2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Enhancing Classical Motion Planners Using RL with Safety Guarantees

Elias Goldsztejn[1] and Ronen I. Brafman[1]

Abstract— **Classical algorithms for autonomous navigation, while well-understood and safe, require manual parameter tuning by experts to perform well. APPL [1] and similar methods use machine learning to dynamically adjust planner parameters during deployment. This approach maintains the safety of classical systems but remains constrained by the underlying algorithm. Instead of parameter tuning, we suggest using classical planners to regulate action selection of a reinforcement learning (RL) algorithm. The resulting policy is *provably* similar to the well-understood classical algorithm, performs better than both a well-tuned classical planner and an unregularized RL-based policy, and can be shown to respect a user-controlled trust region even during training. In experiments, our method reduces traversal time by 8% (vs. DWA [2]) and 43% (vs. TEB [3]), and lowers proximity risk by 24% and 17%, respectively, while matching or surpassing learning-based baselines and aligning more closely with user preferences.**

INTRODUCTION

Classical local navigation algorithms like the Dynamic Window Approach (DWA) [2], and Timed Elastic Bands (TEB) [3] are widely used in robotics because they generate safe and well-understood behavior, and navigate well in most cases. However, they can be suboptimal in some situations, such as moving too slowly in open areas or struggling in crowded spaces. The traditional solution is to manually retune their parameters, which requires expert knowledge and may degrade performance in previously successful scenarios.

Two general approaches address these shortcomings. The first forgoes classical algorithms and uses deep reinforcement learning (RL) [4]–[9] or imitation learning (IL) to train a path planner [10]–[12]. RL often performs well, but returns a black box controller, raising safety and explainability issues; and IL requires expert demonstrations and suffers from distribution drift, implying potentially unsafe behavior in previously unseen situations.

A second approach is to use learning to improve classical solvers. Adaptive Planner Parameter Learning (APPL) [1] enhances existing navigation systems by dynamically adjusting their parameters during deployment using various ML techniques. These techniques include learning contextual parameters from tele-operated demonstrations [13], incorporating corrective interventions from non-expert users [14], learning from binary or scalar evaluative feedback [15], and using RL in simulations [16]. DWA-RL [17] enhances a classical algorithm by learning to select among the actions it proposes. Both methods boost the performance of local classical algorithms while maintaining the safety and transparency they offer. However, both methods are too tightly constrained by the classical algorithm and its greedy nature, whereas RL looks ahead beyond the next action.

This paper seeks to combine the advantages of both end-to-end and classical approaches, specifically by regularizing an RL algorithm using a classical planner. Regularization is a standard RL technique, originally used to encourage exploration and robustness [18]. More recently, [19] used regularization to combine RL with IL. They trained a car controller in simulation [20] with an RL objective that penalizes deviations from expert demonstrations (see Fig. 2). This technique learns good behaviors faster and, more importantly, helps maintain trust in the generated policy. We strongly build on this idea, demonstrating its effectiveness in the domain of robot motion planning. However, instead of expert demonstrations, we rely on a classical algorithm.

Our method has several advantages. First, our path planner typically behaves similarly to the classical algorithm, making it safe and understandable. In fact, it provably stays within a well-defined trust-region around the classical algorithm's policy during deployment and often, in practice, during training. Second, our extensive empirical evaluation, both on a sophisticated and challenging simulation platform and on real robots, using objective and subjective measures, and with two different classical algorithms, shows that our method improves on both RL-based parameter tuning and vanilla RL. Third, our method does not require expert demonstrations and is not limited by pre-generated demonstrations, as we can automatically query the classical algorithm for its choice in any state. Finally, we also inherit the advantage of faster training times demonstrated by earlier methods.

In summary, our main contributions are: (1) The first application, to our knowledge, of RL regularized by a classical algorithm in motion planning. (2) Provably guaranteed trust region. (3) Demonstrating the strength of this method, both in retaining similarity to the classical planner and in improved objective and subjective performance measures, via a comprehensive empirical evaluation within a challenging simulation environment *and* in the real world.

For code and videos, refer to the repository: https://github.com/ecmpurlsg/Enhancing-Classical-Motion-Planners-Using-RL-with-Safety-Guarantees

BACKGROUND

Classical Mobile Planning: In mobile robot planning, there are two main components: global and local planning.

*This work was supported in part by The Israel Science Foundation (grant No. 573/25), the Israel Ministry of Science and Technology (grant No. 08602), Ben-Gurion University of the Negev through the Agricultural, Biological, and Cognitive Robotics Initiative, and the Lynn and William Frankel Center for Computer Science.

[1]Elias Goldsztejn and [1]Ronen I. Brafman are with Faculty of Computer and Information Science at Ben Gurion University. eliasgol@post.bgu.ac.il, brafman@bgu.ac.il

Fig. 1: RoverZero robot used in our real-world experiments; equipped with LiDAR and Odometry sensor.

Global planning maps a path from start to destination over long distances using static knowledge, like maps, and methods such as A* or rapidly exploring random trees (RRT) [21]. Local planning deals with real-time actions, following the global path but adjusting for dynamic obstacles like people.

DWA is a popular local planning algorithm that samples velocities, simulates trajectories, evaluates them for safety and goal alignment, and selects the best trajectory. Timed Elastic Bands (TEB) is a local planner that formulates motion as an online optimization, deforming a time-parameterized "band" of poses to balance smoothness, speed, and goal progress under kinematic and obstacle constraints. While DWA and TEB optimize safety and goal-reaching, they often require parameter re-tuning for different environments and do not fully account for interactions with people, which prompted the rise of ML-based approaches.

RL and Mobile Planning: RL enables robots to learn optimal behavior through trial and error in dynamic environments. It deals with control challenges in a Markov decision process (MDP) using state space S, action space A, transition function Tr, reward function r, and discount factor γ. The goal is to learn a policy $\pi(a_t \mid s_t)$ $(a_t \in A, s_t \in S)$ that maximizes long-term cumulative reward:

$$\max_{\pi} \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1}$$

Here, τ is a trajectory, $p_\pi(\tau)$ is the trajectory distribution under policy π, and T is the time horizon. In mobile planning, the state of a mobile robot includes local data (e.g., surroundings image, waypoint, velocity), while actions control movement (e.g., velocity commands). Rewards penalize collisions and delays. By trying different actions, a simulated mobile robot learns which ones yield the highest cumulative reward, optimizing for fast and safe goal achievement.

Actor-Critic (AC) algorithms combine an actor (policy) $\pi(a_t \mid s_t; \theta^\pi)$, and a critic (value function) $Q_\pi(s_t, a_t; \theta^Q)$. θ^π are the parameters of the policy (typically, weights of a deep network); θ^Q are the parameters of the Q function that seeks to approximate the long-term cumulative rewards of executing a_t in state s_t, and then acting based on π. TD3 (Twin Delayed Deep Deterministic Policy Gradient) [22] is an AC algorithm that uses two critic networks to reduce overestimation bias, delays policy updates to ensure stability, and adds noise to target actions to smooth policy updates.

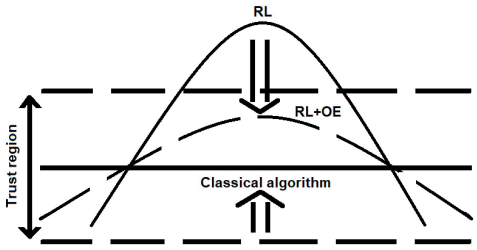

Fig. 2: A classical planner guides policy toward its own behavior, ensuring alignment. The RL loss enables the policy to surpass the classical planner by optimizing performance.

Regularized RL: A variant of TD3, called TD3+BC [23], incorporates a behavior cloning term to regularize the policy. It assumes access to a set $D = \{(s^i, a^i)\}$ of expert choices. Instead of optimizing the standard objective:

$$\pi = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim D}[Q(s, \pi(s))] \tag{2}$$

TD3+BC optimizes the following regularized objective:

$$\pi = \arg\max_{\pi} \mathbb{E}_{(s,a)\sim D} \left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2 \right] \tag{3}$$

Thus, TD3+BC seeks to balance long-term value with a behavior cloning term that penalizes large deviations from the expert demonstrations. λ regulates this trade-off.

This regularization technique reduces training time and develops robust, low-variance policies. Originally designed for offline use to guide policies toward an expert prior, it also works effectively in online settings with similar benefits.

Related Work

Learning for Navigation: Autonomous navigation for mobile robots has been a key focus in robotics. Rapid advances in machine learning (ML) have led to its widespread application in this area. Many of these learning approaches are end-to-end (directly mapping states to actions) [4], [6], [8], [24]. While navigation algorithms trained using RL and IL can perform well, they can be unreliable in new environments and in states that are not typically encountered.

Adaptive Parameters for Classical Mobile Planning: [25] demonstrated that combining ML techniques with classical navigation components often leads to improved and more reliable behavior. Classical planners offer proven safety, explainability, and generalizability, and ML algorithms that build on these planners inherit these crucial aspects. Notably, the APPL paradigm [13]–[16] can re-tune parameters in real-time to adapt to different regions of a complex environment. APPL-Reinforcement (APPLR) [16] stands out by learning in simulation using RL, which eliminates the need for human involvement and enhances reproducibility.

Our method is motivated by this approach. Similarly to APPLR, we use RL in simulation to derive a policy guided by a baseline classical planner. However, instead

of learning parameter space adjustments, we directly learn actions that are regularized by the classical planner. The learned policy remains provably close to that of the classical planner, thus inheriting much of its safety and predictability characteristics, but improves upon it by enabling a broader range of actions and going beyond local optimizations.

Regularized RL for Navigation: Straightforward RL implementations can suffer from poorly designed reward functions or simulation-to-reality gaps, which result in unsafe or ineffective real-world policies. To address these issues, hybrid methods use expert demonstrations and attempt to combine IL with RL to regularize the policy [23], [26]–[29]. Several studies applied these methods to address autonomous driving challenges. For instance, [30] introduced a Kullback-Leibler (KL) divergence method that constrains the RL policy using human demonstrations for lane-changing decisions in urban navigation, formulating the problem as a Lagrangian dual optimization. [19] implemented a regularized RL approach based on TD3+BC for autonomous driving using real-world human driving data.

Our approach builds on [19] and TD3+BC, with several key differences: (1) Instead of relying on behavioral cloning from human experts, we use direct actions from an online planner. While we experiment with DWA and TEB within the Robot Operating System (ROS) [31] navigation stack, the method can be used within any other navigation framework. Beyond saving the effort of obtaining expert demonstrations, we are not restricted to offline data. Our algorithmic "expert" can be queried online on any state. (2) We train in a reactive simulation environment where people interact dynamically with the robot, rather than relying on log-playback scenes with no interaction. (3) We also evaluate our algorithms in real-world situations. (4) Our method provably restricts the actor to a trust region. (5) For state representation, we use only LiDAR and odometry, similar to APPLR, as they are widely available on most mobile robots.

A closely related method uses RL to learn a residual policy that modifies the classic policy [32]. By restricting the size of such modifications, one can also ensure that the combined policy remains within a trust region. As we show later, this method does not perform as well as our approach.

Approach

Problem Definition

We address mobile navigation within a building where a global planner uses A* to generate a path. We model the navigation problem as an MDP $\langle S, A, T, \gamma, R)\rangle$. States in S include the robot's sensor data: 2D LiDAR scan, velocity, and next waypoint location. Action space A consists of linear and angular velocity commands. The reward function (detailed later) penalizes collisions and encourages speed, with the total value being the discounted sum over the path.

Algorithm

Our approach is designed for AC algorithms with an existing algorithm as the expert. We use ROS's *move_base* framework [31] and DWA/TEB with fixed parameters as the expert. Sensor readings are fed to the RL algorithm and *move_base*. Each step generates data from both sources (see Fig. 3), which is recorded and used to train the AC algorithm.

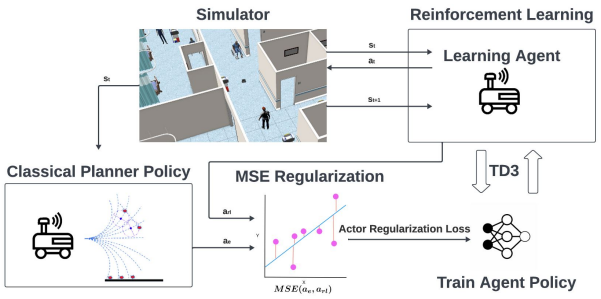

Fig. 3: Framework of the learning strategy with a classical planner regularization.

Our method, **TD3+OE** (RL+OE) (Algorithm 1), builds on TD3+BC with three underlined changes. **(i)** We add an online–expert consistency term to the actor loss (Eq. 3), yielding the additional gradient $\nabla_\phi \|\pi_\phi(s) - \pi_{\text{expert}}(s)\|^2$, where π_{expert} is queried online rather than imitated from a fixed dataset. **(ii)** We replace the regularizer λ with a clipped form $\tilde{\lambda} = \min\{\lambda, \lambda_0\}$, which enables the accompanying theoretical guarantees. This additional term can be viewed as defining a (soft) trust region around the classical policy. We use $\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)\in\mathcal{B}} \left| Q_\theta(s_i,a_i) \right|}$ with $\alpha = 2.5$ as in [23], and $\lambda_0 = \frac{2\,\varepsilon_{\max}}{g_{\max}}$. Our loss is then:

$$\mathrm{L}_{\text{actor}} = \mathbb{E}_{s\sim\mathcal{D}}\Big[\underbrace{\tilde{\lambda}\, Q_\theta\big(s, \pi_\phi(s)\big)}_{\text{RL term}} - \underbrace{\big\|\pi_\phi(s) - \pi_{\text{expert}}(s)\big\|_2^2}_{\text{imitation penalty}}\Big] \quad (4)$$

(iii) We add a hinge penalty to the critic loss to cap the action–gradient at a user–chosen target $g_{\max}$:

$$\mathcal{L}_{\text{GP}} = \beta_{\text{gp}}\, \mathbb{E}_{(s,a)\sim\mathcal{B}}\Big([\, \|\nabla_a Q_\theta(s,a)\|_2 - g_{\max}\,]_+ \Big)^2, \quad (5)$$

$[x]_+ \equiv \max(x, 0)$. The full critic objective is then

$$\mathcal{L}_{\text{critic}} = \mathcal{L}_{\text{TD}} + \mathcal{L}_{\text{GP}}, \quad (6)$$

where $\mathcal{L}_{\text{TD}}$ is the standard TD3 temporal-difference loss and $\beta_{\text{gp}} > 0$ is a small weight.

We seed the replay buffer with classical-planner rollouts perturbed by Gaussian noise, keeping exploration near trusted behavior—appropriate for navigation, where useful actions lie in a narrow band around the global path and random exploration is ineffective.

Theoretical Guarantee

To characterise how closely the learned policy π_ϕ stays to the classical planner π_{expert}, we adapt the analysis of [23].

Gradient-Scale Normalisation via Q-Value Averaging: TD3+BC introduces the normaliser λ to balance the RL and BC gradients automatically. With actions clipped to $[-a_{max}, a_{max}]$, the BC term is bounded (at most $4\cdot a_{max}$ per dimension) whereas the magnitude of Q_θ — and therefore its

gradient — scales with the environment's reward. Dividing by the mini-batch mean absolute Q places both terms on comparable scales, so the single, dimensionless parameter α becomes a task-independent "knob" that governs the trade-off between reward maximization and imitation.

Capped Adaptive Weight and Resulting Deviation Guarantee: To obtain an *explicit* deviation guarantee, we cap the adaptive weight by a constant $\lambda_0 > 0$,

$$\tilde{\lambda} \;=\; \min\{\lambda, \lambda_0\},$$

and optimise (4) with λ replaced by $\tilde{\lambda}$. The cap prevents pathological batches with small $\langle|Q|\rangle$ from inflating the RL term and enlarging the trust region. Because $\tilde{\lambda} \le \lambda_0$, we recover a fixed safety radius while still reaping the automatic scale normalisation whenever $\lambda < \lambda_0$.

Theorem 1 (Planner-deviation bound)**.** *Assume the critic is L-Lipschitz in its action argument, i.e.* $\|\nabla_a Q_\theta(s,a)\| \le L$ *for all* (s,a)*. Let* $\pi_{\phi^\star}$ *be any (local) maximiser of the capped loss. Then for every state* s

$$\big\|\pi_{\phi^\star}(s) - \pi_{\mathit{expert}}(s)\big\| \;\le\; \frac{\tilde{\lambda}\,L}{2} \;\le\; \frac{\lambda_0 L}{2}.$$

Proof: Following [23], stationary points satisfy $\tilde{\lambda} \cdot \nabla_a Q_\theta\big(s, \pi_{\phi^\star}(s)\big) = \big(\pi_{\text{expert}}(s) - \pi_{\phi^\star}(s)\big)$. Taking norms and applying the Lipschitz bound yields $\|\pi_{\phi^\star}(s) - \pi_{\text{expert}}(s)\| \le \frac{\tilde{\lambda} L}{2}$, and the cap gives the second inequality.

Enforcing the Lipschitz cap via a hinge penalty: To turn the bound of Theorem 1 into a *user-chosen and differentiable constraint*, we augment the critic loss with a **hinge penalty** that drives the action–gradient below a target value $g_{\max}$. For every transition (s,a) in the mini-batch we compute $g(s,a) = \nabla_a Q_\theta(s,a)$ and add

$$\mathcal{L}_{\text{GP}} \;=\; \beta_{\text{gp}} \Big[\, \|g(s,a)\| - g_{\max} \Big]_+^2, \tag{11}$$

where $[\cdot]_+ = \max(\cdot, 0)$ and $\beta_{\text{gp}} \in [10^{-3}, 10^{-2}]$ is a small weighting. During training, this term is back-propagated together with the TD error. At convergence, the optimal critic satisfies $\|\nabla_a Q_\theta\| \le g_{\max}$ almost everywhere, so that the Lipschitz constant obeys $L \le g_{\max}$.

Given a desired safety radius $\varepsilon_{\max}$ (e.g. 50% of the $[-a_{\max}, a_{\max}]$ range) we choose

$$\lambda_0 \;=\; \frac{2\,\varepsilon_{\max}}{g_{\max}}, \tag{12}$$

and cap the adaptive weight in the actor loss by $\tilde{\lambda} = \min\{\lambda, \lambda_0\}$. Combining (11)–(12) with Theorem 1 gives the *explicit guarantee*

$$\big\|\pi_{\phi^\star}(s) - \pi_{\text{expert}}(s)\big\| \;\le\; \varepsilon_{\max} \qquad \forall s \in \mathcal{D},$$

independently of the reward scale, because both $g_{\max}$ and $\varepsilon_{\max}$ are chosen directly by the practitioner.

Algorithm 1: TD3+OE (Online Expert) Algorithm

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ.
Initialize target networks: $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$.
Initialize replay buffer B by simulating the expert policy with Gaussian noise: $a = a_e + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma)$.
for $t = 1$ to T **do**
  Select action with exploration noise: $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$, and observe expert action a_e, reward r and new state s'.
  Store (s, a, a_e, r, s') in B and sample a mini-batch $\{(s_i, a_i, a_{e,i}, r_i, s_i')\}_{i=1}^N$.
  $\tilde{a}_i \leftarrow \pi_{\phi'}(s_i') + \epsilon$, with $\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$.
  $y_i \leftarrow r_i + \gamma \min_{j=1,2} Q_{\theta_j'}(s_i', \tilde{a}_i)$.
  Update critics: set

$$L_Q \leftarrow \frac{1}{N}\sum_{i=1}^{N}\Big[(y_i - Q_{\theta_1}(s_i,a_i))^2 + (y_i - Q_{\theta_2}(s_i,a_i))^2 + \beta_{\text{gp}}\big(\tfrac{1}{2}(\|\nabla_a Q_{\theta_1}(s_i,a)\|_2 + \|\nabla_a Q_{\theta_2}(s_i,a)\|_2) - g_{\max}\big)_+^2\Big]$$

  $\theta_1, \theta_2 \leftarrow \arg\min L_Q$.
  if $t \bmod d = 0$ **then**
    Update ϕ by the deterministic policy gradient:

$$\nabla_\phi J(\phi) = \frac{1}{N}\sum_{i=1}^{N} \tilde{\lambda}\,\nabla_a Q_{\theta_1}(s_i,a)\big|_{a=\pi_\phi(s_i)} \nabla_\phi \pi_\phi(s_i) - 2\big(\pi_\phi(s_i) - \pi_{\text{expert}}(s_i)\big)\cdot\nabla_\phi \pi_\phi(s_i)$$

    Update target networks:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$
$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$

  end if
end for

Implementation

We use a differential drive robot, the Rover-robotics Rover Zero robot, equipped with a 360^0 field of view planar laser scan (LiDAR). The robot uses the Robot Operating System *move_base* navigation stack with an A* global planner.

Every 0.5 s, we read the robot's LiDAR, the velocity vector v_t, and the current waypoint w_t (waypoints are spaced $\sim$ 3 m along the global path). LiDAR over a short temporal window is encoded with a 1D CNN followed by a GRU to produce a latent state h_t. The actor concatenates h_t with embeddings of v_t and w_t, passes the result through an MLP, and outputs the 2D action (x_t, y_t). The critic concatenates h_t with embeddings of v_t, w_t, and the action a_t, and uses twin value heads to predict Q. Complete hyperparameter settings are available in the repository.

Training is done in simulation only, and uses a challenging simulation framework built on Gazebo [33], shown on the top left of Figure 3. It features a realistic hospital environment [34], where people move according to the social forces model

[35], a technique widely used for simulating pedestrian movement. Social Forces captures human interaction and their adaptation to the perceived trajectory of the robot. Navigating this environment is particularly challenging due to narrow corridors and rapidly moving dynamic obstacles. Each training episode tasks the robot with navigating between two locations about 30 m apart, traversing rooms within the simulated hospital. The policies learned in simulation were transferred to the real robot. We then conducted experiments in a real building, shown in Figure 1.

Reward Function

'Designing a reward function that captures "good" behavior remains an open challenge' [36]. Regularizing the policy toward a trusted classical planner promotes alignment with well-understood behaviors, providing a stable inductive bias. Our reward function is designed to incentivize progress toward waypoints, minimize time taken, and ensure safety. The overall reward is defined as: $R(s,a) = R_t + R_w + R_c + R_s$.

$R_t = -0.1$ imposes a penalty for each step taken, $R_w = distance(w_t) - distance(w_{t-1})$ rewards progress toward the recommended global path by measuring the distance to the current waypoint, and $R_c = \min(\mathrm{d_{collision}} - \mathrm{d_{c\_offset}}, 0)$ is a safety penalty. $d_{\text{collision}}$ represents the distance to the nearest object. The term $d_{\text{c\_offset}}$ (default value 0.5) is an offset added to encourage the vehicle to maintain a safe distance from surrounding objects. $R_s = -2$ if $\mathrm{distance}(w_t) - \mathrm{distance}(w_{t-1}) < 0.01$ and $\mathrm{d_{collision}} < 0.15$; otherwise $R_s = 0$. This term penalizes getting stuck,i.e., lingering at very low velocity while near obstacles.

EXPERIMENTS

Our experiments compare our method against policies closely aligned with the classical algorithm. In simulation, we use objective metrics to measure performance, while in the real world, we assess the robot's performance subjectively via a user study.

Setup: In simulation, the robot navigates between rooms, with static obstacles, such as walls and furniture, and dynamic obstacles, in the form of walking and wheelchaired people crossing doors and corridors, implemented using Social Forces. The simulation was accelerated during training. The algorithms were trained until convergence for about 10^5 steps (about 6 hours and 1000 episodes) on a single computer with an AMD Ryzen 7 5800X CPU and GeForce RTX 4070 GPU. Robot sensors for LiDAR and Odometry used were: Slamtec RPLiDAR S3 and Realsense T265.

In real-world experiments, the robot navigated through different places in a building. It was recorded as it traversed a corridor, crossed a door, avoided static obstacles and moving people. We recorded videos and showed them to 20 people for evaluation.

Baselines and Metrics: We apply our method to two classical planners, DWA and TEB, with fixed parameters. We compare the original algorithms to our method applied to DWA and to TEB as the online expert, to a parameter-tuning algorithm (APPLR), the residual RL method [32], and a pure RL approach. The residual RL policy augments a fixed expert policy π_e with a bounded stochastic residual policy π_r ensuring that deviations from the expert policy are small by construction. Following [32]: a Gaussian residual, conditioned on sensors and the base action, is added to the base policy. For state s, the executed action is

$$a(s) \;=\; \mathrm{clip}(\pi_E(s) + \pi_R(s),\, -a_{max},\, a_{max})\,, \qquad (7)$$

with a squashed Gaussian residual of radius $\rho = \varepsilon_{\max}$:

$$u \sim \mathcal{N}\big(\mu_\theta(s),\, \Sigma_\theta(s)\big), \qquad (8)$$

$$\pi_R(s) = \rho\, \tanh\big(u\big), \qquad \|\pi_R(s)\|_\infty \le \rho. \qquad (9)$$

Because clipping is non-expansive, this gives a simple approach for guaranteeing bounded deviation from the expert:

$$\begin{aligned}\|a(s) - \pi_E(s)\|_\infty &= \big\|\mathrm{clip}(\pi_E(s) + \pi_R(s)) - \pi_E(s)\big\|_\infty \\ &\le \|\pi_R(s)\|_\infty \le \rho.\end{aligned} \qquad (10)$$

Training follows a standard AC objective on the composed action $\pi_E(s) + \pi_R(s)$; with a Q-critic Q_ϕ we optimize

$$\max_\theta\; \mathbb{E}_s\big[\mathbb{E}_{\pi_R\sim\pi_R(\cdot|s)} Q_\phi\big(s,\, \pi_E(s) + \pi_R\big)\,\big]\,. \qquad (11)$$

In simulation, we evaluated agents based on five criteria: (1) time, (2) proximity risk, (3) similarity to the expert, (4) getting stuck, and (5) aborted episodes. **Time** is measured as the average traversal time, or how long the robot takes to complete an episode. This measures efficiency; a good planner should reach the goal quickly. **Proximity Risk** is assessed by calculating the percentage of time the robot is in a "safety-critical situation," which is defined as being within 50 cm of an obstacle. This metric measures safety by avoiding near-collisions. **Similarity** is determined by calculating the Root Mean Squared Error (RMSE) between the velocity commands of the agent and the baseline local planner (DWA/TEB). This evaluates whether the robot behaves in a way that aligns with the classical algorithm. **Aborted Episodes** denote trials terminated early when the *global planner* could no longer recover a feasible plan and the run was stopped for safety. For the real-world experiments, we used a questionnaire (see supplementary material) inspired by [37] and [38]. It uses a 5-point Likert scale to measure important perceived aspects of robot motion, namely: (1) safety, (2) comfort, (3) predictability, (4) social compliance, (5) appropriateness of speed, and (6) overall satisfaction.

Results

Simulation: The results over 50 simulation episodes are shown for DWA and TEB, APPLR, RL, RL+OE (ours) and Residual in Table I, with the 95% confidence interval. Signals were collected at intervals of 0.5 seconds, including the distance to the nearest obstacle, robot velocity, and velocity commands. Additional details appear in Figures 4-7.

Our reward encourages *speed*, *collision avoidance*, and *avoiding getting stuck*. In practice, however, it is difficult to balance these objectives perfectly with a single scalar reward.

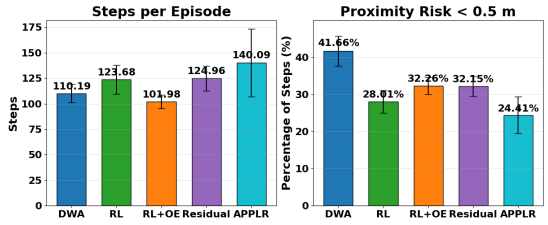

Fig. 4: DWA, RL, RL+OE, Residual, APPLR. **Steps per Episode and and Proximity Risk (lower is better)**. RL+OE achieves a low traversal time and proximity risk.

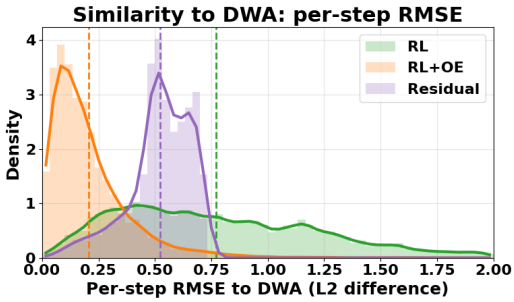

Fig. 5: Similarity tests compared to DWA (lower is more similar) RL+OE is significantly more similar to DWA.

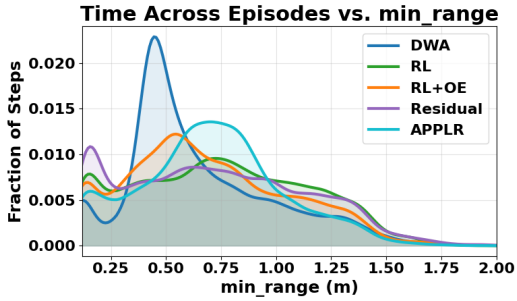

Fig. 6: Distance Histogram. Proportion of time robot spends at different distances from nearest obstacle. The more shifted to the right a curve is, the further the robot is from obstacles.

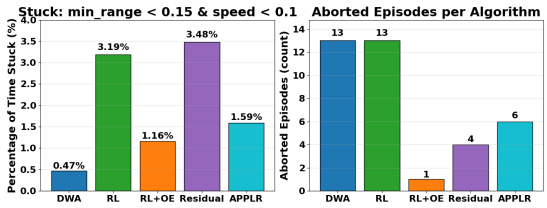

Fig. 7: Stuck Percentage. The proportion of time the robot freezes. **Aborted Episodes (out of 50)**. The amount of episodes aborted because no plan was available.

Steps per episode (Fig. 4) Only our method (**RL+OE**) is faster than the classical planner baseline, achieving $\sim 8\%$ fewer steps on average, though this improvement exhibits low statistical significance (p=0.15).[1]

Proximity risk (Fig. 4) All RL variants reduce the percentage of steps with min_range $< 0.5\,\mathrm{m}$ compared to the baseline. APPLR is the most conservative (about $\sim 43\%$ improvement), and our method improves this metric by $\sim 24\%$ relative to the classical planner; notably, the statistical significance of these improvements is high ($p < 0.01$).

Similarity to the planner (Fig. 5) Our method stays close to the classical planner as enforced by the theory. The *Residual* policy, while respecting the bound, too, has its density centered farther from the planner, i.e., it behaves more differently from the classical planner, without any benefits, performing worse on the measures of interest.

Distance profiles (Fig. 6) Across methods, the total-time curves shift to larger min_range values, indicating that learned policies tend to maintain more space from obstacles.

Stuck behavior (Fig. 7) Getting stuck leads to uncomfortable, hesitant motion around people. The classical planner (DWA/TEB) is strong on this metric, and our method closely matches this behavior, outperforming the other RL baselines.

Aborted Episodes (Fig. 7). Aborts occur when the global planner cannot recover, typically after (i) severe collisions that cause the robot to tip/flip, or (ii) entry into highly constrained spaces that preclude escape. Our method is the most cautious, avoiding these failure modes and yielding the lowest abort rate.

Takeaway. An unconstrained RL policy can degrade desirable navigation traits when a strong classical planner is available. Explicitly constraining an RL policy to remain similar to a strong planner improves several metrics while striking a better balance among them. Moreover, controlling *actions* themselves — not merely retuning navigation parameters — yields more predictable and well-behaved motion.

Results with TEB as the classical planner: We repeated the experiments using **TEB** as the baseline planner. See Table I. Compared to DWA, TEB requires more steps per episode, has a lower percentage of proximity events (min_range $< 0.5\,\mathrm{m}$), and a larger fraction of time stuck. Our **RL+OE** policy improves TEB on *all* reported metrics, and with high statistical significance: it reduces episode length, lowers proximity risk, decreases the stuck percentage, and number of aborted episodes. Across methods, RL+OE is generally among the best or second-best on each metric, while remaining close to the classical planner in the similarity curve, as designed. Because RL+OE explicitly stays near the (weaker) TEB policy, it sacrifices some raw performance that a fully unconstrained RL policy might extract; however, it yields a safer and more balanced navigation profile overall.

Real-World: We evaluated all methods on a physical robot with subjective assessments. For each algorithm, we filmed three scenarios—(1) hallway with a trash can and a person, (2) corridor with a person, and (3) doorway crossing—and asked 20 participants to rate the videos. Our setup exposes a clear sim-to-real gap: real doors and corridors are narrower than in simulation, yet our sim-trained policy transferred zero-shot and still performed strongly.

[1]P-values were calculated using the Welch's t-test on mean and variance.

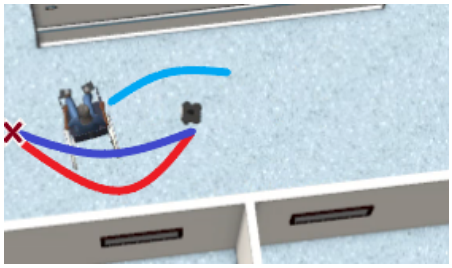

Fig. 8: **DWA**, **RL+OE**. RL+OE demonstrates greater awareness when encountering a person than DWA.

Fig. 9: **APPLR**, **RL+OE**. APPLR takes an exaggerated curve around a static obstacle, while RL+OE deviates less.

Results appear in Table II. Plain RL scored lowest, driven by abrupt accelerations or unpredictable maneuvers. Residual slightly improved the results, although it still showed similar undesired motions as RL. APPLR trailed DWA and RL+OE due to a sim-to-real gap: real doors were narrower than in simulation, and it struggled more to pass. DWA and RL+OE achieved the best, similar scores; however, DWA would likely degrade in more crowded settings because it tends to stop at pedestrians—acceptable for one person (like in the videos) but problematic in dense crowds. Overall, our method avoided unexpected motions, remained smooth and predictable, and reliably negotiated narrow passages.

Conclusion

Learning to optimize safety, efficiency, smoothness, and social compliance in mobile robotics is intrinsically difficult — improving one objective often degrades another. Our approach tackles this by regularizing RL toward a *trusted* classical planner: the learned policy is explicitly constrained to stay within a planner-defined trust region, for which we provide theoretical guarantees. This yields a practical balance across objectives rather than overfitting to any single metric.

Across simulation and real-world trials, these benefits are consistent. With dynamic obstacles (Fig. 8), DWA and APPLR underperform, reflecting limited temporal awareness; APPLR largely increases clearance by inflating the obstacle radius, which improves proximity but can be overly conservative near static geometry. In contrast, the RL-based methods — and especially ours — maintain appropriate spacing around people without retreating unnecessarily from static obstacles (Fig. 9).

In terms of motion quality, classical planners remain highly predictable: they favor lower accelerations and hew closely to the global path. Unconstrained RL, however, often exhibits abrupt accelerations and in-place rotations near obstacles — pathologies that are difficult to eliminate through reward shaping alone. Our method inherits the predictability of the planner while improving several metrics, avoiding the "weird" motions observed with plain RL, and aligning more closely with desirable local-planner behavior.

Another advantage of classical algorithms is that, because they typically rely on fundamental techniques, they can generalize well across static environments. This was demonstrated for DWA as well as RL+OE, which is designed to inherit many of its good properties.

Limitations

This work relied on a 2D LiDAR sensor, which may miss obstacles, particularly those below its mounting height. Integrating sensors like depth cameras or 3D LiDAR could improve obstacle detection and enhance the robot's real-world navigation.

References

[1] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone, "Appli: Adaptive planner parameter learning from interventions," in *ICRA*, 2021, pp. 6079–6085.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[3] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *ECC*. IEEE, 2015, pp. 3352–3357.

[4] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IEEE/RSJ*, 2017, pp. 31–36.

[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," vol. abs/1604.07316, 2016.

[6] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[7] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," vol. 35, no. 11, p. 1289–1307, sep 2016.

[8] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IEEE/RSJ*, 2017, pp. 1343–1350.

[9] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *IEEE/RSJ*, 2018, pp. 3052–3059.

[10] H. Karnan, G. Warnell, X. Xiao, and P. Stone, "Voila: Visual-observation-only imitation learning for autonomous navigation," in *2022 Int. Conf. on Robotics and Automation (ICRA)*, 2022, pp. 2497–2503.

[11] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018, pp. 1111–1117.

[12] C.-Y. Tsai, H. Nisar, and Y.-C. Hu, "Mapless lidar navigation control of wheeled mobile robots based on deep imitation learning," *IEEE Access*, vol. 9, pp. 117 527–117 541, 2021.

[13] X. Xiao, Z. Wang, Z. Xu, B. Liu, G. Warnell, G. Dhamankar, A. Nair, and P. Stone, "Appld: Adaptive planner parameter learning," vol. 154, p. 104132, 2022.

TABLE I: DWA (top) and TEB (bottom) results (mean ±95% CI). Deltas Δ and p-values (p) are relative to the classical baseline, representing the percentage in mean difference ($mean_{method} - mean_{baseline}$) for the **Steps** and **Proximity Risk** metrics. Negative values in deltas indicate improvement.

Method	Steps		Proximity Risk (%)		Stuck (%)	Aborted	Similarity RMSE
	mean ±CI	Δ (p)	mean ±CI	Δ (p)	perc.	count	mean ±CI
Baseline (DWA)	110.19 (101, 119)	—	41.66 (37, 45)	—	0.47	13	0.00 (–)
RL	123.68 (109, 138)	$+12\%$ ($p = 0.12$)	28.01 (25, 31)	-33% ($p < 0.01$)	3.19	13	0.93 (0.90, 0.96)
RL+OE (ours)	101.98 (95, 109)	-8% ($p = 0.15$)	32.26 (30, 35)	-24% ($p < 0.01$)	1.16	1	0.26 (0.24, 0.27)
Residual	124.96 (112, 137)	$+14\%$ ($p = 0.06$)	32.15 (29, 35)	-24% ($p < 0.01$)	3.48	4	0.54 (0.53, 0.54)
APPLR	140.09 (106, 173)	$+27\%$ ($p = 0.09$)	24.41 (19, 29)	-43% ($p < 0.01$)	1.59	6	0.00 (–)

Method	Steps		Proximity Risk (%)		Stuck (%)	Aborted	Similarity RMSE
	mean ±CI	Δ (p)	mean ±CI	Δ (p)	perc.	count	mean ±CI
Baseline (TEB)	360.95 (314, 408)	—	22.89 (20, 26)	—	1.82	9	0.00 (–)
RL	131.05 (116, 146)	-64% ($p < 0.01$)	29.85 (25, 34)	$+30\%$ ($p = 0.01$)	2.36	7	1.10 (1.03, 1.16)
RL+OE (ours)	206.00 (181, 230)	-43% ($p < 0.01$)	18.90 (17, 21)	-17% ($p = 0.03$)	1.30	4	0.52 (0.49, 0.55)
Residual	220.40 (177, 263)	-39% ($p < 0.01$)	31.55 (27, 37)	$+39\%$ ($p < 0.01$)	3.27	3	0.55 (0.54, 0.55)
APPLR	260.91 (207, 315)	-28% ($p < 0.01$)	23.11 (20, 26)	$+0\%$ ($p = 0.92$)	1.84	7	0.00 (–)

TABLE II: Subjective Mean (95% CI) Opinion Scores (MOS; 1=worst, 5=best).

Metric	Baseline (DWA)	RL	Residual	APPLR	RL+OE (ours)
Perceived Safety	3.74 (3.39–4.09)	2.26 (1.96–2.56)	2.85 (2.50–3.21)	3.30 (2.89–3.71)	3.57 (3.28–3.87)
Comfort / Smoothness	3.43 (3.13–3.72)	2.09 (1.85–2.34)	3.06 (2.71–3.40)	3.33 (2.97–3.70)	3.44 (3.20–3.69)
Predictability / Legibility	3.48 (3.12–3.84)	1.72 (1.48–1.97)	3.26 (2.93–3.59)	3.20 (2.84–3.56)	3.31 (3.05–3.58)
Social Compliance	3.61 (3.29–3.94)	2.28 (2.02–2.54)	2.98 (2.67–3.29)	3.30 (2.90–3.70)	3.70 (3.43–3.98)
Appropriateness of Speed	3.61 (3.29–3.93)	2.63 (2.34–2.92)	2.98 (2.65–3.31)	3.30 (2.92–3.68)	3.39 (3.16–3.62)
Overall Satisfaction	3.61 (3.34–3.88)	2.17 (1.95–2.38)	3.11 (2.81–3.41)	3.33 (2.94–3.73)	3.69 (3.47–3.90)

[14] Z. Wang, X. Xiao, B. Liu, G. Warnell, and P. Stone, "APPLI: adaptive planner parameter learning from interventions," vol. abs/2011.00400, 2020.

[15] Z. Wang, X. Xiao, G. Warnell, and P. Stone, "Apple: Adaptive planner parameter learning from evaluative feedback," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7744–7749, 2021.

[16] Z. Xu, G. Dhamankar, A. Nair, X. Xiao, G. Warnell, B. Liu, Z. Wang, and P. Stone, "APPLR: adaptive planner parameter learning from reinforcement," vol. abs/2011.00397, 2020.

[17] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021, pp. 6057–6063.

[18] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proceedings of the 34th Int. Conf. on Machine Learning*, vol. 70, 2017, pp. 1352–1361.

[19] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, *et al.*, "Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios," in *2023 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 7553–7560.

[20] E. Bronstein, S. Srinivasan, S. Paul, A. Sinha, M. O'Kelly, P. Nikdel, and S. Whiteson, "Embedding synthetic off-policy experience for autonomous driving via zero-shot curricula," in *Conference on Robot Learning*. PMLR, 2023, pp. 188–198.

[21] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of a* pathfinding," *Procedia Computer Science*, vol. 179, pp. 507–514, 2021.

[22] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," vol. abs/1802.09477, 2018.

[23] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.

[24] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *ICRA*, 2019, pp. 6015–6022.

[25] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, no. 5, pp. 569–597, 2022.

[26] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[27] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[28] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[29] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[30] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, 01 2009.

[32] X. Hou, J. Zhang, C. He, Y. Ji, J. Zhang, and J. Han, "Autonomous driving at the handling limit using residual reinforcement learning," *Advanced Engineering Informatics*, vol. 54, p. 101754, 2022.

[33] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ*, vol. 3, 2004, pp. 2149–2154 vol.3.

[34] AWS RoboMaker, "AWS RoboMaker Hospital World ROS package," GitHub repository, jul 2021.

[35] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," vol. 51, pp. 4282–4286, May 1995.

[36] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis)design for autonomous driving," vol. abs/2104.13906, 2021.

[37] C. Bartneck, D. Kulić, E. Croft, and S. Zoghbi, "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots," *Int. Jour. of social robotics*, vol. 1, pp. 71–81, 2009.

[38] E. Repiso, A. Garrell, and A. Sanfeliu, "Real-life experiment metrics for evaluating human-robot collaborative navigation tasks," in *2023 32nd IEEE Int. Conf. on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2023, pp. 660–667.