

Nonholonomic Narrow Dead-End Escape with Deep Reinforcement Learning

Denghan Xiong*

ZJUI Institute, International Campus
Zhejiang University
Haining, China
denghan.22@intl.zju.edu.cn

Yutong Chen*

Beijing Jiaotong University
Beijing, China
23222002@bjtu.edu.cn

Yanzhe Zhao*

Tianjin University
Tianjin, China
qq2954253041@tju.edu.cn

Zichun Wang*

University of Nottingham Ningbo China
Ningbo, China
wangzichun2004@gmail.com

ABSTRACT

Nonholonomic constraints restrict feasible velocities without reducing configuration-space dimension, which makes collision-free geometric paths generally non-executable for car-like robots. Ackermann steering further imposes curvature bounds and forbids in-place rotation, so escaping from narrow dead ends typically requires tightly sequenced forward and reverse maneuvers. Classical planners that decouple global search and local steering struggle in these settings because narrow passages occupy low-measure regions and nonholonomic reachability shrinks the set of valid connections, which degrades sampling efficiency and increases sensitivity to clearances. We study nonholonomic narrow dead-end escape for Ackermann vehicles and contribute three components. First, we construct a generator that samples multi-phase forward–reverse trajectories compatible with Ackermann kinematics and inflates their envelopes to synthesize families of narrow dead ends that are guaranteed to admit at least one feasible escape. Second, we construct a training environment that enforces kinematic constraints and train a policy using the soft actor-critic algorithm. Third, we evaluate against representative classical planners that combine global search with nonholonomic steering. Across parameterized dead-end families, the learned policy solves a larger fraction of instances, reduces maneuver count, and maintains comparable path length and planning time while under the same sensing and control limits. We provide our project as an open source on <https://github.com/gitagitty/cisDRL-RobotNav.git>

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning; Robotic planning; • Computer systems organization** → **Robotics.**

* All authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 ACM.

ACM ISBN 978-1-4503-XXXX-X/18/06

<https://doi.org/XXXXXXX.XXXXXXX>

KEYWORDS

Nonholonomic motion planning, Dead-end escape, Ackermann steering, Deep reinforcement learning, Soft Actor-Critic

ACM Reference Format:

Denghan Xiong, Yanzhe Zhao, Yutong Chen, and Zichun Wang. 2018. Nonholonomic Narrow Dead-End Escape with Deep Reinforcement Learning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 6 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

Nonholonomic constraints are nonintegrable relations on configuration derivatives that restrict allowable velocities without reducing the dimension of the configuration space. As a result, an arbitrary collision-free path in configuration space is not necessarily executable by the robot. Differential geometric control links feasibility to accessibility through the Lie algebra rank condition, which establishes local controllability when the generated Lie algebra has full rank, although it does not by itself provide constructive trajectories [6, 7].

A car-like robot with Ackermann steering is a canonical example of a nonholonomic system. Its kinematics impose a tangency constraint and a curvature bound, and the state evolves on $\mathbb{R}^2 \times S^1$ under two controls [6]. In free space, the shortest feasible motions are concatenations of circular arcs and straight segments with possible reversals, that is, cusps, which formalize the need for sequences of forward and reverse maneuvers [11]. Although such systems are locally controllable, small-time local controllability fails at equilibria, so arbitrarily close configurations may still require finite-length motions with nontrivial reorientation [7].

Narrow environments create an additional layer of difficulty. In probabilistic roadmaps, narrow passages occupy regions of small measure in configuration space; therefore, uniform sampling captures them poorly, and specialized sampling, such as bridge tests, remains delicate [16]. More broadly, sampling-based methods can require prohibitive time to discover valid connections in such regions, and the effect is amplified when nonholonomic constraints shape the reachable set [9].

These limitations are particularly acute in the case of dead-end escape for Ackermann vehicles. As free space shrinks, the number

of required maneuvers increases, and feasible motion often demands tightly sequenced forward and reverse actions with steering near saturation [6]. Classical geometric planners that ignore non-holonomic and curvature limits cannot directly supply executable trajectories, and two-stage schemes that approximate a holonomic path and then stitch nonholonomic segments become fragile as clearances tighten near obstacles [5, 6].

Reinforcement learning offers a complementary approach, as it optimizes sequential decisions under kinematic and dynamic constraints derived from interaction data [8]. In parking-like tasks, learning can search for policies that reduce unnecessary gear changes while respecting safety and goal conditions. Recent studies in narrow environments show that learned policies can outperform decoupled planning and control under tight constraints when the feasible action set is explicitly structured [15].

We study nonholonomic narrow dead-end escape for Ackermann vehicles with three components. First, we synthesize training and test scenarios by probabilistically generating trajectories that are compatible with Ackermann kinematics, consisting of multi-phase forward and reverse segments. We then form envelopes so that each instance admits at least one feasible escape path. Second, we construct a training environment that enforces the nonholonomic constraints and train a policy using the soft actor-critic (SAC) algorithm. Third, we compare the learned policy with representative classical planners that combine global and local stages under the same scenarios and metrics.

2 RELATED WORK

2.1 Traditional Path Planning

Nonholonomic constraints pose significant challenges to classical path planning algorithms, as a robot subject to such constraints cannot move in arbitrary directions at any given time (e.g., an Ackermann-steered vehicle cannot move sideways). Consequently, not all geometric paths generated by classical planners are physically feasible. This limitation reduces the applicability of graph-based methods such as the A* algorithm [3] and its numerous variants [15], as they generally disregard kinematic constraints.

Sampling-based planners, such as Rapidly-Exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), scale well in high-dimensional configuration spaces. For instance, FastBKRRT [10] demonstrates superior performance in motion planning for Ackermann-steered vehicles; however, its effectiveness may not extend to scenarios that demand frequent and precise pose adjustments in highly confined environments.

Obstacle avoidance strategies such as the Dynamic Window Approach (DWA), Artificial Potential Field (APF), and Timed Elastic Band (TEB) are widely employed due to their real-time responsiveness and computational efficiency. [14] proposes a deterministic sampling method for DWA that explicitly accounts for uncertainty in differential-drive mobile robots. Additionally, reactive behaviors, such as backup-turn heuristics and velocity-space methods, can rapidly generate collision-free reversal maneuvers. Nevertheless, these approaches are typically short-sighted, prone to local minima, and limited to single-step responses.

2.2 Path Planning with Deep Reinforcement Learning

In contrast to traditional navigation frameworks that rely on high-precision global maps and accurate sensor inputs, DRL enables mapless navigation directly from onboard sensing, thereby enhancing adaptability [17]. RL-based methods have been applied to non-holonomic robots in various contexts. For example, [1] employs DDPG to balance path tracking and obstacle avoidance in wheeled mobile robots subject to nonholonomic constraints. Similarly, [12] demonstrates virtual-to-real transfer using ADDPG with sparse lidar observations, while [4] introduces a survival-penalty reward to address sparse feedback, showing that TD3 outperforms DDPG. Extending the application of RL to path planning, [2] adapts DRL to Ackermann-steered vehicles navigating maze-like environments, highlighting that RL can achieve high performance even under restrictive motion constraints.

Beyond navigation, RL has also been used to resolve contention and collisions in communication systems. Shuai et al. integrate tabular Q-learning into framed slotted ALOHA and propose a fast-convergence MAC protocol that learns a collision-free TDMA-like schedule from local ACK feedback, achieving higher throughput and shorter convergence time than classical ALOHA variants [13]. This line of work further supports the view that RL can learn effective policies in environments dominated by collision and congestion phenomena, which is analogous to our cul-de-sac setting where nonholonomic constraints and narrow geometry create severe “bottlenecks” in the state space.

DRL has also been integrated with traditional path planning techniques. For instance, [15] propose an SAC-based escape policy for differential-drive robots (robotic vacuums), augmented with A* demonstrations. Collectively, these studies demonstrate that DRL not only improves robustness over conventional path planning algorithms but also shows strong potential in addressing the challenges of navigation under stringent nonholonomic constraints.

3 PRELIMINARY

3.1 RL MDP Settings

We cast dead-end escape as a Markov Decision Process (S, A, P, R, γ) . The state $s \in S$ concatenates (i) a LiDAR embedding z_{lidar} from a 2D 360° scan by sectorizing into k bins and taking per-sector min/max after clipping at d_{max} , (ii) the relative goal bearing (d, θ) from odometry/IMU, and (iii) previous control (v, ω) for action smoothing. The action $a \in A$ is (v, δ) with bounds $|v| \leq v_{\text{max}}$ and $|\delta| \leq 1$. The reward combines sparse goal success and collision penalty with light shaping for motion/alignment:

$$r = \lambda_{\text{col}} r_{\text{col}} + \lambda_{\text{goal}} r_{\text{goal}} + \lambda_{\text{move}} (|v| + |\omega|) + \lambda_{\text{vel}} v \cos \theta.$$

The objective is to learn a policy $\pi(a|s)$ maximizing expected return under these constraints.

3.2 Robot Model (Ackermann Kinematics for Nonholonomic Platforms)

We use a four-wheeled, front-wheel-steering platform (JetAcker) that obeys Ackermann steering geometry. Its planar kinematics are

$$\dot{x} = v \cos \theta, \quad \dot{y} = v \sin \theta, \quad \dot{\theta} = \frac{v}{L} \tan \delta,$$

where L is the wheelbase (we also denote it by H in implementation), v the longitudinal velocity, and δ the front-wheel steering angle. Actions output a normalized steering command $\hat{\delta} \in [-1, 1]$ mapped by $\delta = \delta_{\max} \hat{\delta}$, hence the yaw rate used by the low-level controller is

$$\omega = \dot{\theta} = \frac{v}{L} \tan(\delta_{\max} \hat{\delta}) \equiv \frac{v}{H} \tan(\delta_{\max} \hat{\delta}).$$

Ackermann vehicles satisfy a nonholonomic constraint (no lateral slip).

$$\dot{y} \cos \theta - \dot{x} \sin \theta = 0,$$

which prohibits sideways motion and in-place rotation. The curvature is $\kappa = \tan \delta / L$ so the turning radius is $R = 1/|\kappa| = L/|\tan \delta| \geq R_{\min} = L/\tan \delta_{\max}$. These curvature bounds and the inability to spin make cul-de-sac escape inherently multi-step (e.g., back-and-forth) and sensitive to geometry—precisely the setting where learning policies that exploit contact-free maneuvering and goal-directed bias are beneficial.

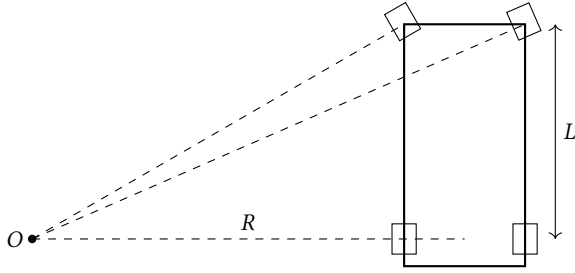


Figure 1: Schematic of Ackermann Steering Geometry

3.3 Problem Statement: Assumptions and Objective

Assumptions. (i) Planar motion with static obstacles during an episode; (ii) onboard odometry/IMU provides bounded-drift relative pose; (iii) a 2D LiDAR with full 360° coverage; (iv) no prior map; (v) control period Δt with bounds $|v| \leq v_{\max}$, $|\delta| \leq \delta_{\max}$ (equivalently $|\dot{\delta}| \leq 1$).

Goal. Starting from a pose possibly inside a cul-de-sac, synthesize a collision-free control sequence that (1) reliably exits the dead-end and (2) reaches the provided goal region within horizon T , subject to Ackermann kinematics and curvature limits.

Optimization Target. Maximize success rate and expected return while minimizing path length/steps under the above constraints.

3.4 Environment Generation for Training and Evaluation

We procedurally generate narrow dead-end layouts with a guaranteed feasible escape by starting from a kinematically valid seed trajectory for an Ackermann-steered robot, wrapping its exact swept area into a compact envelope, and then converting the envelope boundary into obstacles while cutting a single exit aligned with the final heading. By construction, the seed followed by a short straight extension through the exit is collision-free, which provides

families of evaluation maps that are both challenging and feasible under nonholonomic constraints.

Seed trajectories are synthesized under Ackermann kinematics with bounded velocity and steering. The sampler integrates the planar model and produces multi-phase maneuver sequences that include both forward and reverse motion. Two maneuver styles are emphasized. One favors translation along a corridor and yields long, almost straight motion. The other favors tight turning with small displacements and yields turn in place like sequences without violating the no lateral slip constraint. The dataset mixes these styles in a prescribed proportion and includes instances that exit the dead end by moving forward as well as instances that exit by reversing. Each seed is densified along arclength and is then extended a short distance beyond the exit heading to define a clear goal condition and a reference demonstration.

The swept area is obtained by placing the inflated rectangular footprint at each pose of the densified seed and taking the geometric union, followed by a light smoothing and simplification step that closes microscopic gaps while preserving topology. The exit is located by marching from the final pose along the final heading until the footprint clears the envelope interior. Then, remove the boundary segment intersected by a thin strip aligned with that heading, so that a single navigable gap remains.

Two obstacle realizations are instantiated for each envelope in order to probe sensitivity to boundary density. A continuous wall variant extrudes the boundary into thin box walls of fixed thickness and height. A sparse cylinder variant samples points along the boundary at a spacing smaller than the vehicle width and places circular posts that block leakage everywhere except at the exit. Layouts are tiled across many disjoint subregions to form large batches for training and evaluation. For each instance, we export the start pose, the target region at the exit, the seed trajectory, and its corresponding control sequence with timestamps, together with simulator assets. This yields reproducible scenarios that respect nonholonomic kinematics, span long corridors, and turn dominant behaviors in controllable proportions, systematically covering both forward and reverse escape modes.

4 METHODOLOGY

4.1 Overview

We train a continuous-control policy for cul-de-sac escape using *Soft Actor-Critic* (SAC), deployed in a ROS 2 + Gazebo loop. At each control cycle ($\Delta t = 0.1$ s), the agent reads onboard LiDAR and odometry, outputs linear speed and a normalized steering command, which are mapped to Ackermann-consistent angular velocity and executed on the robot.

4.2 Observations and Actions

State. The policy input concatenates

$$s_t = [z_{\text{lidar}}^{(40)} \parallel d_t, \cos \theta_t, \sin \theta_t \parallel v_{t-1}, \omega_{t-1}],$$

where $z_{\text{lidar}}^{(40)}$ is the downsampled/sectorized 2D LiDAR (40 values after range clipping and inf masking), (d_t, θ_t) are goal bearing features from odometry/IMU, and (v_{t-1}, ω_{t-1}) are previous controls for action smoothing. This yields a 45-D input as in the implementation.

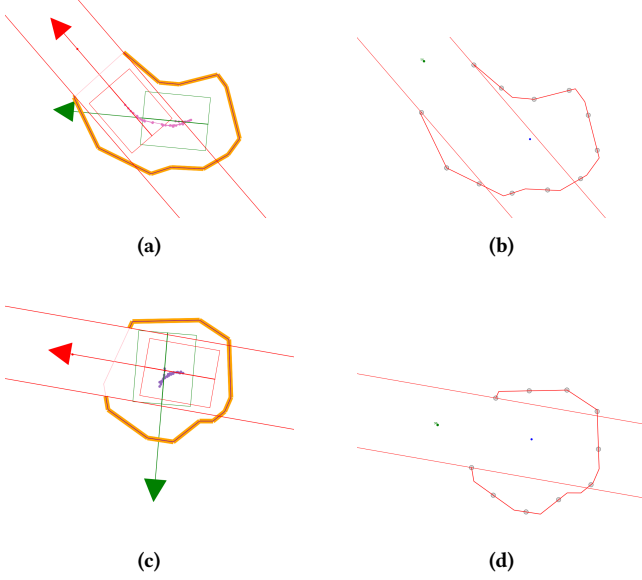


Figure 2: Illustration of procedurally generated narrow dead-end layouts under different trajectory styles and obstacle realizations. Subfigures (a) and (b) are generated from a trajectory biased toward long forward/reverse translation along corridors, whereas (c) and (d) are generated from a trajectory biased toward in-place maneuvering with short displacements and frequent heading changes. For each style, (a) and (c) instantiate continuous wall boundaries, while (b) and (d) instantiate sparse cylindrical boundaries. The green box and arrow indicate the start pose of the seed trajectory, and the red box denotes the terminal pose.

Action and Ackermann mapping. The actor outputs $a_t = (\hat{v}_t, \hat{\delta}_t) \in [-1, 1]^2$. We execute

$$v_t = \hat{v}_t, \quad \omega_t = k_p \frac{v_t}{H} \tan(\hat{\delta}_t \cdot \max_rad),$$

with $k_p=1$, $\max_rad=0.645$ rad and $H=0.21$ m. This realizes the Ackermann yaw-rate model $\omega=(v/L) \tan \delta$ (here $L \equiv H$) while keeping the policy output in a compact normalized range.

4.3 Reward Design

At each step, we combine large terminal signals (goal/collision/crash) with light shaping for movement and goal alignment:

$$r_t = 500 \mathbf{1}[\text{goal}] - 100 \mathbf{1}[\text{collision}] - 500 \mathbf{1}[\text{crash}] + |v_t| + |\omega_t| + v_t \cos \theta_t. \quad (1)$$

A goal is declared when the robot is within 0.2 m of the target without collision. The terminal rewards are set on the order of a few hundred so that safety clearly dominates: both crashes (−500) and minor collisions (−100) are far more costly than any shaping gain. This reflects that, in our dead-end scenarios, a collision can push the car into an unrecoverable pose and, in the real world, can damage the robot and its sensors, so the agent is strongly discouraged from hitting walls. The small shaping terms $|v_t| + |\omega_t|$ and $v_t \cos \theta_t$ encourage purposeful motion and help mitigate sparse

rewards: $|v_t| + |\omega_t|$ discourages freezing, while $v_t \cos \theta_t$ rewards the component of velocity aligned with the exit direction, effectively encouraging the robot to turn toward the opening and move along it, whether it ultimately escapes going forward or in reverse. We assign zero reward when LiDAR data are missing so that sensor glitches do not corrupt the learning signal.

4.4 SAC Training Loop

We use a standard SAC agent (stochastic Gaussian policy; twin Q critics with target networks; replay buffer). Training alternates environment interaction and batched gradient updates:

- **Warm start & offline pretraining.** If enabled, we load a replay buffer from `assets/data.yml` and run 100 pretraining iterations before online interaction.
- **Online interaction.** Each episode runs up to 500 steps; transitions $(s_t, a_t, r_t, s_{t+1}, d_t)$ are appended to a replay buffer of capacity 10^6 .
- **Updates.** Every 2 episodes, we perform 500 SAC update iterations with batch size 40 (uniform sampling).
- **Persistence.** We periodically serialize recent trajectories back to `assets/data.yml` for future pretraining.

4.5 Environment Sampling and Curriculum

Episodes cycle through a configuration list (`configs.json`) that specifies the robot’s start pose and target position. For each episode, we reset the simulation, place entities accordingly, and then execute the policy. Upon reaching the target, we slightly increase the internal target-distance budget (up to 8 m), effectively expanding the operating envelope during training while keeping the geometry fixed and reproducible.

5 SIMULATION EXPERIMENTS

5.1 Simulation Environment

The simulation environment is established in Gazebo, where we generate an environment with a specific method: first, we define a certain position and a random direction as the initial state of a robot with a virtual car length, car width, and wheelbase width that can be defined by users and let the robot move randomly for 50 steps, then record the final state. After that, we generate the wall around the covered area during the movement and make an exit a little wider than the car in front of the final state. Each process will generate a continuous wall version and a version with separate pillars. In an environment, there will be n subsections with the process in each section, where n can be defined by users. The world file is generated during the recording of the initial state and the target position for training.

5.2 Evaluation Metrics

- **Escape success rate on unacquainted layouts with zero-shot.** We evaluate the trained policy without any further learning on a held-out set of N novel dead-end maps. For each map, we run M independent episodes (with randomized start orientations) and compute

$$\text{Success Rate} = \frac{\text{total escapes}}{N \times M}.$$

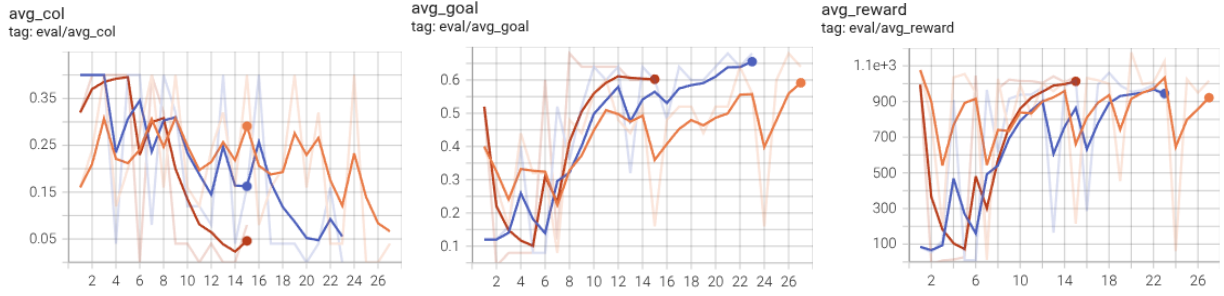


Figure 3: Training results, where the orange line represents the results on the easy environment, the blue line represents the results on the medium environment, and the red line represents the results of the hardest environment.

This measures the policy’s ability to generalize directly to unseen geometries.

- **Average steps to escape & collision count.** Over the same test episodes, we record:

- *Mean Steps to Escape*: average number of control steps taken in successful episodes.
- *Mean Collision Count*: average number of obstacle contacts across all episodes (successful and failed).

These metrics quantify both efficiency (how quickly the robot escapes) and safety (how often it collides).

5.3 Training Results

To evaluate the proposed DRL framework, we present the training results and conduct the simulation experiment in various environments with a comparison of other algorithms. The policy was trained in Gazebo on an RTX 4070 Laptop for around 30 hours. The reward discounting factor is set to 0.99. In each epoch, we train 70 episodes.

Our training framework adopts the curriculum training method. We gradually decrease the length, width, and size of the wheelbase of the virtual car from $length = 0.47cm$, $width = 0.46cm$, $wheelbase = 0.363cm$ to $length = 0.37cm$, $width = 0.36cm$, $wheelbase = 0.263cm$ when establishing the environment. We adjust the difficulty of the environment after the training goal rate reaches 60%. Fig. 3 shows the collision rate, goal rate, and reward per episode during the training process.

Although the learned policy clearly outperforms all classical baselines, it still fails in extremely tight or highly irregular dead ends. In such cases the LiDAR observations become nearly symmetric and the robot may commit too early to a suboptimal turning direction, exhausting its manoeuvring room before discovering the correct escape sequence. We also observe occasional failures when long sequences of reversals are required, suggesting that very deep forward–reverse patterns remain challenging for the current policy architecture and training horizon.

5.4 Baselines and Comparative Results

To better highlight the effectiveness of our proposed method, we compare it against three representative baselines:

Table 1: Performance on unseen dead-end layouts (mean \pm std (CI95)).

Method	Success Rate (%)	Steps	Collisions
DRL (ours)	71.85 ± 5.68 (3.71)	67.9 ± 16.50 (14.46)	5.14 ± 3.00 (2.63)
Hybrid A*	7.20 ± 0.76 (0.67)	24.50 ± 10.18 (8.93)	1.30 ± 0.60 (0.53)
ROS2 TEB	37.33 ± 0.08 (0.07)	320.58 ± 45.97 (63.82)	8.22 ± 6.88 (1.01)
FTG	25.73 ± 2.94 (2.88)	99.60 ± 31.90 (31.20)	12.03 ± 0.98 (0.96)

Each method is evaluated over 180 dead-end instances, repeated across five random generations ($M=180$, $N=5$, 900 trials in total). Reported means, standard deviations, and CI95 values are computed over all 900 trials.

- **DRL**: a plain Soft Actor–Critic agent trained end-to-end.
- **ROS2 TEB**: the standard Nav2 global planner using the TEB local planner with a local controller whose parameters are adapted to our platform. TEB is widely adopted in ROS2 navigation stacks, so it serves as a strong map-based reference for comparison in our setting, even though it can incur high replanning cost and get stuck in cul-de-sacs.
- **FTG (Follow-The-Gap)**: a LiDAR-only reactive baseline that selects the steering direction corresponding to the largest visible gap. We form a safety bubble around the closest obstacle, detect continuous gap segments, and steer toward the midpoint of the widest gap. To enable dead-end escape, we add a reversal–turn heuristic when forward clearance becomes insufficient. FTG thus acts as a minimal classical baseline that requires no global costmap or offline planning, highlighting how far a purely reactive policy can go using only instantaneous geometry.
- **Hybrid A***: a lattice-based kinodynamic planner used as a deterministic classical baseline. LaserScan data are rasterised into a local 2D occupancy grid, and planning is performed over an (x, y, θ) state lattice (0.05 m resolution, 32 headings). Successor states are generated by rolling out the Ackermann kinematics under steering limits in both directions and validated using full-footprint collision checks. The resulting path is tracked via a pure-pursuit controller. Hybrid A* is a widely used planner for car-like robots, so it provides a strong kinodynamic reference independent of learning.

We evaluate all methods on unseen procedurally generated dead-end maps, using the metrics defined previously. Table 1 summarizes the comparison. The proposed DRL framework consistently

achieves the highest escape success rate and a low collision count. While ROS2 TEB guarantees global path optimality with a map, it suffers from partially observable cul-de-sacs. Hybrid A* fails in narrow dead ends because footprint inflation and coarse lattice resolution prune most feasible motion primitives, making multi-point turns inexpressible. FTG fails for the opposite reason: its purely reactive gap selection cannot generate deliberate backward manoeuvres once gaps fall below the admissibility threshold. Both baselines thus struggle when clearance becomes comparable to vehicle width. In summary, the proposed method outperforms all classical baselines: it avoids the local-minima issues of TEB, overcomes the memoryless limitations of FTG, and achieves finer manoeuvring than Hybrid A* without incurring its heavy discretisation cost.

6 CONCLUSION

We studied nonholonomic narrow dead-end escape using deep reinforcement learning and presented a pipeline that couples a feasibility-guaranteed scenario generator, a training environment that enforces car-like kinematics, and a soft actor-critic policy. On unseen layouts, the learned policy achieves higher success with fewer maneuvers and fewer contacts than representative classical planners, indicating that learning to sequence forward–reverse actions under curvature limits is effective when passages are tight and geometric connections are scarce.

Nevertheless, the overall success rate remains moderate, reflecting the inherent difficulty of navigating dead ends and escaping under nonholonomic constraints. Going forward, we will focus on improving experimental coverage and measurement to enable more informative comparisons, enhancing safety, and reducing collisions. Planned extensions include richer scenario families and ablations, unified timing and step metrics across controllers, stronger baseline configurations, and practical evaluation on real platforms with safety monitors and conservative execution to validate deployability.

REFERENCES

- [1] Xiuquan Cheng, Shaobo Zhang, Sizhu Cheng, Qinxian Xia, and Junhao Zhang. [n. d.]. Path-Following and Obstacle Avoidance Control of Nonholonomic Wheeled Mobile Robot Based on Deep Reinforcement Learning. 12, 14 ([n. d.]), 6874. <https://doi.org/10.3390/app12146874>
- [2] Daniel Gleason and Michael Jenkin. [n. d.]. Nonholonomic Robot Navigation of Mazes Using Reinforcement Learning. In *Proceedings of the 19th International Conference on Informatics in Control, Automation and Robotics* (Lisbon, Portugal, 2022). SCITEPRESS - Science and Technology Publications, 369–376. <https://doi.org/10.5220/0011123600003271>
- [3] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107. <https://doi.org/10.1109/TSSC.1968.300136>
- [4] Shyr-Long Jeng and Chienhsun Chiang. [n. d.]. End-to-End Autonomous Navigation Based on Deep Reinforcement Learning with a Survival Penalty Function. 23, 20 ([n. d.]), 8651. <https://doi.org/10.3390/s23208651>
- [5] Mingyang Jiang, Yueyuan Li, Songan Zhang, Siyuan Chen, Chunxiang Wang, and Ming Yang. 2024. HOPE: A Reinforcement Learning-based Hybrid Policy Path Planner for Diverse Parking Scenarios. *arXiv preprint arXiv:2405.20579* (2024).
- [6] J.-P. Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. 1994. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation* 10, 5 (Oct. 1994), 577–593. <https://doi.org/10.1109/70.326564>
- [7] J. P. Laumond, S. Sekhavat, and F. Lamiriaux. 1998. Guidelines in nonholonomic motion planning for mobile robots. In *Robot Motion Planning and Control*, J. P. Laumond (Ed.). Vol. 229. Springer-Verlag, London, 1–53. <https://doi.org/10.1007/BFb0036070> Series Title: Lecture Notes in Control and Information Sciences.
- [8] Zhuoren Li, Guizhe Jin, Ran Yu, Zhiwen Chen, Nan Li, Wei Han, Lu Xiong, Bo Leng, Jia Hu, Ilya Kolmanovsky, and Dimitar Filev. 2025. A Survey of Reinforcement Learning-Based Motion Planning for Autonomous Driving: Lessons Learned from a Driving Task Perspective. <https://doi.org/10.48550/arXiv.2503.23650> [cs].
- [9] Andreas Orthey and Marc Toussaint. 2021. Section Patterns: Efficiently Solving Narrow Passage Problems in Multilevel Motion Planning. *IEEE Transactions on Robotics* 37, 6 (Dec. 2021), 1891–1905. <https://doi.org/10.1109/TRO.2021.3070975>
- [10] Jie Peng, Yu'An Chen, Yifan Duan, Yu Zhang, Jianmin Ji, and Yanyong Zhang. 2021. Towards an Online RRT-based Path Planning Algorithm for Ackermann-steering Vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 7407–7413. <https://doi.org/10.1109/ICRA48506.2021.9561207>
- [11] James Reeds and Lawrence Shepp. 1990. Optimal paths for a car that goes both forwards and backwards. *Pacific J. Math.* 145, 2 (Oct. 1990), 367–393. <https://doi.org/10.2140/pjm.1990.145.367>
- [12] Lei Tai, Giuseppe Paolo, and Ming Liu. 2017. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Vancouver, BC, 31–36. <https://doi.org/10.1109/IROS.2017.8202134>
- [13] Taizhou University, Taizhou, 225300, China, Shuai Xiaoying, Yin Yuxia, and Zhang Bin. 2021. A Fast Convergence ALOHA Based on Reinforcement Learning. *International Journal of Computer Theory and Engineering* 13, 3 (2021), 96–99. <https://doi.org/10.7763/IJCTE.2021.V13.1296>
- [14] Shinya Yasuda, Taichi Kumagai, and Hiroshi Yoshida. 2023. Safe and Efficient Dynamic Window Approach for Differential Mobile Robots With Stochastic Dynamics Using Deterministic Sampling. *IEEE Robotics and Automation Letters* 8, 5 (May 2023), 2614–2621. <https://doi.org/10.1109/LRA.2023.3257681>
- [15] Han Zheng, Jiale Zhang, Mingyang Jiang, Peiyuan Liu, Danni Liu, Tong Qin, and Ming Yang. [n. d.]. Embodied Escaping: End-to-End Reinforcement Learning for Robot Navigation in Narrow Environment. <https://doi.org/10.48550/arXiv.2503.03208> [cs]
- [16] Zheng Sun, D. Hsu, Tingting Jiang, H. Kurniawati, and J.H. Reif. 2005. Narrow passage sampling for probabilistic roadmap planning. *IEEE Transactions on Robotics* 21, 6 (Dec. 2005), 1105–1115. <https://doi.org/10.1109/TRO.2005.853485>
- [17] Kai Zhu and Tao Zhang. 2021. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology* 26, 5 (2021), 674–691. <https://doi.org/10.26599/TST.2021.9010012>