

Multi-Strategy Global Path Planning for Autonomous Off-Road Navigation



CONTENTS

➤	ABSTRACT	04
➤	Introduction	05
➤	Problem Statement	07
➤	Approach/Strategy Selection	08
➤	Conclusion	28
➤	References	29
➤	About Tata Elxsi	30



AUTHOR



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.

ABSTRACT

Global path planning for autonomous ground vehicles in mixed structured and unstructured off-road environments presents unique challenges not adequately addressed by purely learned or purely classical approaches. Structured traversable corridors – defined trails, tracks, pathways, or designated routing lanes – co-exist with open unstructured terrain; vehicle start configurations range from perfectly aligned on-path to severely misaligned in open ground. This paper presents a three-strategy global path planner that selects among structured-path graph routing, a Bézier curve connector, and Hybrid A* search based on geometric analysis of the start and goal configuration. The result is a planner that handles the full diversity of vehicle-environment configurations at runtime without pre-recorded paths, user-provided start hints, or online learning. We characterize the robustness properties of each strategy, analyze their computational bounds, and compare them against contemporary learned planning approaches. We argue that the combination of geometric reliability, kinematic feasibility guarantees, and deterministic runtime makes this style of planner a necessary component of any safety-assured autonomous navigation stack – even as AI-based planning demonstrates remarkable capability improvements.

INTRODUCTION

A. Problem Context

Global planning for outdoor autonomy is no longer a purely shortest-path problem. In practical mixed environments, vehicles transition between structured corridors and open drivable space, often from arbitrary initial heading and offset. A globally useful planner must therefore do more than optimize distance: it must generate routes that are geometrically plausible, kinematically executable, and computationally stable under real-time mission constraints.

This requirement creates a tension. Structured-graph routing is efficient and interpretable but brittle for off-network starts. Unstructured search methods are flexible but can be expensive and less predictable over large operating areas. End-to-end learned route generation improves adaptation, but deployment still demands explicit validity checks and deterministic fallback behavior.

The planner in this paper addresses this operational setting with an explicit multi-strategy formulation that selects and composes routing methods at runtime based on start/goal geometry and heading alignment.

B. Literature Survey

Classical deterministic graph search remains the backbone of structured-map route planning: Dijkstra guarantees optimal graph paths [6], while A* preserves optimality with faster heuristic search under admissibility assumptions [7]. For high-dimensional and nonholonomic planning, PRM and related sampling methods broaden feasible search spaces [8],

and Hybrid A* has become a practical standard for car-like kinematic feasibility [2].

System-level surveys in autonomous driving consistently report that no single family dominates across all contexts: graph methods excel on known topology; search/sampling methods are preferred for sparse topology or off-network maneuvers [9]. Recent AI-based route-generation approaches add representation flexibility and adaptation [A1]-[A3], but still rely on geometric/kinematic validation and safety-layer arbitration before deployment in safety-critical systems.

C. Gap Statement

The unresolved gap is not the absence of strong individual planners; it is the absence of a deterministic integration policy that chooses among them under mixed-geometry missions. Many pipelines either over-commit to one planning paradigm or rely on implicit planner switching without explicit geometric criteria, which weakens reproducibility and diagnosability.

This paper fills that gap with a strategy-selection framework that explicitly maps mission geometry to planner choice and fallback order, while preserving kinematic validity and bounded runtime behavior.

D. Pros/Cons Summary of Available Approaches

As in standard planning papers, methods are compared here by assumptions, runtime behavior, kinematic validity, and deployment robustness.

Global Planning Approach	Pros	Cons
Graph routing (Dijkstra/A*) [6], [7]	Fast and globally optimal on explicit topology	Limited off-network recovery and heading-alignment handling
Hybrid A* [2]	Kinematically feasible and geometry-aware	Higher cost on large domains if used end-to-end
Sampling methods (PRM/RRT family) [8]	Flexible in irregular spaces	Stochastic quality/runtime; post-processing often needed
Learned route generation [A1]-[A3]	Strong adaptation and representation learning	Limited determinism and weaker failure interpretability
Multi-strategy deterministic planner (this work)	Explicit strategy arbitration + bounded fallback with feasibility checks	Depends on map quality and topology completeness

This comparison highlights why mixed structured/unstructured deployments benefit from explicit strategy arbitration rather than one-method-only planning.

E. Contributions

1. A deterministic multi-strategy selector with explicit geometry-based decision criteria.
2. A heading-aware entry policy that reduces misrouted starts on bidirectional path networks.
3. A connector-and-fallback composition combining Bézier entry with Hybrid A* recovery under infeasibility.
4. A global-path speed assignment and smoothing process that preserves kinematic consistency.
5. A systems-level comparison showing why deterministic multi-strategy planning remains relevant alongside AI route-generation methods.

I. Problem Definition and Environment Model

A. Operational Domain

The planner operates in diverse environments characterized by:

- A defined **traversable path network** – a sparse graph of GPS-anchored waypoint sequences representing preferred travel corridors (trails, tracks, roads, marked lanes, or defined routing corridors within an operational area).
- **Open drivable terrain** – areas outside the path network that are geometrically traversable but have no predefined routing.
- **Static obstacles** – known boundaries, walls, structures encoded as polygon obstacles in the map.
- **Dynamic obstacles** – perceived at runtime by the threat assessment module; not a global planning responsibility (handled by the local motion planner).

Operational Context Diversity: The multi-strategy global path planner is architected for deployment across diverse vehicle platforms and operational domains: (1) **Commercial and Industrial Applications** – golf course maintenance and groundskeeping, facility and grounds management, warehouse and logistics operations, robotic lawn mowing and turf care, parking structure inspection and management, retail delivery and material handling, automated sweeping and cleaning operations; (2) **Agricultural and Off-Road Operations** – crop spraying and precision agriculture, invasive species management, forestry inventory and management, mining and quarry operations, utility infrastructure inspection (power lines, gas pipelines, telecommunications), environmental monitoring and data collection, precision scientific surveying; (3) **Defence and Tactical Operations** – autonomous ground vehicles for reconnaissance and surveillance missions, mine detection and clearance operations, checkpoint and perimeter patrol and monitoring, supply line and logistics support,

casualty evacuation and emergency response, hostile-environment exploration and assessment, ordnance disposal and hazardous material platforms, remote sensing and intelligence gathering; (4) **Remote-Operated and Semi-Autonomous Hybrid Control** – teleoperated platforms with autonomous fallback capability during communication loss or latency, human-supervised autonomous systems for inspection and monitoring, tele-operated manipulation platforms with local autonomous station-keeping and stabilization, remotely-commanded convoy and formation systems with inter-vehicle autonomous following, human-in-the-loop semi-autonomous exploration systems with operator override capability.

B. Planning Request

Each planning request specifies: - Start position: the vehicle's current GPS location and heading. - Goal position: a GPS location within the operational area.

The planner is event-driven: it produces a path on demand and republishes if a replan is requested (e.g., after an abort and new goal command). It does not continuously update the path during execution; path tracking is the Motion Planner's responsibility.

C. Map Representation

The map is distributed as a structured data object containing:

Path network graph: a set of named traversable path segments, each defined as a GPS centerline waypoint sequence with junction identifiers at each end. Junctions define connectivity: a junction connects two or more path segment endpoints, forming a navigable graph. Path segments carry no inherent directionality; the planner searches both directions.

Drivable region polygons: convex or concave polygon outlines of open traversable areas. Used by Hybrid A* to determine the free-space boundary.

Static obstacle polygons: known hard obstacles.

Occupancy grid: a binary raster (0 = free, obstacle-value = occupied) of the full operational area, used for footprint-aware collision checking in Hybrid A*.

Cost map: a float32 map (0.0 = free, 1.0 = fully occupied) derived from static obstacles and drivable region boundaries, used for smooth cost gradient computation in Hybrid A*.

Map origin: GPS reference point for the ENU coordinate frame used by all real-time computation.

II. Strategy Selection

A. Geometric Analysis

Before selecting a planning strategy, the planner performs three geometric tests:

Proximity to path network: for both start and goal positions, the planner computes the minimum distance to the nearest structured path segment centerline. If this distance is below a configurable proximity threshold d_{prox} , the position is classified as *near-path*.

Heading alignment: for the start position, the planner computes the angular error between the vehicle's current heading and the nearest path segment's tangent direction. The test accounts for bidirectionality: both the forward and reverse tangent directions are evaluated, and the minimum absolute error is taken.

$$\psi_{err} = \min(|\psi_{veh} - \psi_{path}|, |\psi_{veh} - (\psi_{path} + \pi)|)$$

If $\psi_{err} \leq \psi_{thresh}$,

the vehicle is classified as heading-aligned.

Short-distance check: if the Euclidean distance from start to goal is below a minimum path-routing threshold, structured path routing is bypassed in favor of a direct search.

B. Strategy Assignment Table

Start Position	Heading Alignment	Goal Position	Assigned Strategy
Near path	Aligned	Near path	Strategy 1: Structured graph routing
Near path	Misaligned	Near path	Strategy 3: Mixed (unstructured approach + graph)
Off path	Any	Near path	Strategy 3: Mixed (unstructured approach + graph)
Any	Any	Off path	Strategy 2: Hybrid A* to nearest junction + graph
Short distance	Any	Any	Strategy 2: Pure Hybrid A*

C. The Heading-Alignment Criterion: Why It Matters

The heading-alignment check is the most non-obvious element of strategy selection, yet it prevents a critical failure mode. Consider a vehicle parked at a 90° angle to a bidirectional trail at a junction. The vehicle is physically close to the path (passes the proximity test), but if routed directly onto the graph without a heading correction, the graph search will produce a valid shortest-path route that is geometrically correct but dynamically infeasible for the vehicle's current heading – requiring an instantaneous 90° yaw that an Ackermann vehicle cannot execute.

Without the heading check, the system delivers a path that the Motion Planner will immediately reject or struggle to track, causing degraded behavior or an abort. The heading check reroutes through Strategy 3, which generates a Bézier approach connector that smoothly aligns the vehicle to the path entry point before the structured routing segment begins.

III. Strategy 1: Structured-Path Graph Routing

A. Junction-to-Junction Search

Given a start and goal position both near the path network with adequate heading alignment, the planner:

6. **Maps start and goal to nearest junctions:** for each position, the nearest junction in the graph is identified by Euclidean distance search.
7. **Graph search:** a shortest-path search (Dijkstra or BFS, depending on whether segment weights differ) finds the ordered sequence of junctions from start junction to goal junction. The search is bidirectional-aware: path segments are traversable in both directions.
8. **Waypoint extraction:** for each junction pair in the result, the connecting segment's GPS centerline waypoints are extracted and ordered in the traversal direction.

The result is an ordered list of GPS waypoints from the vehicle's nearest path-entry point to the goal junction.

B. Path Densification

The raw junction-to-junction waypoint list typically has sparse waypoints (every several meters). The planner densifies this to a uniform spacing d_{wp} by linear interpolation along each segment:

$$\mathbf{p}_{interp} = \mathbf{p}_i + t \cdot (\mathbf{p}_{i+1} - \mathbf{p}_i), \quad t = \frac{k \cdot d_{wp}}{d_{seg}}$$

Heading at each interpolated waypoint is assigned as the local tangent direction of the segment.

This dense representation is required by the Motion Planner's nearest-point tracker, which expects waypoints at approximately the motion planner's lookahead resolution.

C. Bézier Entry Connector

If the vehicle is not precisely at the path entry junction — which is the common case for a vehicle parked at an arbitrary position within the path network's proximity zone — a smooth connection from the vehicle's current pose to the first path waypoint is required.

The connector is a **cubic Bézier curve** defined by four control points:

$$\begin{aligned} \mathbf{B}(t) &= (1-t)^3 \mathbf{P}_0 + 3(1-t)^2 t \mathbf{P}_1 + 3(1-t) t^2 \mathbf{P}_2 \\ &\quad + t^3 \mathbf{P}_3 \end{aligned}$$

- $\mathbf{P}_0$: vehicle's current position.
- $\mathbf{P}_1 = \mathbf{P}_0 + \alpha \cdot \hat{h}_{veh}$: tangent extension along vehicle heading.
- $\mathbf{P}_2 = \mathbf{P}_3 - \alpha \cdot \hat{h}_{path}$: tangent extension back along path entry heading.
- $\mathbf{P}_3$: path entry point (first waypoint of structured route).

The offset magnitude $\alpha = \min(0.4 \cdot L_{wb}, 0.3 \cdot d)$ scales with both the vehicle wheelbase L_{wb} (ensuring the curve is proportionate to the vehicle's turning capability) and the connection distance d (preventing overshoot when the vehicle is already close to the path entry). The factor 0.4 on wheelbase was derived empirically to produce curves that respect the vehicle's minimum turning radius for typical approach angles.

The connector is sampled at the same d_{wp} spacing as the main path and prepended to the structured route.

SDF validation: before the connector is accepted, each sampled connector waypoint is checked against the occupancy cost map. If any point falls within a configured clearance margin of an obstacle, the connector is rejected and the system falls back to Strategy 3 (Hybrid A* approach). This ensures the Bézier connector cannot propose a path through an obstacle that is close to the path entry.

D. Robustness Characterization

Strategy 1 has the following robustness properties:

- **Map completeness:** if the path network graph is connected (all junctions reachable from all others), the search will always find a route. The planner validates connectivity at startup and logs disconnected subgraphs.
- **Optimality:** Dijkstra's algorithm produces the globally optimal path within the graph topology (minimum total segment length). This is the only strategy that provides global optimality.
- **Determinism:** identical map and request inputs produce identical outputs on every call.
- **Computational bound:** for a graph with V junctions and E segments, Dijkstra's algorithm runs in $O((V + E)\log V)$ – sub-millisecond for typical operational area graph sizes ($V < 100$).

IV. Strategy 3: Mixed Strategy – Unstructured Approach and Structured Routing

A. Motivation

Strategy 3 is the most frequently invoked strategy for real-world deployments. Vehicles are parked at arbitrary positions and headings after manual operation; they are often not at path junctions and often face away from the nearest path entry. Strategy 1 would produce a geometrically infeasible entry; Strategy 2 (pure Hybrid A*) would work but is slower and produces suboptimal routes on the structured portion. Strategy 3 bridges them.

B. Approach Sub-Plan

The approach sub-plan bridges the vehicle's current pose to the nearest structured path entry point using one of two methods, selected by the approach geometry:

Bézier connector: used when the approach distance is small and the heading misalignment is moderate (misalignment $> 45^\circ$ but $< 120^\circ$, distance $<$ configurable threshold). The connector generation is identical to Section IV.C, with the same SDF validation.

Short Hybrid A* sub-plan: used when the approach distance is large, heading misalignment exceeds 120° , or the Bézier connector fails SDF validation. The Hybrid A* sub-plan (Section VI) is run from the vehicle's current pose to the nearest path entry junction. The resulting sub-path is prepended to the structured route.

C. Structured Route Completion

After the approach sub-plan places the vehicle at the nearest path entry junction, the structured routing (Section IV.A-IV.B) completes the route from that junction to the goal. This produces a hybrid path: an unstructured first segment that handles the vehicle's real-world start configuration, followed by an efficient structured segment that exploits the path network.

D. Why Not Always Use Hybrid A*?

A natural question is why Strategy 3 does not simply run Hybrid A* for the full route from vehicle to goal. There are three reasons:

9. **Computational cost:** Hybrid A* over a large operational area with high resolution is expensive. Running it only for the approach sub-plan (a bounded short distance from the vehicle to the nearest path entry) keeps the computation affordable.
10. **Path quality:** Hybrid A* produces locally optimal paths within the discretized state space, but these paths are not globally optimal across long structured corridors. The structured routing segment, by definition, follows the preferred traversable path – which is preferred precisely because it avoids obstacles, has known surface quality, and is the intended travel corridor.

11. **Predictability:** operators and supervisors in real deployments expect the vehicle to follow the known trail network, not to cut across open terrain in ways that may be geometrically shorter but operationally unexpected.

V. Strategy 2: Hybrid A*

A. Hybrid A* Overview and Motivation

Hybrid A* is an anytime graph-search algorithm that extends classical grid-based A* to handle kinematic constraints (in this case, Ackermann steering dynamics) directly during path generation. The key innovation is the substitution of 4-connected (or 8-connected) grid successors with **kinematically feasible trajectory rollouts** – each successor is generated by forward-simulating the vehicle dynamics over a fixed arc length, ensuring that every path produced is, by construction, executable by the actual vehicle.

This is fundamentally different from classical A* combined with post-hoc smoothing: it avoids the risk of “geometrically valid but kinematically infeasible” paths that smoothing cannot always fix without introducing new collisions near path corners.

B. State Space and Discretization

The Hybrid A* search operates on a discretized 3D state space:

$$\mathbf{s} = (x_{grid}, y_{grid}, \psi_{bin})$$

where:

- (x_{grid}, y_{grid}) are grid cell indices in the occupancy map, with resolution r_g . This discretization reduces the infinite 2D position space to a discrete set of cells, enabling efficient duplicate detection in the open set.
- ψ_{bin} is the discretized heading bin, with a configurable number of bins. Heading discretization balances search granularity against the combinatorial state-space explosion: too few bins miss important orientation changes; too many bins create excessive branching.

Continuous representation: each search node maintains the continuous (x, y, ψ, δ) state (where δ is the instantaneous steering angle in radians), which is needed for accurate kinematic propagation. The discretized grid state is used only for duplicate detection and open-set management.

C. Candidate Trajectory Generation

Candidate trajectory generation is the core operational difference between Hybrid A* and classical A*. From a current state node $\mathbf{s}_{curr} = (x, y, \psi, \delta)$, successor states are generated by discretely sampling steering angle rates within the vehicle’s capability and forward-simulating each candidate over a fixed planning horizon.

C.1 Steering Angle Rate Discretization

The planner generates steering angle rates (in rad/s) over a grid:

$$\dot{\delta}^{(k)} \in \{-\dot{\delta}_{max}, -\dot{\delta}_{max} + \Delta\dot{\delta}, \dots, 0, \dots, +\dot{\delta}_{max}\}$$

where $\dot{\delta}_{max}$ is the maximum steering rate and $\Delta\dot{\delta}$ is the step size. This produces $N_s = 2\lceil\dot{\delta}_{max}/\Delta\dot{\delta}\rceil + 1 = 6$ steering rate candidates per node expansion.

Rationale for rate-based discretization:

steering angle rate is the actionable control input for the vehicle’s steering servo; discretizing rates (rather than absolute angles) ensures that candidates respect the rate limits on the steering actuator. A candidate that requires an instantaneous change in steering angle is physically impossible and will be rejected by the vehicle control layer.

C.2 Kinematic Propagation via Ackermann Model

Each steering rate candidate $\dot{\delta}^{(k)}$ is propagated forward over a fixed arc length l_{arc} under constant velocity v_{plan} using the Ackermann kinematic model:

$$x' = x + v_{plan} \cos(\psi) \Delta t_i$$

$$y' = y + v_{plan} \sin(\psi) \Delta t_i$$

$$\psi' = \psi + \frac{v_{plan}}{L_{wb}} \tan(\delta) \Delta t_i$$

$$\delta' = \delta + \dot{\delta} \Delta t_i$$

where the trajectory is numerically integrated over small time steps Δt_i . The integration is performed iteratively; at each time step, the steering angle is updated by the rate, then clamped to $[-\delta_{max}, +\delta_{max}]$, ensuring no candidate violates steering angle bounds.

The total propagation horizon is $t_{horizon} = l_{arc}/v_{plan}$, which corresponds to approximately integration steps at Δt .

Validation of trajectory feasibility: during integration, the steering angle is clamped. This ensures that no successor state violates the vehicle's steering angle limit – a hard kinematic constraint. If the requested steering rate $\dot{\delta}^{(k)}$ would drive δ outside bounds, the clamping takes effect automatically, and the integration continues. This is a design choice: the path may be suboptimal (limited steering authority), but is always executable.

C.3 Collision Checking During Propagation

A critical implementation detail is **when** collision checking occurs. In this implementation:

1. **Endpoint collision check:** after the full arc-length propagation, the resulting state (x', y', ψ') is checked for collision using vehicle footprint validation.
2. **Footprint representation:** the vehicle is approximated as an axis-aligned rectangle in the local vehicle frame, with half-length $l/2$ and half-width $w/2$, corresponding to typical off-highway vehicle dimensions). Nine check points are sampled:
 - Center: $(0,0)$
 - Four corners: $(\pm l/2, \pm w/2)$
 - Four edge midpoints: $(\pm l/2, 0), (0, \pm w/2)$
3. **World-frame transformation:** each check point is rotated by the vehicle's heading ψ and translated to world position (x', y') :

$$\mathbf{p}_{world} = (x', y') + R(\psi)\mathbf{p}_{local}$$

where $R(\psi)$ is the 2D rotation matrix.

4. **Grid collision detection:** each world-frame check point is converted to a grid cell and validated:
 - If out-of-bounds (beyond map extents), the state is rejected.
 - If the occupancy grid cell value exceeds a collision threshold (127 for uint8 OGM), the state is rejected.
 - If using a float32 cost map, the cell cost c_{cell} must be below the collision threshold c_{coll} .

Alternative collision strategies: some implementations check the entire trajectory segment (not just the endpoint) by sampling the path at fine intervals. This implementation uses endpoint checking with a sufficiently small arc length (1.4 m) to balance safety and computation. The Motion Planner, running at higher frequency, provides reactive collision avoidance.

D. Cost Functions

The search operates on two cost components:

D.1 Accumulated Cost (g-cost)

The accumulated cost $g(s)$ represents the total cost-to-reach a state from the start:

$$g(\mathbf{s}_{next}) = g(\mathbf{s}_{curr}) + \Delta c_{move} + w_{obs} \cdot c_{cell}(\mathbf{s}_{next})$$

where:

- Δc_{move} is the Euclidean distance traveled: $\Delta c_{move} = \sqrt{(\Delta x)^2 + (\Delta y)^2}$. This term drives the search toward paths with shorter arc length.
- $c_{cell}(\mathbf{s}_{next})$ is the cost of the cell at the goal state, obtained from the float32 cost map (0.0 = optimal, 1.0 = obstacle). This term penalizes paths that pass through high-cost regions (near obstacles, steep terrain, etc.).
- w_{obs} is the obstacle cost weight. Setting $w_{obs} > 0$ makes the planner prefer paths through low-cost (free) cells over high-cost (near-obstacle) cells. This is useful in scenarios where there are multiple feasible paths and obstacle proximity should be avoided.

Rationale for decoupled move cost and cell cost: the move cost drives optimality (shortest path), while the cell cost provides a gradient toward safe/preferred regions. By decoupling them, the planner can be tuned to either prioritize distance or safety, or a weighted combination.

D.2 Heuristic Cost (h-cost)

The heuristic cost

Primary: Dubins Curve Heuristic

The **Dubins curve** is the shortest path connecting two poses (position + heading) under a minimum-turning-radius constraint, consisting of at most three arc segments (left turn, straight, right turn or variants). For Ackermann vehicles, the Dubins distance is a tight lower bound on the actual shortest path distance, making it an admissible heuristic.

$$h(\mathbf{s}) = w_d \cdot \text{DubinsDistance}((x, y, \psi), (x_g, y_g, \psi_g), r_{min})$$

where: - $r_{min} = L_{wb}/\tan(\delta_{max})$ is the minimum turning radius. - w_d is the heuristic weight.

The Dubins heuristic provides strong guidance: it considers both the vehicle's position and its heading relative to the goal, which standard Euclidean distance does not.

Fallback: Euclidean Distance Heuristic

If Dubins distance computation fails (rare, but can occur for numerical edge cases), a fallback heuristic is used:

E. Completion Criteria

The Hybrid A* search terminates when one of three acceptance conditions is met:

E.1 Primary Goal: Exact Target Pose

The search accepts a state $\mathbf{s}$ if it falls within a tolerance box around the goal pose:

$$\|(x - x_g, y - y_g)\| < \epsilon_{xy} \quad \text{AND} \quad |\psi - \psi_g| < \epsilon_\psi$$

where: - ϵ_{xy} is the position tolerance. - ϵ_ψ is the heading tolerance.

This is the standard termination for point-to-point navigation.

E.2 Secondary Goal: Merge onto Structured Route

In the mixed strategy (Strategy 3), Hybrid A* is tasked with planning to a structured route line (waypoint sequence) rather than a fixed goal point. A secondary completion criterion allows merging **anywhere along the route** with compatible heading:

$$\min_i \|(x - x_{route,i}, y - y_{route,i})\| < \epsilon_{xy} \quad \text{AND} \quad |\psi - \psi_{route,i}| < \epsilon_\psi$$

where $\psi_{route,i}$ is the tangent direction of the route at waypoint i .

Dynamic heuristic targeting: when planning to a route line, the heuristic $h(\mathbf{s})$ is computed as the Dubins distance to the **closest point on the route**, not a fixed goal. This dynamically updates as the search explores: if a node is closer to a different part of the route, the heuristic re-targets. This keeps exploration focused on the route without requiring a fixed merge point in advance.

Start-distance constraint: to prevent pathological merges (accepting a route point far from the start), a secondary goal is only accepted if the distance from the start to the proposed merge point is less than r_{route} .

E.3 Tertiary Goal: Cost-Map-Confirmed Cart/Trail Path

When a cost map is available (float32 representation of preferred routes), an additional acceptance criterion uses cost-map evidence:

$$c_{cell}(\mathbf{s}) < c_{path} \quad \text{AND} \quad |\psi - \psi_{route}| < \epsilon_\psi$$

where c_{path} is a threshold cost value (, indicating low-cost cells that correspond to the cart path or preferred route in the cost map).

This criterion allows merging onto a route even if the geometric waypoint tolerance is not met, provided the vehicle has reached a low-cost cell in the cost map with compatible heading. This is useful for handling imprecise route waypoint data or gradual entry ramps.

Practical implication: if the structured route is represented in the cost map as a band of low-cost cells (rather than a single centerline), the planner can terminate as soon as it reaches the edge of that band with correct heading, without requiring exact waypoint alignment.

F. Computational Analysis

F.1 Search Complexity

For a bounding region of side length A (e.g., 40 m) at grid resolution $r_g = 1.0$ m, the number of grid cells is $(A/r_g)^2 = 1600$ cells. With $N_b = 24$ heading bins, the theoretical maximum state space is $1600 \times 24 = 38,400$ states.

However, the actual number of expanded nodes is typically much smaller:

- **Heuristic pruning:** the Dubins heuristic is tight and biases the search away from high-f-cost regions.
- **Collision pruning:** many successor candidates hit obstacles and are immediately rejected.
- **Goal proximity:** once the search approaches the goal, many nodes satisfy the termination criteria.

Empirical measurements over typical 20 m approach distances show **~500-1000 node expansions**, with a total planning time of 20-50 ms on commodity CPU hardware (ARM Cortex-A72, circa 2020).

F.2 Branching Factor and Node Generation

Each node generates $N_s = 6$ successor candidates (from steering rate discretization). With collision checking and heuristic-driven pruning, typically 3-5 successors per expanded node are added to the open set. This gives an effective branching factor of 3-5, much lower than an unguided graph search.

F.3 Memory Requirements

The open set (priority queue) typically holds 100-500 nodes concurrently. The closed set (visited map) is an unordered hash map with

G. How This Implementation Differs from Classical Hybrid A*

The reference formulation of Hybrid A* (Dolgov et al., 2008) was optimized for highway driving (smooth, straight-biased paths with few obstacles). This implementation introduces several modifications for off-road and tactical environments:

Aspect	Classical Hybrid A*	This Implementation	Rationale
Steering discretization	Fixed steering angles (left, straight, right)	Steering angle rates with fine discretization	Rates align with vehicle control inputs; finer grid enables smoother paths in tight spaces
Cost map integration	Obstacle avoidance only (binary)	Gradient-based obstacle proximity penalty	Soft cost penalties steer around obstacles rather than treating them as hard walls; enables multi-objective optimization
Route merging (secondary goal)	Not applicable; fixed single-goal search	Dynamic route-to-line termination with heading alignment	Mixed-strategy composition requires seamless integration with structured paths; heading alignment prevents skew-entry

Aspect	Classical Hybrid A*	This Implementation	Rationale
Heuristic choice	Euclidean or simple distance	Dubins curve primary + Euclidean fallback	Dubins distance is tighter (lower bound closer to true cost) and heading-aware; critical for off-road navigation where turning radius is significant
Collision checking	Continuous footprint checks along trajectory	Endpoint footprint check with fine arc length	Reduces computation; the fine arc length (1.4 m) ensures safety while keeping planning time under 50 ms
Multi-goal termination	Not applicable	Three-tier completion criteria (exact pose, route merge, cost-map confirmation)	Enables flexible deployment in structured and unstructured domains without algorithm modification
Velocity profile in planning	Fixed constant velocity	Constant velocity for planning; smooth profile assigned post-hoc	Separates global topology (velocity-agnostic) from motion execution (velocity-aware), improving reusability across vehicle types and missions

H. Robustness and Failure Modes

Strengths: - **Kinematic feasibility:** every path is guaranteed to be drivable. - **Deterministic behavior:** identical inputs produce identical outputs (search is deterministic with a priority queue and fixed discretization). - **Anytime convergence:** early solutions are available; quality improves with time. - **Bounded computation:** hard timeout ensures real-time operation.

Failure modes and mitigations: - **No path exists:** if obstacles block all paths from start to goal, the planner reports infeasibility.

Fallback to human teleoperation or static replanning (waiting for obstacle clearance). - **Start state in collision:** if the vehicle's current pose is in collision (e.g., after an accident), the planner immediately reports infeasibility. The Motion Planner (not global planner) handles collision escape. - **Timeout before convergence:** if 5 s elapses without finding a path, the search halts. Partial solution (best node found so far) is not returned; instead, infeasibility is reported. This conservative approach prioritizes safety over latency. - **Heuristic failure (Dubins computation):** if Dubins distance computation encounters numerical issues, the fallback Euclidean heuristic is used. Search proceeds but may be slower.

VI. Velocity Profile Assignment

A. Curvature-Based Speed Limits

After path generation, each waypoint is assigned a maximum longitudinal speed based on the local path curvature $\kappa(s)$ estimated over a lookahead window:

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

with $\alpha = 15$ (empirically tuned), $v_{max} = 3.0$ m/s, $v_{min} = 0.5$ m/s. At zero curvature (straight path), $v_i = v_{max}$. At high curvature (tight bend, $\kappa = 0.3$ rad/m), $v_i \approx v_{max}/5.5 \approx 0.5$ m/s.

The curvature is estimated at each waypoint as:

$$\kappa_i = \frac{|\Delta\psi_i|}{\Delta s_i}$$

where $\Delta\psi_i$ is the heading change across a segment of arc length Δs_i centered at waypoint i . A configurable lookahead window averages over multiple waypoints to reduce sensitivity to local path noise.

B. Kinematic Speed Smoothing

The curvature-assigned speeds may violate vehicle acceleration limits: two adjacent waypoints with significantly different curvature assignments could produce a required acceleration exceeding the

physical limits. A forward-backward smoothing pass enforces maximum acceleration a_{max} and maximum deceleration a_{min} :

Forward pass: $v_{i+1} =$

$$\min(v_{i+1}^{curv}, \sqrt{v_i^2 + 2a_{max}\Delta s_i})$$

Backward pass: $v_i =$

$$\min(v_i^{forward}, \sqrt{v_{i+1}^2 + 2|a_{min}|\Delta s_i})$$

The result is a speed profile that is simultaneously constrained by curvature (upper bound) and kinematically smooth (acceleration limits). This profile is used directly by the Motion Planner's lookahead tracker as the target speed at each waypoint.

C. Why Assign Speed at the Global Planner Level

Assigning speed at the global planner level — rather than leaving it entirely to the Motion Planner — provides two benefits:

1. **Preview:** the Motion Planner can begin decelerating for an upcoming tight bend several seconds before reaching it, because the global path's speed annotation signals the upcoming speed limit. Without this annotation, the Motion Planner only sees the bend when it appears in its lookahead window.
2. **Consistency:** multiple Motion Planner instances (in a multi-vehicle fleet) receiving the same global path will apply identical speed limits, producing predictable fleet-level behavior.

VII. Path Trimming and Validity

A. Closest-Point Trim

When the global path is published, it may contain waypoints behind the vehicle's current position (due to the vehicle having moved between the planning request and the path receipt). These must be removed to prevent the Motion Planner's nearest-point tracker from engaging with already-passed segments.

The planner computes the waypoint index i^* nearest to the vehicle's current ENU position and trims all waypoints with index $< i^*$. The published path begins at i^* .

B. Validity Criteria

A path is published with $is_valid = true$ only if all of the following hold:

- At least two waypoints exist after trimming.
- No waypoint lies within the obstacle clearance distance of a static obstacle.
- The Bézier connector (if generated) passed SDF validation.
- The final waypoint is within a configurable goal-arrival tolerance of the requested goal position.

VIII. Robustness Analysis

A. Handling All Vehicle Start Configurations

The three-strategy selection guarantees that all vehicle start configurations are handled without failure:

Configuration	Strategy	Mechanism
Perfectly aligned on path	1	Direct graph routing
Slightly offset, good heading	1 + Bézier connector	Short connector prepended
Moderate misalignment	3 + Bézier approach	Bézier corrects heading before entry
Large misalignment or far off-path	3 + Hybrid A* approach	Hybrid A* generates feasible approach
Very short distance	2	Pure Hybrid A*
Goal off-path	2 or 3	Hybrid A* to junction, then graph

The only failure case is a disconnected path network graph (goal unreachable via any junction chain). This is detected at startup, logged, and results in a clean infeasibility report to the Mission Manager.

B. Sensitivity Analysis

Map noise: the path network graph is GPS-anchored. GPS accuracy in the target environment is typically 0.5-2.0 m CEP. The proximity threshold d_{prox} is set above this value (typically 5-10 m), ensuring that GPS uncertainty does not cause a near-path vehicle to be misclassified as off-path.

Curvature estimation noise: the curvature estimate is smoothed over a configurable window. Noisy curvature from GPS-anchored path data produces over-conservative speed limits (more braking than necessary), never under-conservative (speeding through a bend). This asymmetry is intentional: false positives in braking are safe; false negatives are not.

Bézier SDF validation threshold: the clearance margin used for Bézier validation is set conservatively (larger than the vehicle half-width). False positives (connector rejected when it is actually safe) cause a fallback to Hybrid A*, which succeeds. False negatives (connector accepted when it clips an obstacle) are prevented by the conservative margin.

C. Multi-Level Fallback Architecture

The three-strategy planner is, at its core, a **safety architecture** as much as a planning architecture. Every strategy selection decision carries an implicit fallback: if the preferred – most computationally efficient – strategy is geometrically infeasible or fails a validation gate, a more general strategy is attempted before infeasibility is reported to the Mission Manager. This layered structure ensures that no single component failure propagates directly to a mission abort or, worse, to an invalid path reaching execution.

The complete fallback chain operates as follows:

Level 1 – Bézier Connector (Preferred):

When Strategy 1 is selected, a Bézier entry connector is first attempted to bridge the vehicle's current pose to the path entry point. This is the computationally lightest option and succeeds for the majority of near-path, heading-aligned starts.

Level 2 – SDF Connector Validation Gate:

Before the Bézier connector is accepted, every sampled connector waypoint is checked against the static occupancy cost map for obstacle clearance. A connector that clips any mapped obstacle within the clearance margin is rejected. This gate is hard: no obstacle-penetrating connector is ever forwarded to execution regardless of how close to feasibility it comes.

Level 3 – Hybrid A* Approach Sub-plan: If the Bézier connector fails SDF validation, or if the heading misalignment or approach distance exceeds the Bézier applicability range, the system invokes a short Hybrid A* sub-plan as the approach segment. This sub-plan is computationally bounded to the approach region and inherits the full kinematic constraint set of the global Hybrid A* search, providing completeness guarantees for the approach segment.

Level 4 – Path Publication Validity Gates: Before any path is forwarded to the Motion Planner, all validity criteria must pass (Section VIII.B): minimum waypoint count after trimming, per-waypoint obstacle clearance, successful SDF validation of all connector segments, and goal arrival tolerance. A path failing any criterion is not published; the Mission Manager receives an infeasibility status and transitions the vehicle to a safe stationary state.

Level 5 – Structured Infeasibility Report: If all planning strategies are attempted and no valid path can be produced, the system publishes a structured infeasibility report specifying: (i) which strategy was attempted, (ii) which fallback levels were invoked, and (iii) which specific criterion caused the final failure. This report is logged for post-mission analysis, forwarded to the Mission Manager for safe-state transition, and surfaced to the operator interface with an actionable description. The vehicle remains stationary in its last safe pose.

The critical safety property of this chain is **monotonicity under fallback**: each fallback level is strictly more general than the level it replaces (Bézier → Hybrid A* approach → full Hybrid A*), and each level applies the same safety validation gates. The system can only reach the infeasibility report if all strategies are genuinely infeasible – not merely if the preferred strategy was computationally convenient.

D. Safety Validation Gates

Three independent validation gates operate across all strategies, enforcing the invariant that no kinematically infeasible or obstacle-penetrating path is ever forwarded to execution:

Gate 1 – SDF Clearance Check (Per-Waypoint): Every path waypoint produced by any strategy – graph routing waypoints, Bézier connector samples, and Hybrid A* successor nodes – is checked against the static occupancy cost map for obstacle clearance at a margin exceeding the vehicle’s half-width. This gate operates during path generation, not as a post-processing step, so infeasible intermediate results are detected before they influence the assembled output.

Gate 2 – Kinematic Curvature Validation (Per-Segment): Paths produced by graph routing and Bézier connectors are checked for maximum implied curvature at each segment. A curvature violation – consecutive waypoints implying a turning radius below the vehicle’s minimum – causes the offending connector to be rejected and the fallback chain to advance to the next level. Hybrid A* paths are exempt from this gate because kinematic feasibility is guaranteed by construction through Ackermann forward simulation.

Gate 3 – Complete Path Validity Check (Pre-Publication): The final assembled path is validated as a whole before publication (Section VIII.B). This gate aggregates the results of all prior checks and adds the minimum waypoint count and goal arrival tolerance conditions. It acts as the last line of defense: even if a per-waypoint or per-segment check were somehow bypassed, this gate prevents publication of an invalid complete path.

The three gates are architecturally independent: they operate at different granularities (waypoint, segment, complete path), use different data sources (occupancy cost map, curvature computation, assembled path structure), and are applied at different pipeline stages. An error or bypass in one gate does not disable the others.

E. Alignment with Functional Safety Standards

The design principles of the multi-strategy planner align with the requirements of automotive and autonomous system functional safety frameworks – a prerequisite for deployment in safety-assured operational environments:

ISO 26262 (Road Vehicles – Functional Safety): The explicit fallback chain (Section IX.C) implements a defense-in-depth architecture consistent with ISO 26262 Part 4 (product development at the system level). The structured infeasibility reporting and clean safe-state abort behavior align with safe-state requirements for ASIL-B and ASIL-C system elements. The deterministic, reproducible planning output enables the repeatable test coverage required for safety case construction.

ISO 21448 (SOTIF – Safety of the Intended Functionality): The heading-alignment check and approach sub-plan directly address a SOTIF trigger scenario: a purely graph-based planner that receives a vehicle with severe heading misalignment performs its intended function (graph routing) correctly, but produces an output that is kinematically infeasible for the actual vehicle configuration – a SOTIF failure mode where the system operates within specification yet produces a dangerous result. The heading check detects this condition and reroutes to a strategy that produces a geometrically and kinematically correct output, eliminating the trigger.

ISO/PAS 8800 (AI and Machine Learning in Road Vehicles, emerging): The hybrid AI-perception + classical-planning architecture described in Section X.B separates the AI component (perception and map generation, where statistically approximated outputs are tolerable after validation) from the safety-critical planning component (route generation, where deterministic and kinematically correct outputs are mandatory). This separation is consistent with the principle of ASIL decomposition: the AI perceptual component can be qualified at a lower integrity level than the classical planning component that generates the executable path.

Determinism as a Certification Pre-Condition:

A planning component that produces different routes for identical inputs cannot be exhaustively tested or formally verified. Non-determinism introduces a combinatorial explosion of possible behaviors that cannot be covered by a finite test suite. The multi-strategy planner's strict determinism (identical map + identical request → identical output, every time) is not merely a convenience property; it is a structural pre-condition for constructing a valid safety case under any applicable functional safety standard.

IX. Relevance in the AI Era

A. The Current State of AI Path Planning

Learned path planning approaches – including neural network path prediction from overhead imagery [A1], diffusion-model trajectory generation [A2], and transformer-based route planners [A3] – have demonstrated impressive generalization across diverse environments. They can implicitly encode terrain traversability, path network topology, and operator preferences without explicit map representations.

However, as of 2026, these approaches share common limitations in safety-critical deployments:

Property	Classical Multi-Strategy Planner	Learned Planner
Kinematic feasibility guarantee	Yes (by construction)	No (requires post-validation)
Deterministic output	Yes	No (stochastic sampling)
Computational bound	Yes (graph search $O((V+E)\log V)$)	No (variable inference time)
Interpretable failure mode	Yes (disconnected graph, etc.)	No
Map independence	No (requires path network map)	Partial (can use raw imagery)
Generalization to novel environments	Limited	Strong
Certification path (ISO 26262 / similar)	Established	Not yet established

The multi-strategy planner is weaker where learned planners are stronger (generalization, map independence) and stronger where learned planners are weaker (kinematic guarantees, determinism, certification). This suggests a **complementary** rather than competitive relationship.

B. The Hybrid Architecture: AI Perception + Classical Planning

The most practically useful near-term architecture combines AI-based terrain perception (traversability estimation, obstacle detection, trail detection from imagery) with classical planning that consumes this perception output. Concretely:

- A semantic segmentation network classifies image or LiDAR data as trail, open terrain, or obstacle.
- The output updates the occupancy grid and cost map used by the multi-strategy planner.
- The planner uses this AI-updated map to generate kinematically feasible, globally optimal routes.

C. Where Classical Planning Remains Irreplaceable

Three specific properties of classical path planning cannot currently be replicated by learned approaches:

1. Kinematic correctness by construction.

Hybrid A* generates paths through Ackermann forward simulation; every path point is, by construction, reachable from the previous one with the vehicle's kinematic model. Learned planners do not share this property: a learned trajectory may require instantaneous velocity changes that are physically impossible. Post-hoc kinematic validation of a learned path adds latency and complexity.

2. Formal completeness guarantees. If a path exists within the planner's state space, Hybrid A* will find it (given sufficient node expansions). Learned planners have no such completeness guarantee; they may fail to find a valid path in cases where one clearly exists because the training distribution did not include that configuration.

3. Graceful failure with interpretable cause.

When the classical planner fails, it reports a specific cause (no path to junction, Bézier validation failed, node expansion limit reached). This cause is actionable: a map engineer can correct a disconnected graph; an operator can reposition the vehicle. Learned planner failures produce no such actionable information.

D. Long-Term Evolution

A productive long-term evolution is:

- **Immediate term:** retain the full classical multi-strategy planner; upgrade the occupancy and cost maps using AI-based perception.
- **Medium term:** use a learned planner as a parallel candidate generator; select between learned and classical outputs based on a shared kinematic validation and SDF feasibility check.
- **Long term:** if a learned planner can be shown to produce kinematically correct, SDF-feasible, globally optimal paths with bounded computation across all configurations in the operational envelope, it may replace the classical planner. The kinematic validation step and SDF feasibility check remain regardless of which planner produced the candidate.

The key insight is that **kinematic feasibility validation is not optional** regardless of how the path was generated. The classical planner embeds this validation in its generation step; any AI planner requires it as a post-processing step.

E. State of the Art in AI Planning (2024-2026)

The 2024-2026 period has produced a generation of AI planning systems substantially more capable than the foundational approaches referenced in Section X.A. Three convergent developments have driven this progress: the scaling of vision-

language models to autonomous driving domains, the maturation of diffusion-based generative models for trajectory prediction, and the emergence of end-to-end differentiable neural driving architectures trained on large-scale real-world datasets.

Representative systems relevant for comparison with the multi-strategy planner include:

End-to-End Neural Driving Architectures:

UniAD [A4] demonstrated that a unified transformer architecture performing joint perception, prediction, and planning in a single model can achieve competitive trajectory quality on structured benchmarks such as nuScenes. SparseDrive and related vectorized-scene architectures [A5] reduce computational cost over dense raster approaches while maintaining trajectory quality. These systems demonstrate that end-to-end differentiable planning is practically feasible, but they produce trajectories that require post-hoc kinematic validation before execution — a step that the multi-strategy planner performs natively.

Vision-Language Model (VLM) Planners:

DriveVLM and related approaches [A6] apply large vision-language models fine-tuned on driving scenarios to generate natural-language route rationales alongside trajectory outputs. These architectures improve interpretability compared to purely neural planners. However, their outputs remain stochastic, and a natural-language rationale is not a substitute for formal kinematic or SDF feasibility validation.

World Model-Based Planners: GAIA-1 and

DriveDreamer [A7] use video diffusion models to predict photorealistic future scene states conditioned on planned actions, enabling planning through imagined futures. These systems represent the frontier of model-based neural planning but require substantial computational resources and exhibit distribution shift degradation in environments outside their training domain — exactly the novel off-road settings targeted by this planner.

Diffusion-Based Trajectory Planners:

MotionDiffuser [A8] and DiffTraj [A9] apply score-matching diffusion processes to sample diverse, physically plausible trajectory distributions. These models handle multi-modal uncertainty elegantly and produce smooth outputs. However, their stochastic sampling is fundamentally incompatible with the deterministic safety assurance requirements that the multi-strategy planner satisfies.

Each of these systems represents a genuine capability advance over earlier learned planners. The critical observation is that not one has eliminated the need for a deterministic kinematic validation and SDF feasibility layer before path execution – a function performed natively and with hard guarantees by the multi-strategy classical planner.

F. Foundation Models and Large Language Model-Based Planning

The application of large language models (LLMs) and vision-language foundation models to autonomous navigation has advanced substantially by 2026. Models pre-trained on large multimodal corpora have demonstrated emergent capabilities in route reasoning: identifying semantically meaningful waypoints, reasoning about terrain type from imagery, decomposing multi-step navigation tasks from natural-language mission descriptions, and applying common-sense knowledge to unusual obstacle interpretation.

These capabilities offer genuinely new value that classical geometric planning cannot provide:

1. **Common-sense obstacle interpretation:** an LLM-based system can reason that a visible trail obstruction requires a detour even if the obstacle has not yet been added to the operational map, using perceptual context that a geometric planner cannot access.
2. **Natural-language mission specification:** operators can express goals and constraints conversationally, which the LLM translates into structured planning parameters — reducing the barrier to mission configuration.

3. **Contextual priority adaptation:** mission urgency, cargo sensitivity, and terrain risk preference can be expressed naturally and used to modulate cost weights dynamically without manual parameter adjustment.

However, LLM-based route generation introduces structural challenges that are not solvable by scaling:

- **Non-determinism:** all current LLMs sample from probability distributions; identical inputs produce probabilistically different outputs. This is structurally incompatible with deterministic safety case construction.
- **Kinematic unawareness:** LLMs operate on tokenized representations with no inherent vehicle kinematic model. Generated waypoints have no guarantee of physical reachability by the actual vehicle.
- **Hallucination under distribution shift:** LLMs may generate plausible-sounding but incorrect routes for environments underrepresented in training data — precisely the novel off-road environments where this planner is intended to operate.
- **Inference latency unpredictability:** transformer inference time varies with input length, batching, and hardware load, providing no hard computational bound suitable for real-time embedded systems.

The productive integration is as a **mission decomposition layer** above the classical planner: the LLM translates natural-language mission descriptions into structured planning queries (goal positions, cost weight adjustments, constraint specifications), which the classical planner then executes with full kinematic and safety guarantees. In this architecture the LLM's expressive power is preserved while the safety-critical execution path remains entirely within the deterministic classical planner.

G. World Models, Diffusion Planners, and the Limits of Statistical Feasibility

World models and diffusion-based planners share a characteristic that distinguishes them from both classical planners and earlier neural approaches: they represent distributions over future states or trajectories rather than producing single deterministic outputs. This distribution-based representation captures multi-modal futures, enables uncertainty quantification, and produces smooth trajectories that reflect the statistics of real-world vehicle motion.

However, they share a fundamental limitation that is structural rather than a matter of scale or training data: **statistical plausibility is not equivalent to physical feasibility**. A trajectory sampled from a well-trained diffusion model is likely to be kinematically feasible — it will be feasible for most samples, in most configurations, most of the time. For safety-critical deployment, “most of the time” is not an acceptable guarantee. The requirement is a hard guarantee that no infeasible trajectory is ever forwarded to execution.

The classical multi-strategy planner provides this hard guarantee by construction. Hybrid A* generates every successor via explicit Ackermann forward simulation; no path point can violate kinematic feasibility because kinematic feasibility is the mechanism by which path points are generated. This property cannot be approximated statistically; it is either present by construction or absent.

Diffusion planners and world model planners can in practice produce high-quality trajectories that satisfy kinematic constraints — but they require a separate kinematic projection or validation step to guarantee this for all samples. That projection step, if implemented rigorously, reintroduces classical kinematic constraint enforcement at the output boundary — functionally equivalent to the Hybrid A* correction the multi-strategy planner already performs natively.

autonomous vehicle systems with power and size constraints. A world model that fails to generate a valid future for a given start configuration also provides no useful guidance — it degrades silently rather than reporting a structured infeasibility cause.

H. The Persistent Safety Gap: Certified Deployment vs. Research Demonstration

The most important distinction between the multi-strategy classical planner and any current AI-based planning system is not capability — it is the gap between research benchmark performance and safety-certified deployment in a real operational system.

As of 2026, no AI path or trajectory planner has received functional safety certification at ASIL-B or above for production deployment in a safety-critical autonomous vehicle system. This gap reflects structural requirements of safety certification frameworks that are not addressable by improved model performance:

Formal worst-case behavior bounds. ISO 26262 requires that safety-critical software components have demonstrated worst-case timing bounds and failure mode catalogues. An AI model with variable inference time and stochastic output has no worst-case behavioral bound in the required sense, and its failure modes are not enumerable from first principles.

Reproducible test coverage. A deterministic system can be exhaustively tested against a finite set of representative inputs and produce verified, repeatable outputs. A stochastic system requires statistical coverage arguments that are more complex to certify and structurally cannot exclude tail-distribution failures.

Explainable failure attribution. When a certified system fails, the cause must be traceable to a specific component, input condition, and state transition. Neural model failures are attributed to high-dimensional weight configurations that are not directly interpretable or actionable by a field engineer or safety analyst.

Adversarial robustness. AI-based planners are potentially vulnerable to adversarial perturbations: subtle sensor noise patterns or map corruptions that cause the model to produce grossly incorrect outputs while appearing confident. Classical planners based on geometric analysis and graph search are structurally immune to this class of failure – there is no differentiable model to be adversarial perturbed through gradient-based attacks.

The multi-strategy planner’s determinism, explicit fallback chain, per-level failure attribution, and bounded runtime are not incidental design choices; they are the properties that make the planner certifiable under functional safety standards. Any AI planning system aspiring to safety certification must either acquire these properties natively or demonstrate equivalence through an alternative formal argument – which, as of 2026, none has accomplished.

Extended Comparison Table: Multi-Strategy Classical Planner vs. 2024–2026 AI Planning Systems

Property	Multi-Strategy Classical Planner	UniAD / End-to-End Neural [A4]	SparseDrive [A5]	DriveVLM / LLM Planner [A6]	Diffusion Planner [A8,A9]	World Model Planner [A7]
Kinematic feasibility guarantee	Yes (by construction)	No – post-validation required	No – post-validation required	No – post-validation required	No – projection required	No – post-validation required
Deterministic output	Yes	No	No	No	No	No
Hard computational bound	Yes – $O((V+E)\log V)$ + bounded A^*	Fixed inference; variable path quality	Fixed inference; variable quality	Variable – LLM dependent	Variable – diffusion steps	Very high – video generation
Multi-level safety fallback	Yes (5 levels)	None built-in	None built-in	None built-in	None built-in	None built-in
Interpretable failure mode	Yes (per-level structured report)	No	No	Partial (natural language only)	No	No
Certifiable at ASIL-B+	Yes (established path)	No (as of 2026)	No (as of 2026)	No (as of 2026)	No (as of 2026)	No (as of 2026)
Adversarial robustness	High (geometric, model-free)	Low	Low	Low-Moderate	Low	Low
Generalization to novel environments	Limited (map-dependent)	Strong	Strong	Strong	Moderate	Moderate
Embedded system viability (SWaP-C)	Yes	Moderate	Moderate	No (cloud inference)	Moderate	No
Requires external validation layer	No (built-in)	Yes	Yes	Yes	Yes	Yes
Structured corridor compliance	Preferred by design	Not guaranteed	Not guaranteed	Not guaranteed	Not guaranteed	Not guaranteed

The pattern across all AI approaches is consistent: every system requires an external validation layer that performs exactly the safety functions the multi-strategy classical planner performs natively. This is not a temporary limitation awaiting a larger model; it reflects the structural incompatibility between statistical learning and hard safety guarantees.

X. Advantages Over End-to-End and General Global Planning Approaches

A. Advantages Over End-to-End Route Generation

End-to-end route generation typically maps sensor/map context directly to a path without explicit decomposition into strategy selection, graph routing, and search fallback. This can improve nominal adaptability but introduces uncertainty in guarantees needed for field reliability.

The proposed multi-strategy planner improves on end-to-end approaches through:

1. **Deterministic route generation:** identical inputs produce identical routes.
2. **Kinematic validity by construction:** Hybrid A* and connector validation ensure drivable outputs.
3. **Interpretable failure modes:** infeasibility has explicit, actionable causes.
4. **Bounded runtime behavior:** graph routing and bounded search provide predictable latency.
5. **Operator-trust alignment:** route preference for known structured corridors improves operational predictability.

Table IV: Multi-Strategy Planner vs. End-to-End Planning

Criterion	End-to-End Route Generation	Multi-Strategy Global Planner (This Work)
Determinism	Low	High
Kinematic guarantees	Post-hoc or absent	Built-in
Failure interpretability	Low	High
Runtime predictability	Variable	Bounded
Structured-corridor compliance	Not guaranteed	Preferred by design
Certification alignment	Limited	Stronger

B. Comparison with Common Global Planning Systems

Table V: Comparison Across Relevant Global Planning Systems

Global Planning System	Strengths	Common Failure Mode	Why the Proposed Multi-Strategy Planner Performs Better
Graph-only routing	Fast, globally optimal on graph	Fails for off-path starts and heading misalignment	Adds unstructured approach generation (Bézier/Hybrid A*) before graph routing
Pure A* on occupancy grid	Complete over discretized space	Produces non-Ackermann-feasible sharp turns	Uses Hybrid A* kinematic state expansion for drivable paths
Pure Hybrid A* (whole route)	Kinematically feasible everywhere	Expensive over large maps, can ignore preferred corridors	Applies Hybrid A* only where needed; uses graph routing for long-haul efficiency
Sampling planners (RRT/RRT*)	Handles complex spaces	Stochastic quality and variable runtime	Deterministic strategy selection with bounded compute and smoother operator-expected routes
End-to-end learned planner	Strong adaptation potential	Weak guarantees, hard-to-explain failures	Keeps deterministic geometric core while still allowing AI map enhancement
Proposed multi-strategy planner	Robust across mixed structured/unstructured starts	Depends on map quality	Best practical blend of reliability, performance, and interpretability

C. Safety Architecture as a First-Class Design Property

Beyond functional planning performance — route quality, computational cost, and generalization capability — the safety architecture of a planner as a deployable system component deserves explicit comparison. The multi-strategy planner treats safety as a first-class design property rather than a post-hoc add-on: every structural decision from strategy selection to path publication is a safety mechanism, not merely an engineering convenience.

Table VI: Safety Architecture Property Comparison

Safety Property	Graph-Only Planner	Pure Hybrid A*	End-to-End AI Planner (any)	Multi-Strategy (This Work)
Kinematic correctness by construction	No	Yes	No	Yes
SDF obstacle clearance guarantee	Partial (path nodes only)	Yes (footprint check)	No (post-hoc)	Yes (3 independent gates)
Multi-level fallback on failure	None	None	None	Yes (5 levels)
Deterministic, reproducible failure modes	Yes	Yes	No	Yes
Safe stationary state on infeasibility	Yes (clean abort)	Yes (clean abort)	Undefined	Yes (structured abort + cause report)
Per-level failure attribution to operator	No	No	No	Yes
External validation layer required	No	No	Yes	No
Certifiable under ISO 26262	Partial	Yes	No (as of 2026)	Yes

The external validation layer required by all current AI planners introduces an architectural vulnerability: the planner and its validator are separate components with separate failure modes and a timing interface between them. A validator that has latency, is bypassed under high-frequency replanning requests, or fails silently under unusual inputs creates a time window during which an invalid path could be forwarded to execution. The multi-strategy planner eliminates this window by embedding validation within the generation process – a path that fails any validation gate is never assembled and forwarded, not merely rejected after the fact.

This difference is significant for safety engineering practice: the correctness of built-in validation is demonstrable through code review and static analysis of the planning pipeline itself. By contrast, the correct operation of a separate validation layer requires additional integration testing, interface formal specification, and an argument that the validator is always reached before path execution – a materially more complex safety case.

CONCLUSION

The multi-strategy global path planner presented in this paper provides a robust, computationally bounded, kinematically correct global route for any vehicle-goal configuration in a mixed structured and unstructured off-road environment. The three-strategy selector ensures that the most efficient planning approach is applied for each configuration, while the Bézier connector and Hybrid A* fallback handle the diverse real-world vehicle start conditions that a purely graph-based approach cannot address.

The comparison with AI-based planning approaches demonstrates that the classical multi-strategy planner and learned planners are complementary: AI excels at perception-driven map generation and novel-environment generalization; the classical planner provides kinematic correctness guarantees, deterministic output, and interpretable failure modes that remain essential for safety-critical deployments. The recommended architecture combines AI perception with classical planning, gaining the benefits of both.

As AI planning matures, the validation interface between AI-generated candidates and kinematic/SDF feasibility checking will remain permanent. The classical planner provides the reference implementation and formal correctness benchmark against which AI planner outputs can be validated.

REFERENCES

- [1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [2] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Practical search techniques in path planning for autonomous driving,” *Proc. AAAI Workshop on Autonomous Driving*, 2008.
- [3] H. Choset *et al.*, *Principles of Robot Motion*. MIT Press, 2005.
- [4] E. Frazzoli, M. A. Dahleh, and E. Feron, “Maneuver-based motion planning for nonlinear systems with symmetries,” *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1077-1091, 2005.
- [5] T. Lozano-Pérez, “Spatial planning: A configuration space approach,” *IEEE Trans. Comput.*, vol. C-32, no. 2, pp. 108-120, 1983.
- [6] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, pp. 269-271, 1959.
- [7] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100-107, 1968.
- [8] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566-580, 1996.
- [9] 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 Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33-55, 2016.
- [A1] R. Shah *et al.*, “ViKiNG: Vision-based kilometer-scale navigation with geographic hints,” *RSS*, 2023.
- [A2] M. Janner *et al.*, “Offline reinforcement learning as one big sequence modeling problem,” *NeurIPS*, 2021.
- [A3] D. Driess *et al.*, “PaLM-E: An embodied multimodal language model,” *ICML*, 2023.
- [A4] Y. Hu *et al.*, “Planning-oriented autonomous driving,” *CVPR*, 2023. (UniAD)
- [A5] H. Sun *et al.*, “SparseDrive: End-to-end autonomous driving via sparse scene representation,” *arXiv preprint arXiv:2405.19620*, 2024.
- [A6] Z. Wang *et al.*, “DriveVLM: The convergence of autonomous driving and large vision-language models,” *CoRL*, 2024.
- [A7] A. Hu *et al.*, “GAIA-1: A generative world model for autonomous driving,” *arXiv preprint arXiv:2309.17080*, 2023.
- [A8] Z. Jiang *et al.*, “MotionDiffuser: Controllable multi-agent motion prediction using diffusion,” *CVPR*, 2023.
- [A9] L. Zhong *et al.*, “DiffTraj: Generating GPS trajectory with diffusion probabilistic model,” *NeurIPS*, 2023.

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

© 2026 Tata Elxsi Limited. All rights reserved.