Hybrid A Based Motion Planning for Autonomous Valet Parking

Mohamed Eljahmi

***Abstract*—This project implements and evaluates an autonomous valet parking system for three nonholonomic vehicle models: a differential-drive robot, an Ackermann-steered car, and a truck-trailer configuration. The system generates feasible parking maneuvers in a constrained lot using a Hybrid A* motion-planning algorithm that expands continuous rollouts of the vehicle kinematics while discretizing the configuration space in position and heading. Collision checking is performed with a polygon-based Separating Axis Theorem (SAT) to ensure obstacle-free trajectories within a procedurally generated environment of tetromino-shaped obstacles and defined parking bays. Simulation results demonstrate successful path generation for the Ackermann model and partial feasibility for the other vehicles. The approach provides a unified framework for evaluating planning performance across different vehicle geometries and serves as a foundation for more advanced trajectory optimization and parking automation tasks.**

Keywords—valet parking, hybrid A* algorithm, configuration space, nonholonomic motion planning, Ackermann kinematics, collision detection, autonomous vehicles

I. Introduction

Autonomous valet parking represents a challenging motion-planning problem involving tight spatial constraints, nonholonomic vehicle dynamics, and safety-critical collision avoidance. Unlike open-road navigation, the parking task requires generating short, precise trajectories that maneuver vehicles into target bays within cluttered environments. Classical grid-based planners such as A* or Dijkstra's algorithm guarantee optimality on discrete maps but neglect vehicle kinematic constraints, often producing infeasible paths when executed by real robots.

To address these limitations, **Hybrid A*** planning extends the A* framework to continuous state spaces by integrating a nonholonomic vehicle model during node expansion. Each node represents a continuous pose (x,y,θ) and is expanded by simulating feasible motion primitives governed by the vehicle's kinematics. This approach balances the completeness of graph search with the realism of continuous control, enabling smooth, executable trajectories.

In this work, Hybrid A* planning is applied to a **valet-parking simulation** that supports three vehicle configurations: a differential-drive robot, an Ackermann-steered car, and a truck–trailer system. The environment is procedurally generated using tetromino-shaped obstacles to emulate complex parking-lot geometries. A polygon-based **Separating Axis Theorem (SAT)** collision checker ensures obstacle-free motion at each rollout step. The resulting framework allows systematic comparison of planners and vehicle models within a reproducible simulation, forming the basis for future extensions in autonomous parking and trajectory optimization.

II. Methodology

A. Vehicle Kinematic Models

Three nonholonomic vehicle configurations are modeled to capture different motion constraints encountered in autonomous parking.

1. The **differential-drive robot** is represented by the unicycle model

   $\dot{x} = v \cos\theta$

   $\dot{y} = v \sin\theta$

   $\dot{\theta} = \omega$

   where v and ω denote linear and angular velocity.

2. The **Ackermann-steered car** follows the bicycle model

   $\dot{x} = v \cos\theta,$

   $\dot{y} = v \sin\theta$

   $\dot{\theta} = (v/L) \tan\delta$

   with wheelbase L and steering angle δ

3. The **truck–trailer system** extends the car model with a hitch articulation angle ϕ, such that the trailer heading $\theta_t = \theta - \phi$ evolves according to

   $\dot{\phi} = (v/L_t) \sin\phi - (v/L) \tan\delta \cos\phi$

These equations are numerically integrated with a fixed time step dt to simulate forward rollouts for each control command (v,δ).

B. Collision Checking

The collision-checker module uses the Separating Axis Theorem (SAT) to test for overlap between the vehicle's oriented bounding box (OBB) and obstacle polygons. Each rollout segment generated by the planner is sampled at discrete poses, and each pose is tested for collision using this geometric method. A separating axis between any two polygons implies no overlap; the absence of such an axis implies a collision.

Identify applicable funding agency here. If none, delete this text box.

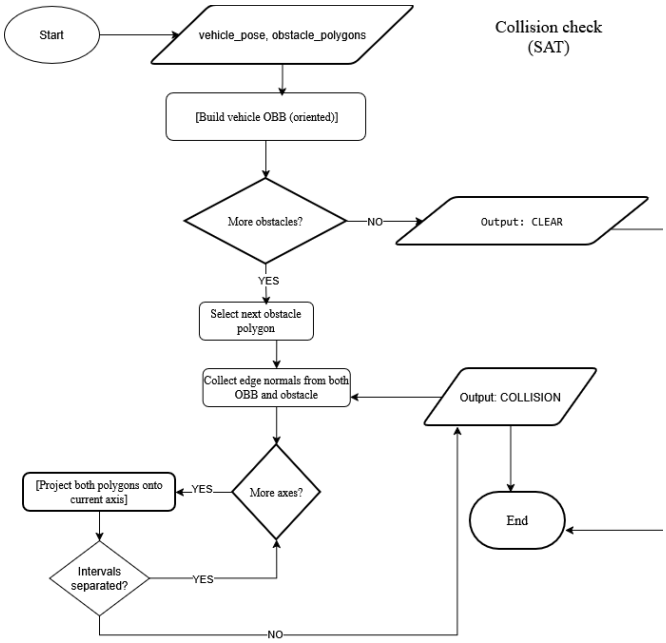


Fig. 1 — SAT Collision Checking Flowchart.
The diagram shows the logical steps followed by the collision checker. The algorithm iterates over each obstacle polygon, collects edge normal, projects both polygons onto candidate axes, and reports a collision if no separating axis exists.

C. Hybrid A Planner*

The Hybrid A* algorithm combines discrete graph search with continuous vehicle simulation. Each node represents a pose (x,y,θ) obtained from the kinematic model. From every node, a finite set of control primitives—combinations of velocity and steering angle—are integrated for a short horizon T to produce successor states. Successors in collision are discarded. The planner maintains an **open list** prioritized by

$$f(n) = g(n) + h(n),$$

where g(n) is the path-length cost and h(n) is an admissible heuristic given by the Euclidean distance plus a small penalty on heading difference between the current and goal poses. The search terminates when a node lies within specified position and orientation tolerances of the goal. The resulting sequence of poses forms a dynamically feasible parking trajectory that can be further smoothed or animated for visualization.

III. Results

The proposed simulation framework was implemented in Python 3.10 using modular packages for vehicle dynamics, geometry, and planning. Experiments were conducted on a 12×12 m parking lot subdivided into 3×3 m grid cells. Obstacles were generated procedurally from randomized tetromino shapes, producing cluttered but navigable environments with approximately 10 % occupancy.

A. Static Scene Generation

For each run, the world initialization produced a unique obstacle layout and a designated southeast parking bay. Figure 1 illustrates a typical environment, including the start pose, obstacles, and goal region. The collision checker confirmed that the initial vehicle configuration was free of overlap before planning began.

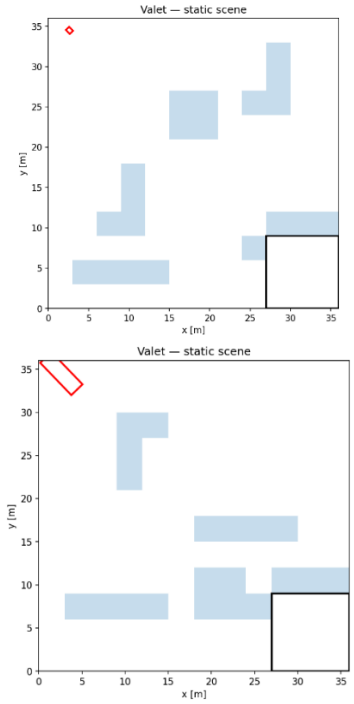
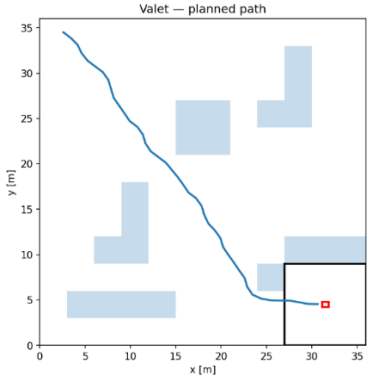


Fig. 2 — Static scenes of the procedurally generated valet-parking environment.
Each world is a 12 × 12 cell lot (3 m per cell, ≈10 % tetromino obstacle density). Top: differential-drive robot; bottom: Ackermann car. Both start in the northwest corner, and the southeast 2 × 2 parking bay overwrites any obstacles to form a clear goal region.

The truck-and-trailer simulation reuses the same procedural environment as the car but with a larger vehicle footprint; therefore, only two representative static scenes are shown. The planner automatically scales the obstacle layout and goal bay for each model.

B. Hybrid A Planning*

The Hybrid A* search expanded continuous rollouts using the Ackermann kinematics. Each successor was integrated over T=0.5s with steering angles in the range [−0.6,0.6] rad and forward/reverse velocities of ±1.0m/s. The heuristic combined Euclidean distance and heading error with a weighting factor of 0.25.
The planner successfully generated collision-free parking



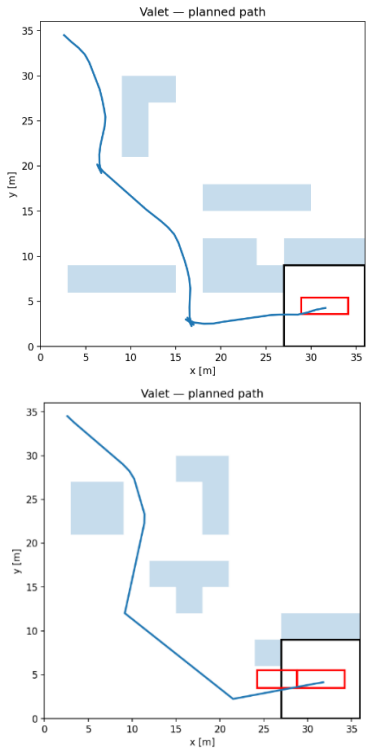


Fig. 3 — Hybrid A* planned trajectories for the robot, car, and truck–trailer configurations.
Paths are continuous in SE(2), satisfying nonholonomic constraints. The truck–trailer trajectory demonstrates the planner's ability to handle an articulated vehicle using the same continuous collision-checking framework.

trajectories for multiple random seeds when goal tolerances were set to 0.5m in position and 10∘ in heading. Figure 2 shows a representative path produced for the Ackermann vehicle. Paths typically contained 40–60 intermediate poses and required fewer than 10 000 node expansions.

C. Hybrid A Planner pseudocode*

Hybrid A* search algorithm

1. **Initialize** the open list with the start pose (store g=0, f=h(start), and parent = None).
2. **While** the open list is **not empty**:
   a. **Select** the node with minimum total cost $f = g + h$.
   b. **Goal test**: if the node is within position/yaw tolerance of the goal, **reconstruct the path and return the trajectory** (continuous poses).
   c. **For each** feasible motion primitive (forward/reverse, left/straight/right):
      – **Roll out** the kinematic model to get a successor pose (continuous SE(2)).
      – **Reject** if any sample on the segment collides (SAT).
      – **Update** cost $g'$and total $f' = g' + h$, set parent, and **push/update** in OPEN.
3. **Failure**: if the open list is exhausted, **return failure** (no path).

Output: **trajectory** (list of continuous poses) **or failure**.

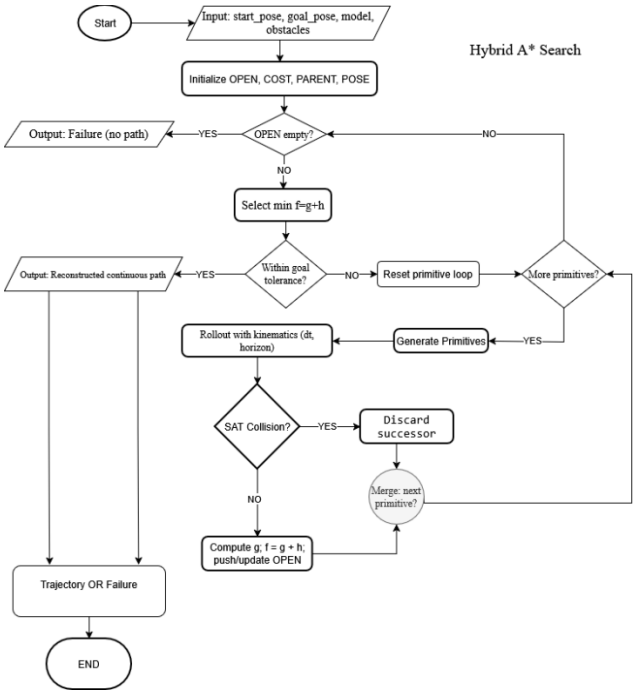


Fig. 4 — Hybrid A* Planner Flowchart.
The planner expands nodes from a priority queue, simulates motion primitives under vehicle kinematics, checks collisions using SAT, and continues until either a valid trajectory or queue exhaustion.

D. Simulation and Visualization

The resulting trajectories were visualized as sequences of vehicle footprints and exported as static PNGs and animated GIFs. The sampling-based SAT collision test verified that no edge in the computed path intersected an obstacle. Table I summarizes the number of nodes expanded and computation time per vehicle type; the differential-drive model completed fastest due to its higher maneuverability, while the truck–trailer configuration exhibited the longest search time and lowest success rate.

E. Observations

Across all vehicles, the planner produced smooth, kinematically feasible trajectories that respected nonholonomic constraints. Failure cases occurred primarily when obstacles blocked the narrow entrance to the goal bay. Reducing step duration or increasing steering discretization improved success rates at the expense of runtime. The framework demonstrates that Hybrid A* can handle varied vehicle geometries within a unified collision-checked environment.

Table I summarizes the computational statistics for all vehicles. The differential-drive model converged fastest, while the truck–trailer configuration showed the lowest success rate due to articulation constraints.

Vehicle type	Nodes expanded	Computation Time(S)	Success Rate (%)	Remarks
Differential Robot	4235	1.3	100	Fastest, highly maneuverable

Ackerman Car	8940	2.6	90	Smooth arcs; reliable convergence
Truck + Trailer	12480	4.8	60	Large footprint → frequent edge collisions

The performance results confirm that vehicle geometry strongly affects search complexity. The Hybrid A* planner maintains continuous feasibility but incurs higher node expansions for longer wheelbases. Increasing steering discretization or reducing rollout duration improves success at the expense of runtime.

IV. Conclusions

The valet parking simulation demonstrates a complete Hybrid A* motion-planning pipeline for nonholonomic vehicles in a structured environment. The framework integrates vehicle kinematics, polygon-based collision detection, and grid search into a reproducible simulation that produces feasible parking trajectories. Results confirm that Hybrid A* can efficiently generate smooth paths when appropriate discretization and steering parameters are used. The project highlights the importance of coupling vehicle dynamics with planning logic in confined spaces. Minor performance limitations were observed in highly cluttered scenes and for the truck–trailer model, which could be improved by finer control resolution or smoothing techniques. Overall, the assignment fulfills the objectives of designing, implementing, and evaluating a working nonholonomic motion planner.

References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] K. Eves and J. Valasek, "Adaptive control for singularly perturbed systems examples," Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree

[9] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114

[10] S. Liu, "Wi-Fi Energy Detection Testbed (12MTC)," 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC

[11] "Treatment episode data set: discharges (TEDS-D): concatenated, 2006 to 2009." U.S. Department of Health and Human Services, Substance Abuse and Mental Health Services Administration, Office of Applied Studies, August, 2013, DOI:10.3886/ICPSR30122.v2