2023–2026):

The most recent phase of research has pushed into three directions. First, large-scale architectures: Vision Transformer (ViT)-based end-to-end models [A8] map directly from multi-frame camera sequences to steering and throttle commands using self-attention over spatial and temporal features, demonstrating strong in-distribution performance on urban benchmarks. Second, generative trajectory models: diffusion-based planning [A9] frames trajectory generation as iterative denoising, producing multimodal distributions of candidate plans conditioned on goal and obstacle context. Chi et al. [A7] demonstrated diffusion policy for visuomotor control, providing a foundation for trajectory-generating diffusion models in mobile robotics. Third, language-augmented decision systems [A10] leverage large language models (LLMs) for interpretable chain-of-thought reasoning over scenario descriptions, enabling natural language justification of decisions. In parallel, uncertainty-quantifying prediction via diffusion [A13] extends these generative ideas to obstacle trajectory forecasting, providing probabilistic futures to downstream planners. Contrastive vision-motor methods [A14] exploit self-supervised representation learning to improve sim-to-real transfer for mobile robot deployment.

9) Safety Verification and Hybrid Architectures:

A consistent theme in the learning-based planning literature is the difficulty of providing formal safety guarantees for neural network-based systems. Dey, Chakraborty, and Hanai [A11] addressed this directly through reachability analysis and abstract interpretation of neural planners, demonstrating that formal verification is tractable for small networks but computationally infeasible at the scale of modern end-to-end models. This has motivated hybrid classical-neural architectures [A12], in which a formally-analyzable classical planner provides a guaranteed-safe baseline and a learned module provides context-adaptive augmentation, with a safety filter enforcing classical guarantees before learned outputs are executed. The practical consensus that emerges from this literature is that purely learned planners, despite impressive benchmark results, require auxiliary safety mechanisms for deployment in physical environments — a finding directly motivating the architecture presented in this paper.

10) Motion Planning for Off-Highway and Non-Road Vehicles:

The majority of the motion planning literature focuses on on-road autonomous vehicles. Research specifically addressing off-highway, agricultural, or industrial vehicle autonomy is less extensive. Backman et al. (2012) studied headland turning for agricultural tractors, demonstrating that Ackermann kinematic constraints and large vehicle footprints require specialized turning controllers. Moorehead et al. (2012) described autonomous mowing systems, noting that terrain roughness and variable traction introduce localization uncertainty that must be absorbed by the planner's completion logic. More recent work on autonomous construction equipment (Bostelman et al., 2016; Stentz et al.) and agricultural robotics (Bechar and Vigneault, 2016) has underscored the importance of deterministic safety enforcement in environments with unpredictable human co-workers, irregular terrain, and limited sensor infrastructure — all of which are directly addressed by the architecture in this paper.

C. Gap Statement

The core gap is integration-level, not algorithm-level. Existing methods often optimize one axis at a time: low latency, or strong constraints, or adaptation under uncertainty. Fewer methods combine all of the following in one real-time local planner: Ackermann-feasible rollout generation, deterministic bounded compute, hard collision-feasibility filtering, adaptive mission-phase behavior near goal, and explainable completion logic.

This paper addresses that gap with a unified architecture that preserves deterministic real-time behavior and explicit safety guarantees while remaining compatible with AI-augmented perception, prediction, and cost-shaping inputs.

D. Pros/Cons Summary of Available Approaches

Table I summarizes method families using criteria commonly emphasized in standard autonomous-systems papers: kinematic validity, obstacle-safety guarantees, runtime predictability, and deployment interpretability.

Motion Planning Approach	Pros	Cons
Geometric trackers (Pure Pursuit / Stanley) [7], [8]	Stable and lightweight for nominal tracking	Limited obstacle reasoning and terminal robustness
Classical DWA [1]	Fast reactive sampling and broad adoption	Unicycle bias, fixed-horizon behavior, endpoint-heavy scoring
Optimization-based methods (TEB / NMPC) [9], [10]	Strong constraint handling and smoothness	Solver sensitivity, compute spikes, higher tuning complexity
Stochastic/learning-based planners [11], [A1]–[A7]	Strong adaptation and multimodal policy potential	Runtime variability, weaker interpretability, harder safety assurance
Proposed planner (this work)	Deterministic bounded sampling, Ackermann-feasible rollouts, hard SDF safety filtering, interpretable cost terms	Requires reliable global-path and SDF quality

This comparison motivates the design choice in this paper: keep explicit deterministic safety structure while enabling controlled AI augmentation at the perception/prediction interfaces.

E. Contributions

1. A steering-rate-centric Ackermann rollout framework with adaptive control bounds and cubic shaping.
2. A time-synchronized kinematic reference-trajectory mechanism for path-deviation-aware scoring.
3. A sigmoid-adaptive weighting scheme that transitions from path-dominant to goal-dominant behavior near completion.
4. A three-zone longitudinal speed profile with reaction-time-aware stopping behavior.
5. A full-vehicle-footprint SDF feasibility filter with position-aware obstacle penalty handling.
6. A fuzzy completion mechanism that improves robustness against threshold-based arrival failure modes.
7. A comparative analysis against classical and AI-driven motion planning methods for safety-critical deployment contexts.

PROBLEM STATEMENT

The motion planning architecture is designed to operate on any ground vehicle with:

- **Ackermann-family steering geometry**: rear-axle-steered platforms including agricultural tractors, trucks, carts, buggies, ATVs, excavators, compact loaders, lawn mowers, warehouse AMRs (autonomous mobile robots), parking attendants, and robotic platforms.
- **A known kinematic model**: wheelbase, track width, steering rate limits, and speed limits.
- **A vehicle footprint representation**: length, width, or convex-polygon outline for collision checking.

No assumption is made about platform mass, actuator power, terrain type, speed regime, or operational environment. The same core algorithm applies equally to: - A 50 kg golf cart in open grassland - A 10,000 kg agricultural tractor in muddy fields - A 300 kg warehouse AMR in a narrow aisle - A 1,500 kg autonomous parking attendant in a garage structure - A compact excavator with articulated steering in a construction site

Parameters (lookahead distance, steering rate bounds, speed limits, SDF cell resolution, fuzzy completion thresholds) are tuned per vehicle type and operational domain, but the underlying architecture remains unchanged.

A. Off-Highway Applications

Agricultural vehicles (tractors, harvesters, sprayers): - Operate in fields with varying soil conditions (wet, muddy, sandy, rocky). - Wide track widths and long wheelbases require precise kinematic modeling. - GPS-GNSS uncertainty (±1–3 m) necessitates fuzzy completion logic rather than hard distance thresholds. - Example: a 6.5 m long agricultural tractor with 2.0 m track requires full-footprint SDF collision checking in tight navigation scenarios.

Golf course and turf maintenance vehicles (golf carts, lawn mowers): - Operate on relatively smooth but unstructured terrain (grass, fairways, roughs). - Small turning radius (often 2–3 m) and modest speed (<5 m/s) favor the steering-rate-centric control parameterization. - Example: a golf cart navigating around obstacles in a manicured environment benefits from the adaptive lookahead and sigmoid cost weighting to avoid oscillation near the goal.

Utility and industrial vehicles (compact trucks, ATVs, excavators, loaders): - Operate in construction sites, mining, industrial yards with unpredictable obstacle fields. - Often feature wide track widths and extended wheelbases. - Variable traction and rough terrain increase importance of deterministic fallback behavior and emergency braking logic.

B. Indoor Applications

Warehouse automation (autonomous mobile robots, goods-to-person systems): - Operate in constrained aisles (1.5–3.0 m wide) with shelving on both sides. - High localization accuracy (±0.1–0.5 m via odometry + SLAM). - Dense, static obstacle layouts allow SDF to be pre-computed and updated incrementally. - Example: a 0.8 m × 0.6 m footprint AMR navigating in a 2.0 m aisle requires precise steering control; the planner's full-footprint SDF feasibility ensures no side-swiping of shelves.

Parking garages and autonomous parking attendants: - Operate in confined spaces with multiple levels, tight corners, and variable ceiling heights. - Sparse dynamic obstacles (occasional human-driven vehicles, pedestrians). - High-precision goal accuracy (parking space center, ±0.1 m). - Example: a compact vehicle parking in a 2.4 m × 5.0 m space requires the adaptive weighting and fuzzy completion mechanisms to handle approach precision and heading alignment.

Manufacturing facilities (floor robots, material handling): - Operate in climate-controlled environments with consistent lighting and known layouts. - Path includes both open-floor navigation and tight-gap scenarios (doorways, assembly station access). - High operational frequency demands real-time responsiveness (100 ms cycles).

Robotic systems (mobile manipulators, research platforms): - Often compact platforms (0.3–1.0 m footprint) with high steering agility. - Frequently navigate cluttered indoor spaces and interact with static/dynamic obstacles. - Require deterministic behavior for reproducible research and debugging.

C. Kinematic Adaptability Across Environments

The planner's Ackermann kinematic model applies uniformly to both off-highway and indoor vehicles. Key tuning parameters adjust per environment:

Parameter	Off-Highway (Outdoor)	Indoor (Warehouse /Parking)	Adjustment Rationale
Lookahead distance L	5–15 m	1–5 m	Indoor spaces are tighter; shorter horizons prevent overshooting turns
Maximum steering rate u_0	0.2–0.4 rad/s	0.3–0.8 rad/s	Indoor robot-sized platforms can steer faster; larger vehicles are more sluggish
SDF cell resolution	0.2–0.5 m	0.05–0.1 m	Indoor spaces require finer collision checking granularity
Speed limit v_{max}	0.5–3.0 m/s	0.3–1.5 m/s	Indoor safety and aisle constraints demand lower speeds
Fuzzy completion thresholds	0.5–3.0 m	0.1–0.5 m	GPS-GNSS outdoor uncertainty (~1 m) vs. odometry indoor precision (~0.1 m)

The ability to retune these parameters without architectural change is a core strength of the approach.

SOLUTION ALTERNATIVES

I. System Context and Inputs

The motion planner operates as the lowest-level autonomous planning module in a three-module pipeline: a hierarchical FSM-based Mission Manager orchestrates lifecycle, a multi-strategy Global Path Planner provides the global route, and this module generates real-time local trajectories.

Inputs received each 100 ms planning cycle:

Input	Source	Contents
Vehicle state	Localization module	Position (x, y) in ENU, heading ψ, speed v
Global path	Global Path Planner	Dense waypoints $(x_i, y_i, \psi_i, v_i^{max}, \kappa_i)$
SDF static map	Threat assessment module	Vehicle-centered float32 SDF tile, 0.0 = free, 1.0 = obstacle
Obstacle threats	Threat assessment module	Severity score $s \in [0,1]$, emergency brake flag, forward distance
Start/stop commands	Mission Manager	Activate/deactivate planner, configure mission-context parameters

Output each cycle: a time-indexed trajectory — a vector of state points $(x_k, y_k, \psi_k, v_k, \delta_k, t_k)$ — published to the vehicle controller, which executes the first point's v and δ as immediate commands and uses subsequent points as a feed-forward preview

Per-Cycle Planning Pipeline

The planning cycle executes seven deterministic stages in sequence:

Stage 1:
Nearest-point tracking —

advance path pointer; update lateral deviation

Stage 2:
Lookahead computation —

compute speed-adaptive lookahead distance L

Stage 3:
Longitudinal profile —

curvature speed update; multi-zone braking; obstacle modulation

Stage 4:
Kinematic reference —

propagate reference trajectory along global path

Stage 5:
Candidate enumeration —

Ackermann rollouts with inline deviation accumulation

Stage 6:
Cost evaluation —

SDF footprint cost + adaptive weighted total cost

Stage 7:
Completion check —

three-rule Mamdani fuzzy inference

If an emergency brake flag is active (from Stage 3 / threat input), the pipeline skips Stages 4–5 and publishes a zero-velocity trajectory directly. This ensures the emergency brake path is not gated by the trajectory optimization computation.

II. Global Path Tracking and Lookahead

A. Monotonic Nearest-Point Tracker

A pointer into the global path waypoint array is maintained across planning cycles. It advances monotonically: each cycle, the tracker scans forward from the current pointer until the nearest waypoint (minimum Euclidean distance to vehicle) is found. If the distance begins increasing after the first minimum, the scan terminates early. This prevents re-engagement with already-passed segments even when the vehicle deviates laterally from the path.

The monotonicity guarantee is important: without it, a vehicle that temporarily deviates laterally could have its nearest-point pointer snap backward, causing the path tracker to route the vehicle in reverse along the completed portion of the path.

B. Speed-Adaptive Lookahead Distance

The lookahead horizon scales with vehicle speed:

$$L = \text{clamp}\left(v_{current} \cdot T_{predict} \cdot factor,\ L_{min},\ L_{max}\right)$$

with $L_{min} = \min threshold$ m, $L_{max} = \max threshold$ m, $T_{predict}$ configured per mission context. The factor provides a forward bias to compensate for path curvature: a vehicle at speed v on a curved path needs to look slightly further ahead than $v \cdot T_{predict}$ to see the full turning required.

The clamp ensures a minimum lookahead even from rest (providing a pull toward the path) and caps lookahead to prevent the planner from fixating on distant waypoints through intermediate obstacles.

C. Lookahead Goal Node

The planner walks forward from the nearest-point pointer, accumulating arc length until reaching L. Only waypoints in the forward half-plane (within ±135° of the vehicle's heading) are eligible — this filters out waypoints behind the vehicle in U-turn scenarios.

A virtual goal node is placed at exactly L. ahead via linear interpolation within the final segment:

$$\mathbf{g} = \mathbf{p}_{nearest} + \frac{L - d_{accumulated}}{d_{segment}} \cdot (\mathbf{p}_{next} - \mathbf{p}_{nearest})$$

The goal node carries position (g_x, g_y), heading ψ_g, maximum speed, and curvature from the underlying waypoint. The target speed for the current cycle is set to the minimum of the path-assigned speeds encountered during the lookahead walk — ensuring the planner decelerates for the most restrictive upcoming constraint, not just the immediate one.

III. Kinematic Reference Trajectory

Before candidate trajectory generation, a **time-indexed reference trajectory** $\{\mathbf{r}_k\}_{k=0}^{N}$ is computed. This reference represents where the vehicle *should be* at each future time step $t_k = k \cdot \Delta t$ if it were following the global path exactly at the currently planned speed profile.

A. Physics-Based Propagation

The reference is generated by integrating a kinematic observer along the global path at the vehicle's current speed and target acceleration:

$$s(t) = \int_0^t v\,(\tau)\,d\tau, \quad v(\tau) = \text{clamp}\big(v_0 + a_{target} \cdot \tau,\ 0,\ v_{target}\big)$$

using trapezoidal integration. At each time step t_k, the accumulated arc-length $s(t_k)$ locates the corresponding global path waypoint by segment-walking, with sub-segment interpolation for position, heading, curvature, and steering angle.

The reference is generated over a horizon of $T_{predict}$ time steps, capped at the lookahead distance L.

B. Why a Reference Trajectory Instead of a Goal Point

Classical DWA evaluates candidates against a single goal point — the endpoint distance and heading error are the cost signals. This has a fundamental weakness: two trajectories that arrive at the same endpoint from completely different paths receive identical goal costs, regardless of whether one of them closely tracked the global path and the other veered significantly off it.

The kinematic reference trajectory resolves this: each candidate trajectory's k –th point is compared against the reference's k -th point (representing the *expected position at time* t_k). A candidate that stays close to the global path will have small deviation at every time step. A candidate that takes a shortcut or a detour will accumulate large deviation even if its endpoint is near the goal.

This **time-synchronized comparison** is the key conceptual departure from classical DWA. It transforms the cost function from an endpoint-only evaluation into a full trajectory-quality evaluation that rewards global-path fidelity throughout the entire horizon.

IV. Candidate Trajectory Generation

A. Control Space Parameterization

The planner samples only the **steering angle rate** $\dot{\delta}$— the rate of change of the front-wheel steering angle. This is a deliberate architectural choice with three justifications:

1. **Physical correspondence**: $\dot{\delta}$ maps directly to the steering actuator command. No intermediate conversion from an abstract yaw rate to a physical steering angle is needed.
2. **Separation of concerns**: longitudinal speed is planned by an independent braking physics model (Section VII) that is aware of global path geometry and obstacle distance. The trajectory optimizer samples lateral control only.
3. **Simplicity of the feasibility window**: the dynamic window for $\dot{\delta}$ has a simple structure — it shrinks with remaining distance and with vehicle speed — whereas the DWA velocity window must simultaneously bound acceleration and deceleration for both v and ω.

B. Adaptive Control Range

The maximum steering-rate magnitude u_{max} is adaptive:

Speed-based bound: at open-run speed, rapid steering causes instability. The range narrows as speed increases:

$$u_{max}^{speed} = u_0 \cdot \left(1 - \frac{v}{v_{max}}\right)$$

At $v = v_{max}$, $u_{max} \to 0$ (vehicle cannot steer at full speed — it must decelerate first). At $v = 0$, the full range is available.

Proximity-based bound: as the vehicle approaches the destination ($d_{eff} < 1.0$ m), the range tapers linearly to prevent large steering corrections at very close range:

$$u_{max}^{prox} = \text{clamp}\left(\frac{u_0 \cdot d_{eff}}{1.0},\ u_{min},\ u_0\right),$$

The operative bound is: $u_{max} = \min(u_{max}^{speed},\ u_{max}^{prox})$.

These two bounds together implement a steering authority envelope: maximum steering authority at rest far from the goal; minimal steering at speed or near the goal. This directly addresses a common failure mode in fixed-range planners: large steering oscillations as the vehicle approaches the destination.

C. Cubic Input Shaping

The N_c, candidate control inputs u_i are sampled uniformly from $[-u_{max}, +u_{max}]$. Each is mapped to an actual steering rate through a cubic nonlinearity:

$$\dot{\delta}_i = \frac{u_i^3}{\max(|u_i|, \epsilon)}$$

This cube-law shaping has a specific statistical motivation: uniform sampling of $[-u_{max}, +u_{max}]$ distributes candidates uniformly, but the cost landscape is typically deepest near $\dot{\delta} = 0$ (straight ahead is almost always a low-cost trajectory). The cubic map concentrates candidates near zero while retaining coverage of large steering inputs — functionally equivalent to importance-weighted sampling without probabilistic overhead. The result is better discrimination between near-straight trajectories (which the planner most often cares about) with no loss of coverage at extreme steering.

Figure 1: Cubic Input Shaping for Steering-Rate Candidate Selection

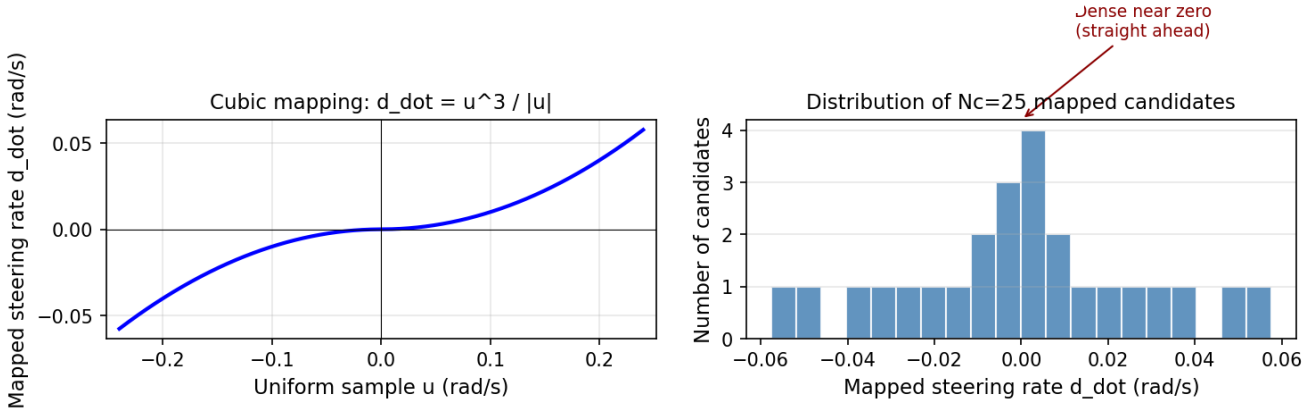

Figure 1 (conceptual): Uniform u samples mapped through $u^3/|u|$: density concentrates near zero; extremes are still represented.

D. Ackermann Kinematic Integration

Each candidate $\dot{\delta}_i$ is integrated forward for N_{steps} time steps until the trajectory arc-length equals L:

$$\delta_{k+1} = \text{clamp}\left(\delta_k + \dot{\delta}_i \cdot \Delta t,\ -\delta_{max},\ +\delta_{max}\right)$$

$$\omega_k = \frac{v_k}{L_{wb}} \tan(\delta_{k+1})$$

$$\psi_{k+1} = \psi_k + \omega_k \cdot \Delta t$$

$$x_{k+1} = x_k + v_k \cos(\psi_{k+1}) \cdot \Delta t$$

$$y_{k+1} = y_k + v_k \sin(\psi_{k+1}) \cdot \Delta t$$

$$v_{k+1} = \text{clamp}\left(v_k + a_{target} \cdot \Delta t,\ 0,\ v_{target}\right)$$

The rear-axle reference point is used (standard Ackermann convention). The integration terminates when $\sum_k v_k \cdot \Delta t \geq L$, not at a fixed number of steps — so faster-moving vehicles produce longer-duration trajectories with the same spatial coverage.

E. Inline Deviation Accumulation

During integration, the Euclidean deviation between the candidate trajectory point $\mathbf{p}_k$ and the k-th reference trajectory point $\mathbf{r}_k$ is accumulated:

$$e_k = \| \mathbf{p}_k - \mathbf{r}_k \|_2, \quad \Sigma_e = \sum_{k=0}^{N} e_k$$

This co-computation with the rollout loop avoids a separate evaluation pass and adds no asymptotic complexity: the deviation accumulation is $O(1)$ per integration step.

V. Longitudinal Speed and Acceleration Planning

Speed and acceleration are computed analytically, independently of the trajectory optimization loop.

A. Curvature-Based Path Speed Assignment

At the start of each planning cycle, the global path's per-waypoint speed limits are updated from curvature:

$$v_i^{curv} = \max\left(v_{min}, \frac{v_{max}}{1 + \alpha \cdot |\kappa_i|}\right),$$

These limits update the global path waypoints in place every cycle, so upcoming turns are visible to the lookahead tracker immediately.

B. Effective Remaining Distance

The remaining arc-length distance from the current nearest waypoint to the path end is computed. Two modifiers are subtracted to obtain the effective remaining distance:

$$d_{eff} = d_{arc} - d_{margin} - T_{react} \cdot v_{current}$$

- d_{margin}: a configurable stopping margin (e.g., 0.5 m for precision arrival, 1.5 m for open-area following). Ensures the vehicle stops before the goal, not at it, providing mechanical stopping distance.

- $T_{react} \cdot v_{current}$: reaction-time correction. The planning cycle (100 ms), communication latency, and actuator response together introduce a total delay $T_{react} \approx 0.5$s. At 2.0 m/s, this contributes 1.0 m of reaction distance that must be pre-subtracted from the available braking distance.

If a danger-level obstacle is detected, d_{eff} is additionally capped to the obstacle's forward distance minus a safety clearance margin, regardless of the path-based distance.

C. Three-Zone Braking Controller

The braking zone is selected by comparing d_{eff} against stopping-distance thresholds derived from current speed:

$$d_{hard} = \frac{v^2}{2|a_{hard}|}, \quad d_{smooth} = \frac{v^2}{2|a_{smooth}|}$$

Zone	Condition	Acceleration	Speed Target
Emergency	$d_{eff} \leq d_{hard}$	$a = a_{hard}$ (maximum deceleration)	$v_{target} = 0$
Proportional	$d_{hard} < d_{eff} \leq d_{smooth}$	$a = -v^2/(2d_{eff})$	Minimize overshoot
Accelerate	$d_{eff} > d_{smooth}$	$a = a_{smooth}$	v_{target} from path

The proportional zone is kinematically precise: $a = -v^2/(2d_{eff})$ is exactly the deceleration required to bring a vehicle from current speed to zero at d_{eff} ahead — the minimum-energy, no-overshoot stopping deceleration. This zone produces smooth, controlled stops without oscillation around the stopping point.

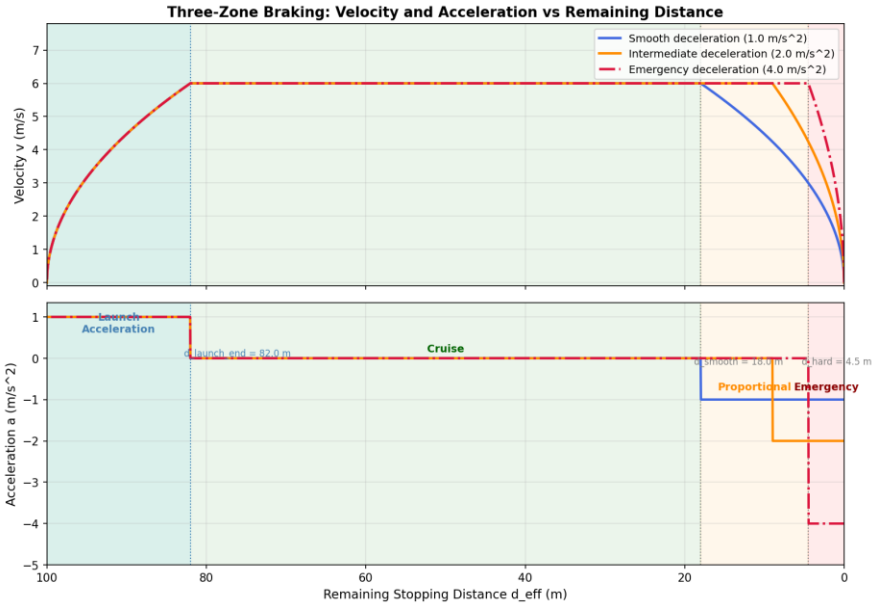

Figure 2 (conceptual): Speed profile vs. remaining distance — accelerate zone (flat, near v_{max}); proportional zone (parabolic descent to zero); emergency zone (steep linear descent).

D. Obstacle Severity Speed Modulation

For caution-level obstacles (not triggering emergency brake), the threat assessment module provides a continuous severity score $s \in [0,1]$. The target speed is modulated:

$$v_{target} \leftarrow \max\left(v_{min},\ v_{target} \cdot (1 - 0.5 \cdot s)\right)$$

A maximum-severity caution obstacle **(**$s = 1.0$**)** reduces speed by 50%. A low-severity obstacle **(**$s = 0.2$**)** reduces speed by 10%. This continuous modulation avoids the binary behavior (full speed or emergency stop) of hard-threshold obstacle responses and produces smoother deceleration profiles in crowded environments.

V1. Obstacle Safety: Vehicle-Footprint SDF Evaluation

A. The SDF Representation

The static map is provided as a float32 Signed Distance Field tile. Each cell value encodes the occupancy probability of the corresponding spatial location: 0.0 = fully free, 1.0 = fully occupied obstacle. Intermediate values represent proximity gradients near obstacle boundaries, providing a smooth cost signal for near-miss avoidance without the binary on/off cost of raw occupancy maps.

The SDF tile is vehicle-centered: the tile origin is synchronized to the vehicle's current ENU position at the start of every planning cycle, before any lookup. The ENU-to-pixel transform is:

$$p_x = \left\lfloor \frac{x_w - x_{veh} + B}{r} \right\rfloor, \quad p_y = \left\lfloor \frac{S - (y_w - y_{veh})}{r} \right\rfloor$$

where $B = N_{cols} \cdot r/2$ and $S = N_{rows} \cdot r/2$ are the tile half-extents in meters. This centering ensures the vehicle is always at the center of the valid SDF region regardless of its absolute ENU position on the operational map — critical for correct operation near map boundaries or in large-scale environments.

B. Full Vehicle-Footprint Sweep

At each trajectory point (c_x, c_y, ψ)(rear-axle reference), a grid of footprint sample points is generated covering the full vehicle body:

- Longitudinal span: $[0, L_{veh}]$ from rear axle forward, at grid spacing r.
- Lateral span: $[-W_{veh}/2, +W_{veh}/2]$, at grid spacing r.

Each footprint cell (l_x, l_y) is rotated to the ENU frame and queried:

$$\begin{bmatrix} x_w \\ y_w \end{bmatrix} = \begin{bmatrix} c_x \\ c_y \end{bmatrix} + \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} l_x \\ l_y \end{bmatrix}$$

The maximum SDF value across all footprint cells at each trajectory point k is the point collision cost c_k.

C. Why Full-Footprint vs. Point-Based Checking

Classical DWA uses the obstacle distance to the vehicle's center point (or center of mass) as the obstacle cost. This significantly underestimates collision risk for wide vehicles navigating narrow gaps: a trajectory that keeps the vehicle center 0.5 m from an obstacle may still clip the vehicle's corner.

Full-footprint evaluation eliminates this underestimation. For a vehicle 1.5 m wide, the difference between center-point and full-footprint collision detection is ± 0.75 m on each side — sufficient to miss real collisions in typical off-road environments. The computational cost increase (from one point lookup per trajectory step to $N_{footprint}$ lookups) is bounded and predictable.

D. Position-Aware Collision Penalty

The obstacle cost for a candidate trajectory is not a simple sum over all points. If the highest-cost point i_{max} (the closest predicted obstacle encounter) occurs in the second half of the trajectory ($i_{max} > N/2$), the peak cost is discounted:

$$C_{obs} = c_{i_{max}} \cdot \left(1 - \frac{i_{max}}{N}\right)$$

This encodes the physical heuristic that a near-term obstacle encounter (in the first half of the trajectory) is immediately dangerous — the vehicle is committed to that path segment within the current planning horizon. A distant obstacle encounter (second half of the trajectory) is less critical: the vehicle will replan before reaching that point.

When no trajectory point reaches the hard collision threshold (1.0), the average footprint cost across all trajectory points serves as the obstacle cost, providing a smooth gradient cost for proximity avoidance.

A trajectory is declared **infeasible and discarded** only when the maximum per-point footprint cost equals or exceeds 1.0 (fully occupied cell). This hard-cut criterion eliminates all trajectories that would collide with a confirmed obstacle.

VII. Cost Function and Adaptive Weighting

A. Four Cost Terms

Four independent cost terms are computed for each feasible candidate trajectory:

Goal cost C_{goal}— Gaussian-shaped penalty on the endpoint's Euclidean distance to the lookahead goal node:

$$C_{goal} = 1 - \exp\left(-\frac{\| \mathbf{p}_{end} - \mathbf{g} \|^2}{2L^2}\right)$$

The Gaussian normalization by L^2 ensures the cost scale is consistent across different lookahead distances. A trajectory that ends at exactly the goal node receives $C_{goal} = 0$.

Heading cost $C_{heading}$ — Gaussian-shaped penalty on heading error at trajectory end:

$$C_{heading} = 1 - \exp\left(-\frac{(\psi_{end} - \psi_g)^2}{2\pi^2}\right)$$

The π^2 normalization treats a 180° heading error as approximately maximum cost.

Path deviation cost C_{path} — penalty on cumulative deviation from the kinematic reference trajectory:

$$C_{path} = 1 - \exp\left(-\frac{1}{2}\left(\frac{20 \cdot \Sigma_e / L}{1.0}\right)^2\right)$$

The factor $20 \cdot \Sigma_e / L$ normalizes the cumulative deviation by the lookahead distance and amplifies it: a mean per-step deviation of 5% of L $\Sigma_e = 0.05L/20$) maps to unit normalized cost. Larger deviations saturate rapidly toward 1.0.

Static map cost C_{obs} — vehicle-footprint SDF result from Section VIII, clamped to [0,1].

B. Sigmoid Distance-Adaptive Weighting

A central weakness of fixed-weight trajectory cost functions is terminal oscillation: a planner optimized for path-following keeps tracking the path even when the vehicle is close to the goal, causing it to overshoot and loop back. A planner optimized for goal attraction pulls the vehicle off the path early in the mission.

The sigmoid-adaptive weighting eliminates this trade-off by smoothly transitioning the weight balance based on remaining distance. Define the normalized remaining distance:

$$\rho = \frac{d_{eff}}{d_{threshold}}$$

and the sigmoid scale factor:

$$\sigma(\rho) = \frac{1}{1 + e^{2(\rho - 1)}}$$

σ is close to 0 when $\rho \gg 1$ (far from goal) and close to 1 when $\rho \ll 1$ (near goal). The adapted weights are:

$$w_g^* = w_g \cdot (1 + 2\sigma), \quad w_h^* = w_h \cdot (1 + 2\sigma)$$

$$C_{total} = w_g^* \cdot C_{goal} + w_h^* \cdot C_{heading} + w_p \cdot C_{path} + w_{obs} \cdot C_{obs}$$

When $\rho \gg 1$ ($\sigma \approx 0$): $w_g^* = w_g$, $w_h^* = w_h$— path cost dominates, vehicle follows the global path.

When $\rho \approx 0$ ($\sigma \approx 1$): $w_g^* = 3w_g$, $w_h^* = 3w_h$— goal and heading weights triple, vehicle pulls directly toward the goal with correct heading.

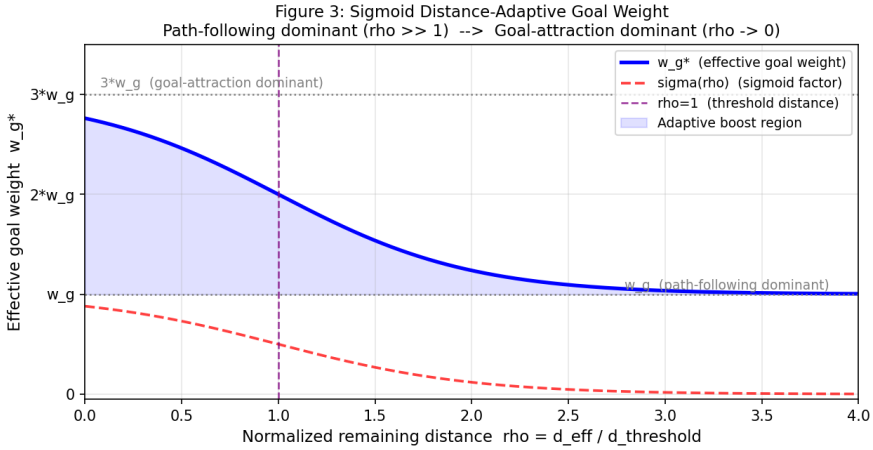

Figure 3 (conceptual): Effective w_g^* vs. remaining distance ratio ρ. Flat at w_g for large ρ; sigmoid ramp to $3w_g$ as $\rho \to 0$.

The sigmoid shape (rather than a step or linear ramp) is important: it concentrates the weight transition around $\rho = 1$ (at the threshold distance), producing a smooth behavioral transition without an abrupt discontinuity in the cost landscape.

VII. Fuzzy Path Completion Detection

A. The Failure Mode of Threshold-Based Completion

Standard completion detection uses a single distance threshold: when $\| \mathbf{p}_{vehicle} - \mathbf{g}_{final} \| \leq \epsilon$, the mission is complete. This has two failure modes:

- **False negatives**: the vehicle oscillates around the goal point due to control-loop dynamics, repeatedly passing through the threshold zone without remaining in it long enough to be detected. The mission never completes.
- **False positives**: the vehicle passes through the threshold zone while still in motion (e.g., at speed) and is declared complete before fully stopping.

Both failure modes are common in real deployments, particularly on curved paths where the final goal is approached from an angle.

B. Progress Metric

A monotonically non-decreasing progress metric is maintained:

$$\rho_{track} = \max\left(1 - \frac{d_{arc,remaining}}{d_{arc,total}},\ \frac{i_{nearest}}{N_{path}}\right)$$

Taking the maximum of two independent progress signals ensures ρ_{track} never decreases even if the path is partially replanned (which could reduce $d_{arc,remaining}$) or if the nearest-point pointer jumps (which could change $i_{nearest}$). A monotonically non-decreasing metric prevents the completion scorer from "forgetting" progress.

C. Membership Functions

Five linguistic variables are defined over the relevant physical quantities:

Variable	Type	Membership Function
nearGoal	Distance d	Left-shoulder trapezoid
veryNearGoal	Distance d	Left-shoulder trapezoid
stopped	Speed v	Left-shoulder trapezoid
highProgress	Progress ρ_{track}	Right-shoulder trapezoid
passedGoal	Boolean geometry	Crisp binary

The left-shoulder trapezoids for distance and speed encode "near" and "slow" as gradually increasing membership as the value decreases — reflecting the continuous nature of physical approach and deceleration. The right-shoulder trapezoid for progress encodes "almost done" as gradually increasing above 80% completion. The double-confirmation in *passedGoal* (final waypoint AND lookahead index both behind the vehicle) prevents premature overshoot detection when only one signal momentarily passes behind.

D. Three Mamdani Rules

$$R_1 = \min(\text{nearGoal},\ \text{stopped}) \quad \text{[Stopped close to goal]}$$

$$R_2 = \text{passedGoal} \quad \text{[Vehicle has overshot]}$$

$$R_3 = \min(\text{veryNearGoal},\ \text{highProgress}) \quad \text{[Very close]}$$

$$S_{complete} = \max(R_1, R_2, R_3)$$

Rule 1 handles normal arrival: the vehicle decelerates to near-stop within 3 m of the goal. This is the expected case for well-tuned operation.

Rule 2 handles overshoot: if the final path point passes behind the vehicle, the mission is complete regardless of speed. This catches cases where the vehicle's momentum carries it slightly past the goal. Without this rule, an overshot vehicle would recirculate indefinitely trying to return to a goal now behind it.

Rule 3 handles the high-confidence close approach: even if not fully stopped, a vehicle within 0.2 m of the goal that has covered ≥ 95% of its total route is considered arrived. This catches cases where the path terminates slightly short of the physical goal location due to GPS discretization.

The OR combination (max) means any single rule achieving high score triggers completion — the rules encode three sufficient conditions for arrival, not conjunctive necessary conditions.

When S_complete≥threshold, the planner transitions to its completed state and notifies the Mission Manager.

IX. Comparison with Classical DWA

Table II: Classical DWA vs. This Planner — Complete Comparison

Dimension	Classical DWA	This Planner	Justification for Departure
Control space	(v, ω) velocity pairs	$\dot{\delta}$ steering rate only	Physical Ackermann correspondence; speed planned independently
Motion direction	Predominantly forward reactive motion	Segment-aware forward/reverse execution	Supports constrained-space backing while preserving deterministic structure
Kinematic model	Unicycle (differential drive)	Ackermann rear-axle	Correct for steered vehicles
Trajectory length	Fixed time horizon	Arc-length = lookahead distance L	Speed-adaptive spatial coverage
Reference for deviation cost	None (goal point only)	Time-indexed kinematic reference trajectory	Time-synchronized path tracking

Dimension	Classical DWA	This Planner	Justification for Departure
Control input shaping	Uniform sampling	Cubic nonlinearity $u^3/\|u\|$	Importance-weighted near-straight sampling
Steering authority range	Fixed	Adaptive (speed-based + proximity-based)	Prevent steering oscillations at speed and near goal
Obstacle evaluation	Point-based nearest obstacle	Full vehicle-footprint SDF sweep	Correct for wide vehicles in narrow gaps
Collision penalty	Binary (feasible / not)	Position-weighted continuous	Near-term encounters penalized more than distant ones
Cost weighting	Fixed	Sigmoid distance-adaptive (1× → 3×)	Eliminate terminal oscillation without compromising open-path tracking
Completion detection	Hard distance threshold	Three-rule Mamdani fuzzy inference	Robust to oscillation, overshoot, and GPS discretization
Output format	(v^*, ω^*) one-step command	Full trajectory $(x_k, y_k, \psi_k, v_k, \delta_k, t_k)$	Feed-forward preview for vehicle controller

X. Relevance and Safety in the AI Era

A. Current State of AI-Based Motion Planning

AI-based motion planning approaches have proliferated significantly across multiple paradigms. Classical approaches include model predictive control with learned models [A1], imitation learning from expert trajectories [A2], reinforcement learning in simulation with sim-to-real transfer [A3], and diffusion-based trajectory generation [A4]. More recent methods leverage vision transformers for end-to-end sensorimotor control [A8], learned neural trajectory planning with safety regularization [A9], and integration with large language models for decision-level augmentation [A10]. These systems have demonstrated impressive performance on benchmark evaluation tracks and in controlled deployments. They offer adaptive behavior in novel environments, implicit terrain cost encoding, and the potential for end-to-end optimization from perception to control.

However, for safety-critical deployments in outdoor physical environments, these approaches share common limitations that the classical trajectory optimizer addresses by design. Even recent efforts to verify learned motion planners [A11] or propose hybrid classical-neural combinations [A12] acknowledge this gap — safety guarantees in learning-based systems remain probabilistic or require post-hoc classical filtering:

Safety Property	Classical Trajectory Optimizer	AI-Based Planner
Kinematic feasibility of every output trajectory	Guaranteed by construction	Requires post-validation
Hard obstacle avoidance guarantee	Guaranteed by SDF feasibility cut	Not guaranteed; probabilistic or requires auxiliary filter
Deterministic behavior for identical inputs	Yes	No (stochastic sampling / inference)
Bounded computation per cycle	Yes	Variable (inference time depends on model and hardware)
Graceful degradation when emergency brake needed	Immediate zero-velocity fallback	May require additional logic
Interpretable failure cause	Yes (no feasible trajectory, SDF violation)	No; black-box or high-dimensional latent
Certification path (ISO 26262 / similar)	Established for deterministic code	Not yet established; verification methods emerging [A11]

B. The Kinematic Feasibility Argument

This is the central safety argument: **every trajectory produced by the Ackermann integration is, by construction, drivable by the vehicle**. The integration propagates exactly the kinematic model that governs the physical vehicle; if the integration can execute a steering command within the joint limits, the physical vehicle can follow it.

AI planners do not share this property. A trajectory generated by a neural network has no inherent guarantee that consecutive points are kinematically reachable from each other. A post-hoc kinematic validation step can detect infeasibility, but in a real-time planning system, an infeasible trajectory detected after generation leaves a gap: either the last feasible trajectory is re-used (potentially stale by several hundred milliseconds) or a zero-velocity fallback is triggered.

In a safety-critical setting, the kinematic feasibility guarantee is not a performance feature — it is a **behavioral bound** that the system can commit to regardless of environmental novelty.

C. The SDF Hard-Cut Guarantee

Similarly, the hard-cut criterion (any trajectory with a footprint cell ≥ 1.0 is discarded) provides a **provable obstacle avoidance guarantee within the SDF's validity region**. Given a correctly maintained SDF and a correctly implemented footprint check, no published trajectory will command the vehicle through a confirmed static obstacle. This guarantee holds for all inputs, not just those represented in a training distribution.

AI planners that optimize a soft obstacle cost cannot provide an equivalent hard-cut guarantee without an explicit post-hoc feasibility filter. The classical planner's hard cut is the simpler and more verifiable approach.

D. The Appropriate Role of AI in Motion Planning Architecture

Rather than replacing the classical trajectory optimizer, AI components can augment it at specific leverage points:

AI-enhanced SDF: rather than computing the SDF purely from LiDAR-detected obstacles, a semantic segmentation network can classify terrain traversability (soft ground, vegetation, slope hazard) and inject these as elevated SDF values. The classical planner's footprint sweep then naturally avoids terrains the AI has classified as dangerous — without the AI needing to generate trajectories.

AI-predicted obstacle motion: a trajectory prediction network can anticipate the future positions of dynamic obstacles and pre-populate future SDF frames. The classical planner's evaluation of trajectory points at time t_k can then query the SDF frame at t_k rather than the current SDF, providing dynamic obstacle avoidance without modifying the planner architecture.

AI-suggested reference trajectory: the kinematic reference trajectory (Section V) could be replaced or augmented by a learned reference that reflects operator preference patterns (preferred clearance from obstacles, preferred route curvature in open terrain). The classical deviation cost then penalizes departure from the AI-suggested style, not just departure from the geometric global path

AI-tuned cost weights: the fixed base weights w_g, w_h, w_p, w_{obs} could be adapted per-environment by a learned model that observes recent trajectory performance (how often trajectories are discarded, how large the average deviation cost is) and adjusts weights accordingly. The sigmoid adaptive mechanism remains, but the base weights it operates on are learned rather than hand-tuned.

In each of these augmentations, the AI provides improved information to the classical planner; the classical planner's kinematic correctness, SDF hard-cut guarantee, and deterministic output remain unchanged.

E. When AI Planners Become the Right Choice

The transition point at which AI planners should replace the classical trajectory optimizer is well-defined: when an AI planner can be proven to produce kinematically feasible, SDF-feasible, computationally bounded trajectories across all inputs in the operational envelope. "*Proven*" here means verified by exhaustive testing or formal methods over the full input space — not demonstrated on a held-out test set.

Until that verification is achievable (which requires both more powerful verification tools and more constrained AI architectures), the classical trajectory optimizer remains the appropriate primary planner. AI augmentation at the perception, prediction, and parameter-tuning interfaces provides performance improvement while the safety-critical planner maintains its formal guarantees.

XI. Advantages Over End-to-End Control and Alternative Motion Planning Methods

A. Advantages Over End-to-End Control Algorithms

End-to-end control systems, particularly recent vision-transformer-based approaches [A8] and learned neural planners [A9], directly infer actuator commands from perception inputs, often without explicit trajectory-level safety structure. While this can provide strong adaptation in benign conditions and scaling benefits through self-supervised learning, it weakens hard guarantees that are essential for physical safety in failure-critical environments.

The proposed motion-planning architecture provides explicit advantages:

1. **Trajectory-level safety envelope:** every candidate is kinematically integrated and obstacle-checked before publication, unlike black-box end-to-end policies.
2. **Hard feasibility cut:** trajectories with footprint collision are discarded deterministically; no probabilistic fallback.

2. **Bounded compute path:** fixed candidate count and deterministic scoring bound worst-case runtime; no model-inference variance.
3. **Structured emergency fallback:** zero-velocity output path exists independent of optimizer success; no dependency on learned policy robustness.
4. **Interpretability:** costs and completion logic are decomposed into explainable terms; failure root causes are traceable.
5. **Safety verification potential:** deterministic algorithms admit formal verification; learned end-to-end policies do not [A11].

Table III: This Motion Planner vs. End-to-End Control (Recent and Classical)

Criterion	End-to-End Control [A8], [A9]	Proposed Trajectory Optimizer
Kinematic validity guarantee	Limited or post-hoc validation	Explicit by construction
Collision safety guarantee	Probabilistic; may require learned auxiliary safety net	Deterministic hard-cut via footprint SDF
Runtime predictability	Variable (model inference + potential safety checks)	Bounded by fixed sampling/evaluation budget
Failure diagnostics	Low (learned representations)	High (cost terms + feasibility flags)
Safe fallback behavior	Model-dependent or requires explicit fallback layer	Architecture-defined zero-velocity path
Certification suitability	Limited; formal verification open [A11]	Stronger; verification methods established

B. Comparison with Common Motion Planning Systems

Table IV: Comparison Across Relevant Motion Planning Systems

Motion Planning System	Strengths	Typical Limitation	Why the Proposed Planner Is Better for Safety-Critical Off-Road Use
Pure waypoint tracking (PID/Stanley/Pure Pursuit)	Simple, efficient	Weak local obstacle reasoning	Adds explicit trajectory optimization with SDF collision feasibility
Classical DWA	Fast reactive avoidance	Unicycle assumptions, fixed horizon, endpoint-only bias	Ackermann rollouts, distance horizon, time-indexed reference tracking
Nonlinear MPC (single optimizer)	Strong model-based control	Sensitive to initialization and compute spikes	Uses bounded candidate set with deterministic feasibility filtering
TEB / elastic-band methods	Smooth trajectories in narrow spaces	Can struggle with global consistency and terminal behavior	Uses global-path time-synced reference + adaptive weighting + fuzzy completion
MPPI / stochastic trajectory sampling	Handles nonlinear costs well	Sampling variance and runtime variability	Fixed deterministic sampling and scoring pipeline
RL or imitation local control policy	Strong adaptation potential	Hard-to-verify safety and fallback behavior	Explicit kinematic and collision guarantees with interpretable decisions
Proposed planner	Balanced safety and performance	Requires SDF/map quality	Best trade-off of deterministic safety, interpretability, and real-time behavior

C. System-Level Position in an Autonomous Stack

At full-system level, this planner is stronger than monolithic local-control pipelines because it preserves clean interfaces with the global planner and mission manager: global path intent comes from above, hard safety and feasibility filtering happens locally, and mission completion is reported through explicit state transitions. This modularity prevents local-control improvements from destabilizing mission-level behavior and allows safe incremental adoption of AI enhancements.

XII. Discussion

A. Computational Complexity

The per-cycle computation consists of: - One kinematic reference generation: $O(N_{path})$ for the global path walk. - $N_c = 25$ candidate integrations: each $O(N_{steps})$ where $N_{steps} = L/(v \cdot \Delta t)$. At $L = 10$ m, $v = 1.0$ m/s, $\Delta t = 0.1$ s, $N_{steps} = 100$. - Per step: one SDF footprint check of $A = \pi r^2$ cells. For a typical vehicle footprint at 0.2 m grid resolution: (7×5)=35 cells. - Total per cycle: 25×100×35=87,500 SDF lookups.

At modern CPU lookup speeds (DRAM access time ≈ 50 ns), this is ≈4.4 ms of SDF lookup time, well within the 100 ms cycle budget. The CPU computation (trigonometric operations in the Ackermann integration) adds a comparable amount; total planning time is typically 5–15 ms per cycle, leaving 85–95 ms for other module tasks.

B. Parameter Sensitivity

The most sensitive parameters are:

- $u_0 =$: the maximum steering rate. Reducing this improves stability at high speed; increasing it improves turn-in response. Should be set based on vehicle's steering actuator bandwidth.
- α in curvature speed: increasing reduces speed more aggressively on curves; decreasing is more permissive. Tune to the operational area's tightest bends.
- $d_{threshold}$ in sigmoid weighting: the distance at which goal attraction begins to dominate. Set to the vehicle's typical deceleration distance from operating speed to stop.
- Fuzzy completion thresholds (for nearGoal): should be matched to the GPS accuracy and vehicle speed at approach. Higher GPS accuracy allows tighter thresholds.

C. Limitations

The planner assumes the global path is valid and updated less frequently than the local planner runs. If the global path is stale by more than several seconds (e.g., due to a temporary communication failure with the path planner), the kinematic reference trajectory and nearest-point tracking may diverge from the vehicle's actual situation. The emergency-brake path remains active in this case, ensuring the vehicle stops safely, but path-following fidelity degrades until a fresh global path is received.

D. Reverse Maneuver Capability (Bidirectional Extension)

Many real deployments require short reverse maneuvers: aisle deconfliction in warehouses, parking corrections, dead-end recovery, headland alignment in agriculture, and loader/excavator repositioning. The current planner architecture can incorporate reverse operation without changing its fundamental real-time structure.

1. Segment-level direction intent: The global path is represented as segments carrying context and direction metadata. A segment can be executed in FORWARD or REVERSE mode, and the active direction is propagated into candidate rollout generation and trajectory publication.

2. Signed-speed Ackermann rollout: Reverse is represented by a negative target speed with the same steering-rate sampling and kinematic integration loop.

3. Direction-aware waypoint gating: Forward-only half-plane filters must be mirrored for reverse segments. Equivalently, front-of-vehicle tests should be replaced by direction-aware longitudinal projection, so lookahead and nearest-point logic do not reject valid reverse waypoints.

4. Reverse safety policy: The same footprint of SDF hard cut remains at the primary safety gate. For deployment robustness, reverse-specific guards are recommended: (i) lower speed cap, (ii) tighter steering-rate bound, (iii) hysteresis when switching between forward/reverse, and (iv) rear-sector obstacle confidence checks before enabling reverse motion.

5. Runtime and safety invariants preserved: Reverse extension does not increase asymptotic complexity. Candidate count, per-step footprint checks, and deterministic compute bounds remain unchanged; collision feasibility continues to be enforced by the same hard-cut criterion.

CONCLUSION

The trajectory optimizer presented in this paper advances beyond classical DWA in core safety-and-performance dimensions, each motivated by a concrete limitation of the original formulation, and extends this foundation with bidirectional reverse maneuver capability. The steering-rate control space, distance-based horizon, kinematic reference trajectory, sigmoid-adaptive weighting, full-footprint SDF evaluation, position-weighted collision penalty, fuzzy completion detection, and segment-aware reverse execution collectively produce a planner that is more robust in normal operation and more safety-assured in edge cases than DWA.

The approach is broadly applicable across diverse operational domains for any Ackermann-steered vehicle platform. Its core strength — deterministic, kinematically-correct trajectory generation with hard obstacle safety guarantees — is equally valuable across off-highway and indoor environments: Off-highway applications (agricultural tractors, golf carts, utility vehicles, ATVs, excavators, lawn mowers) benefit from the approach's robustness to terrain variation, GPS uncertainty, and the mechanical diversity of field-deployed platforms. Off-highway environments demand explicit kinematic correctness and deterministic safety enforcement because sensors are often less precise and fallback mechanisms must be reliable in remote locations.

Indoor applications (warehouse AMRs, parking attendants, manufacturing robots, retail delivery systems) benefit from the precise Ackermann kinematic modeling, adaptive lookahead for tight spaces, and full-footprint SDF collision checking in constrained aisles and corridors. Indoor environments present different challenges — high-precision localization, dense static obstacles, frequent tight-gap navigation — that the planner's architecture directly addresses through parameter tuning (finer SDF resolution, shorter lookahead, tighter completion thresholds) without algorithmic changes.

The comparison with AI-based motion planning approaches establishes a clear complementarity: AI excels at perception-driven environmental representation and adaptive behavior in novel conditions; the classical trajectory optimizer provides kinematic feasibility guarantees, SDF hard-cut obstacle avoidance, deterministic computation, and certifiable safety properties that remain necessary for physically deployed autonomous systems across all domains. The recommended long-term architecture combines AI augmentation of the planner's inputs (SDF quality, obstacle prediction, cost weight adaptation) with retention of the classical optimizer as the trajectory generation and safety enforcement core.

The planner's safety properties — kinematic correctness by construction, hard obstacle avoidance guarantee within the SDF validity region, deterministic bounded computation, and interpretable failure modes — are not features that degrade as AI planning capability improves. They are permanent architectural requirements for systems that operate in consequential physical environments, whether in open fields, construction sites, warehouses, parking structures, or any other domain where Ackermann-steered vehicles must navigate reliably.

REFERENCES

[1] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, 1997.

[2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. MIT Press, 2011.

[3] J. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[4] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[5] E. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[6] E. H. Mamdani, “Application of fuzzy algorithms for control of simple dynamic plant,” *Proc. IEE*, vol. 121, no. 12, pp. 1585–1588, 1974.

[7] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” Carnegie Mellon Univ., Tech. Rep. CMU-RI-TR-92-01, 1992.

[8] S. Thrun *et al.*, “Stanley: The robot that won the DARPA Grand Challenge,” *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006.

[9] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, “Trajectory modification considering dynamic constraints of autonomous robots,” *Robot. Auton. Syst.*, vol. 61, no. 6, pp. 540–554, 2013.

[10] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, “Predictive active steering control for autonomous vehicle systems,” *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, 2007.

[11] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. Rehg, B. Boots, and E. A. Theodorou, “Information-theoretic model predictive control: Theory and applications to autonomous driving,” *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, 2018.

[A1] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, pp. 187–210, 2018.

[A2] D. A. Pomerleau, “ALVINN: An autonomous land vehicle in a neural network,” *NeurIPS*, 1989.

[A3] J. Chen *et al.*, “Learning to navigate using mid-level visual priors,” *CoRL*, 2019.

[A4] T. Pearce *et al.*, “Imitating human behaviour with diffusion models,” *ICLR*, 2023.

[A5] H. Caesar, C. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles,” *arXiv preprint arXiv:2106.11810*, 2021.

[A6] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Sallab, S. Yogamani, and P. Pérez, “Deep reinforcement learning for autonomous driving: A survey,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.

[A7] C. Chi, Z. Feng, S. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion Policy: Visuomotor policy learning via action diffusion,” *Proc. Robotics: Science and Systems (RSS)*, 2023.

[A8] T. Akada, D. Paxton, and O. Savarese, “End-to-end autonomous driving with transformers,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2024.

[A9] M. Janner, X. Girase, Y. Du, M. Anandkumar, and S. Levine, “Planning diffusers: Diffusion models for closed-loop motion planning,” in *Proc. Robot. Learn. (CoRL)*, 2024.

[A10] S. Park, J. Liang, O. Salzmann, and M. Jain, “Dialogue agents for autonomous vehicles: Safety-aware decision-making through language models,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024.

[A11] D. Dey, K. Chakraborty, and R. Hanai, “Verifying learned motion planners: Safety constraints and reachability analysis for neural networks in autonomous systems,” *IEEE Trans. Autom. Sci. Eng.*, vol. 22, no. 2, pp. 1021–1036, 2025.

[A12] F. Zhou, H. Chen, and C. J. C. H. Watkins, “Hybrid classical-neural planning: Safe autonomy through deterministic fallback and learned augmentation,” in *Proc. Advances Neural Inf. Process. Syst. (NeurIPS)*, 2024.

[A13] P. Luc, A. Srivastava, and L. Cao, “Diffusion models for multimodal trajectory prediction: Uncertainty quantification in autonomous systems,” *arXiv preprint arXiv:2412.09874*, 2024.

[A14] N. Varma, R. Mania, and Y. Bengio, “Vision-to-motor control via contrastive representation learning: Deployment on mobile robots,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024.

ABOUT TATA ELXSI

Tata Elxsi is among the world's leading providers of design and technology services across industries including automotive, broadcast, communications, healthcare, and transportation. Tata Elxsi works with leading OEMs and suppliers in the transportation industries for R&D, design, and product engineering services from architecture to launch and beyond. It brings together domain experience across autonomous, electric, connected vehicle technologies and software-defined vehicles (SDVs), supported by a worldwide network of design studios, development centres, and a global talent pool of over 12,500 engineers and specialists.

For more information, please visit: Off-Highway Equipment: Autonomous, Electric, and Connected Solutions

---

Canada | China | France | Germany | India | Ireland | Japan | Netherlands | Portugal | South Africa | Spain | UAE | UK | USA

Tata Elxsi Limited, ITPB Road, Whitefield Bangalore 560048, India

+91 80 2297 9123

info@tataelxsi.com

www.tataelxsi.com

TATA ELXSI