International Journal of Robotics and Control Systems
Vol. 5, No. 5, 2025, pp. 2414-2431
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Simulation-Based Validation of a TD3–RRT Hybrid Learning Framework for Safe and Adaptive Robot Navigation

Hamza Ben Roummane [a,1], Cherki Daoui [b,2]

[a,b] Department of Information Processing and Decision Laboratory, Sultan Moulay Slimane University, Beni Mellal, Morocco
[1] hamza.benroummane@usms.ac.ma; [2] daouic@yahoo.com
*Corresponding Author

ARTICLE INFO

Article history
Received August 05, 2025
Revised October 12, 2025
Accepted November 17, 2025

Keywords
Autonomous Robots;
Hybrid Navigation;
TD3;
RRT;
Obstacle Avoidance

ABSTRACT

Autonomous navigation in dynamic environments remains a major challenge due to real-time constraints and unpredictable obstacles. Unlike prior work that relies solely on local planning or suffers from unstable convergence, our method aims to ensure safe and efficient navigation for autonomous robots in dynamic environments through a more stable architecture. The research contribution is a TD3+RRT hybrid architecture enhanced with transfer learning, enabling rapid adaptation in dynamic environments while maintaining trajectory feasibility. The TD3 agent is first trained in a static ROS–Gazebo environment and then transferred to a larger, dynamic setting. RRT operates in parallel to generate feasible global paths, while TD3 handles real-time obstacle avoidance. The state space includes LIDAR data and relative goal positioning, and the action space produces continuous velocity commands. The hybrid model achieved a 92% success rate, a 50% reduction in collision rate, and the highest cumulative reward compared to baseline TD3, DDPG, and SAC methods. Trajectory analysis further confirmed smoother and more consistent paths in dynamic scenarios. These results highlight the effectiveness of combining model-based and model-free strategies for reliable autonomous navigation, offering a promising step toward real-world deployment.

1. Introduction

Mobile robots are entering our hospitals, warehouses, and campuses. They must move safely, on time, and without disrupting users. Real-world environments are dynamic, obstacles move, plans change, sensors are imperfect, and Navigation. Therefore, becomes critical to the reliability and acceptance of these systems [1].

Traditional planners provide overall guarantees. A*, Dijkstra, and RRT produce feasible paths in known maps [2]. They are useful for long-term planning. However, they often assume quasi-static scenes [3]. In the presence of moving obstacles, they require frequent replanning. These costs computing time. It degrades responsiveness. They can also fall into local dead ends [4], [5].

Deep reinforcement learning (DRL) offers the other side of the coin [6]. It learns to avoid obstacles in real time using sensors [7]. It handles non-linearity and uncertainty. It reduces dependence

on precise models. Thus, when faced with rapid changes, DRL reacts better than purely planned approaches [8]. However, it has its limitations. Policies may converge slowly or in an unstable manner. Generalization to new scenarios is fragile. Without a global vision, DRL can produce inefficient or risky trajectories [9], [10].

There is a twofold need. The overall feasibility of the route must be maintained. Rapid local adaptation must also be guaranteed. In other words, plan far ahead and react locally [11]. Integration is necessary. Planners ensure structure and safety along the route [3]. The DRL provides agility in response to immediate events. This complementarity is central in a dynamic environment [12].

This work fills this gap with a hybrid strategy that combines a global planner with a local learning-based controller. We utilize RRT to maintain a feasible global route and TD3 to avoid obstacles in real-time, based on onboard sensors. To reduce training costs and improve stability, we leverage transfer learning, where the TD3 agent is first trained in a simpler static scenario and then adapted to a larger, dynamic environment. The design targets practical ROS-Gazebo deployments with continuous LIDAR observations and velocity commands, aligning the method with common robotics stacks. The goal is straightforward: to achieve safe, efficient, and robust navigation in real-world, dynamic environments. The contribution of this research is:

(1) We introduce a hybrid navigation architecture that combines RRT for global feasibility with TD3 for reactive local adaptation in dynamic environments.
(2) We integrate transfer learning to accelerate convergence and improve policy robustness when scaling from static to dynamic scenarios.
(3) We provide a comprehensive evaluation in ROS–Gazebo showing superior performance over baselines.

2. Related Work

The application of DRL to autonomous mobile robot navigation is attracting growing interest in the scientific community [7], [10]. Many researchers have proposed solutions to overcome challenges related to slow convergence, the lack of effective global planning, and the lack of adaptability to dynamic environments [8]. However, most existing hybrid approaches suffer from weak coordination, limited adaptability to dynamic obstacles, or a lack of scalability [13].

2.1. Hybrid DRL Classical Planning

Recent works have proposed hybrid models that integrate DRL with traditional planners to leverage their complementary strengths. For instance, Singh et al. [14] used A* for global planning and SAC for local obstacle avoidance in dense human environments. While effective in static maps, their architecture relies on offline A* routes that cannot adapt to environmental changes. In addition, the decoupling between the global and local layers often leads to inconsistent trajectories when dynamic obstacles force local deviations.

Liu et al. [15] proposed a TD3+DWA system in which TD3 dynamically adjusts the sampling window of DWA. This hybrid improves short-term responsiveness, but DWA's reliance on a fixed sensing range and local optimization horizon limits global adaptability. Similarly, Gao et al. [16] combined PRM with TD3 to maintain a feasible path while learning reactive control. Yet PRM graphs are computed offline and do not handle updates effectively. Moreover, these models often fail to re-synchronize the planner and policy when the environment changes suddenly, leading to outdated plans and increased failure rates.

Sharma et al. [17] proposed a Hybrid Classical/RL Local Planner that dynamically switches between a traditional rule-based controller and a deep reinforcement learning policy according to the complexity of the local environment. Their system improves navigation efficiency and reduces collisions in semi-structured spaces. However, the switching mechanism introduces latency and may cause instability when frequent transitions occur between the classical and learned planners, limiting its robustness in highly dynamic settings.

Dey et al. [18] presented Learning Whom to Trust in Navigation, an adaptive architecture that decides when to rely on a classical planner or a neural policy. The approach enhances decision-making in uncertain environments and enables flexible behavior selection. Nevertheless, the framework focuses mainly on discrete environment types and lacks continuous synchronization between the global and local levels, resulting in inconsistent long-term trajectories under dynamic changes.

These studies highlight the challenge of hybrid coordination: most frameworks use global planners in a static or loosely coupled manner. They fail to update plans incrementally or to propagate local corrections globally. Our method addresses these shortcomings by employing RRT as an incremental planner better suited for dynamic replanning, and by tightly integrating it with TD3 to maintain consistency across planning levels during execution.

2.2. Enhancing DRL with Transfer Learning, Memory, and Replay Techniques

Beyond architectural combinations, researchers have explored various techniques to improve DRL's stability and convergence. Liu et al. [19] combined SAC with imitation learning and Prioritized Experience Replay (PER) to accelerate learning. Li et al. [20] introduced a TD3 variant enhanced with PER and transfer learning, improving policy learning in static simulations. However, these methods are limited to 2D or grid-based tasks and rarely generalize to realistic dynamic environments.

Other studies have introduced memory modules like LSTM to capture temporal patterns in navigation. Huang et al. [21] used LSTM in a TD3 agent, improving trajectory smoothness, while Park et al. [22] applied Hindsight Experience Replay (HER) to better exploit failed trials in sparse-reward settings. Although these techniques reduce training time and improve decision stability, they do not address the need for long-horizon planning or real-time adaptation to moving obstacles. Their contributions remain local in scope and ignore the global structure of the navigation task. Ibarz et al. [23] reviewed the process of transferring reinforcement learning models from simulation to the real world. Their approach involved simulation training, followed by the application of techniques such as domain randomization, visual adaptation, and limited fine-tuning directly on the robot. Despite these efforts, they reported persistent failures caused by the reality gap, where a policy that performs well in simulation loses effectiveness once deployed on real hardware. Their studies mainly focused on robotic manipulation and legged locomotion. In contrast, our work addresses long-term mobile robot navigation in dynamic environments.

2.3. Positioning of Our Contribution

This work builds upon the above research while directly addressing its core limitations. We propose a hybrid architecture that combines RRT as an incremental global planner and TD3 as a local adaptive controller, trained with transfer learning to accelerate convergence. Unlike previous methods, we ensure dynamic coordination between the planning and learning layers, allowing the agent to revise its path in response to real-time changes. Our implementation is validated in realistic ROS–Gazebo simulations with dynamic obstacles.

3. Method

3.1. Approach Overview

Robots operating in dynamic environments must balance two key goals: maintaining a feasible path in the long term and responding promptly to local changes. Our approach addresses this challenge through a hybrid navigation framework that combines a global planner with a local controller.

The global planner relies on the RRT [24]. It generates an initial path from the robot's starting point to the destination, ensuring that the route remains valid and avoids static obstacles. For local control, we adopt the TD3 [25] algorithm. Unlike the global planner, this controller operates in real time, adjusting the robot's trajectory to handle moving obstacles and noisy sensor readings. To enhance training efficiency and strengthen the learned policy, we apply transfer learning. The TD3

model is first trained in a static environment, then refined in more complex and dynamic scenarios [26].

The following subsections describe each part of the architecture. We discuss the choice of RRT, the design and training of the TD3 agent, and the integration of these components within the ROS Gazebo simulation framework.

3.2. Global Planner RRT

The global planner in our framework relies on the RRT algorithm. RRT is a sampling-based motion planning method that incrementally builds a tree from the start position toward the goal. Its ability to efficiently explore high-dimensional, non-convex, and partially unknown spaces makes it well-suited for mobile robot navigation in complex environments [27].

In contrast to classical planners such as A*, Dijkstra, or the Probabilistic Roadmap (PRM), RRT does not depend on complete and static maps [28]. A* is optimal in static scenarios but performs poorly in dynamic scenes because it relies on fixed heuristics [29]. PRM requires a precomputed roadmap, which limits reactivity [30]. RRT, by comparison, adapts more naturally to dynamic or unknown environments by extending the tree incrementally. It supports online replanning, can reuse parts of the existing tree when obstacles appear, and remains computationally lightweight, which makes it attractive for real-time navigation [31].

In our implementation, the RRT planner operates on a global costmap that accounts for static and slowly changing obstacles. It takes as inputs the robot's current position, the global target, and a two-dimensional occupancy grid map provided by the simulated environment. The output is a sequence of waypoints forming a collision-free path toward the goal. These waypoints serve as intermediate targets that guide the local controller based on TD3.

To improve efficiency, the global plan is computed only once at the beginning of each episode. When the robot drifts too far from the planned route or when a new obstacle blocks the way, the planner updates the tree. In such cases, partial tree reuse is applied so that the planner extends or repairs the existing tree rather than restarting from scratch. This strategy improves responsiveness while keeping the computation cost low in changing environments.

3.3. Local Planner

For local navigation in dynamic environments, we use the TD3. This algorithm extends the actor–critic framework and has shown strong results in continuous control tasks [32]. Compared with the earlier DDPG [33] method, TD3 reduces overestimation bias, training instability, and poor sample use through three key mechanisms. First, twin Q-networks are used, and the lower of the two values is chosen to reduce overestimation. Second, policy updates are delayed, which means the actor network is updated less frequently than the critics, resulting in more stable training. Third, target policy smoothing is applied by adding small Gaussian noise to the target actions, which improves robustness [34].

Other reinforcement learning methods were considered. Proximal Policy Optimization (PPO), performs well in discrete action spaces but is less suited for continuous velocity commands [35], [36]. Soft Actor–Critic (SAC) ensures strong exploration through entropy, but needs higher computational resources [37], [38]. TD3 offers a good balance of stability, performance, and efficiency, which makes it appropriate for real-time local planning [20]. Fig. 1 shows the overall architecture of the TD3 local controller, including the policy and value networks, the replay memory, and their interaction with the simulated environment.

The performance of deep reinforcement learning algorithms in robotic navigation depends critically on the design of the state representation and action space [39]. In our framework, both were carefully defined to balance reactivity, simplicity, and relevance to the task. We model the local navigation problem as an MDP [40].

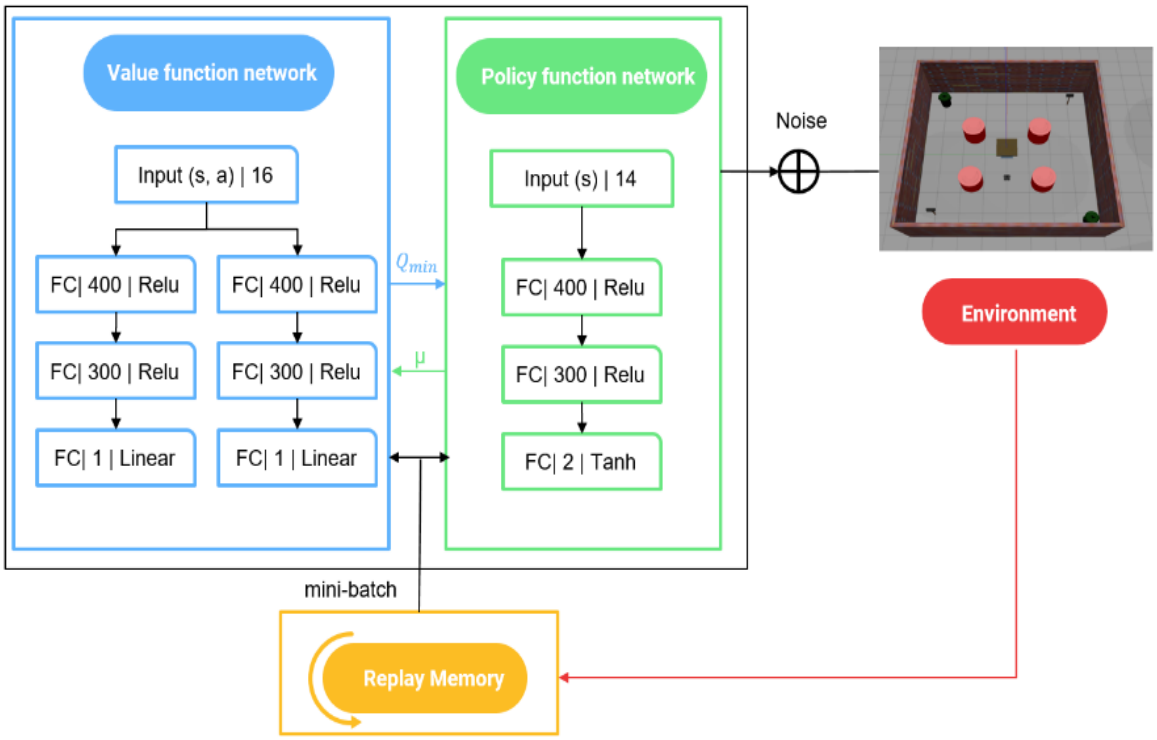

Fig. 1. TD3 architecture

- **State:** The state space includes 180-degree LiDAR scans that are filtered and normalized, along with the relative distance and angle to the next sub-goal provided by the global planner. Formally, the state s_t at time t is:

$$s_t = [\text{LiDAR}_{360}, d_{\text{subgoal}}, \theta_{\text{subgoal}}] \quad (1)$$

This representation ensures that the local controller is aware of both immediate surroundings and directional intent.

- **Action:** The output of the TD3 agent is a pair of continuous control commands:

Linear velocity $v \in [0, v_{\max}]$

Angular velocity $\omega \in [-\omega_{\max}, \omega_{\max}]$

These actions are passed directly to the ROS velocity command interface, and the sampling frequency is set to 10 Hz to align with LiDAR acquisition and simulation timestep.

- **Reward:** The reward function was designed to encourage safe and efficient behavior on the part of the robot. It takes several situations into account: reaching the target results in a highly positive reward as soon as the distance to the target falls below 0.2 m; any collision with an obstacle within 0.2 m results in a significant penalty; exceeding the time limit is also penalized. In addition, progress towards the target is rewarded with a reward proportional to the reduction in distance to the target. The general function can be expressed as follows:

$$R = \begin{cases} 100 & if\ reached\ the\ goal \\ -120 & if\ collision\ occurred \\ -120 & if\ time\ exceeded \\ \xi(d_{t-1} - dt) & otherwise \end{cases}$$

where ξ =300 is an amplification coefficient and dt represents the change in distance to the goal.

During each episode, the robot selects an action based on the current state, generates a new state, and receives a reward depending on the situation encountered. The episode ends as soon as a success goal is achieved, a collision, or a timeout occurs.

The TD3 agent processes sensor data and the sub-goal from the RRT planner in real time. The outputs are continuous velocity commands, which are sent to the ROS controller. In this way, the local controller reacts to dynamic obstacles while still following the global route defined by RRT. If the robot deviates too far from the planned path, the system triggers a new planning step to keep the global and local objectives aligned.

3.4. Transfer Learning Strategy

Deep reinforcement learning often requires millions of interactions before a model converges [41]. This makes training slow and expensive, especially when working in simulators such as ROS Gazebo where physics must be computed at every step. To overcome this limitation, we apply transfer learning so that knowledge gained in simpler scenarios can be reused when the environment becomes more complex [42].

Transfer learning has already played an important role in robotics. It allows researchers to reduce the number of samples needed, improve stability, and speed up training. More importantly, it helps the learned policy adapt to tasks that share similar dynamics without having to start from zero each time [43].

In our work, the strategy follows two stages. During the first stage, the TD3 agent is trained in a static environment where there are no moving obstacles. This early phase gives the robot the chance to learn basic navigation skills such as moving safely toward a goal and avoiding fixed obstacles. In the second stage, the model is moved into a dynamic setting where obstacles change position. The weights obtained from the static environment are used as the starting point for both the actor and the critic networks.

Only the essential knowledge is transferred. The network architecture remains the same, and the weights and biases are reused to guide the agent in the new environment. The replay buffer, however, is not kept, since using past experiences from the static environment could bias the policy. A new buffer is created so that the agent can collect fresh interactions adapted to dynamic conditions.

3.5. The Hybrid Architecture

This section presents the operational pipeline and the communication mechanisms that coordinate the modules of our hybrid navigation system. The design is modular, fully implemented under ROS, and relies on inter-process communication through topics and synchronized nodes. The execution pipeline contains five main modules. The first is the sensor node, which publishes real-time LiDAR scans. The raw data are processed through filtering and normalization to provide reliable inputs for navigation. The second module is the global planner. It is activated at the start of each episode and whenever the robot deviates significantly from the planned route or encounters a new obstacle. The planner computes a global path using the RRT algorithm on a two-dimensional costmap and outputs a sequence of sub-goals. These sub-goals are then published as a topic to guide the robot.

The third module is the sub-goal manager. It continuously monitors the robot's position and selects the next reachable target from the list provided by the planner. It also publishes the relative distance and orientation to this target. The fourth module is the local controller, which integrates the TD3 agent. This controller receives the preprocessed LiDAR data together with the sub-goal information and produces continuous velocity commands. These commands are sent to the velocity controller via the standard ROS topic for motion commands. Finally, the fifth module is the replanning monitor. It checks whether the robot deviates too much from its path or if the environment has changed, and when necessary, it requests partial or full replanning through a service call to the planner. Fig. 2 presents the workflow of the proposed training process, detailing how the agent interacts with the environment, receives rewards, and resets after reaching the goal or encountering collisions.

The communication across modules relies on standard ROS primitives. Topics are used for continuous data exchange, such as LiDAR scans, velocity commands, and sub-goal updates. Services are employed for event-triggered replanning, while the TF library manages transformations between the robot and the map frames. For efficiency, nodelets are used in modules where intra-process communication is frequent.

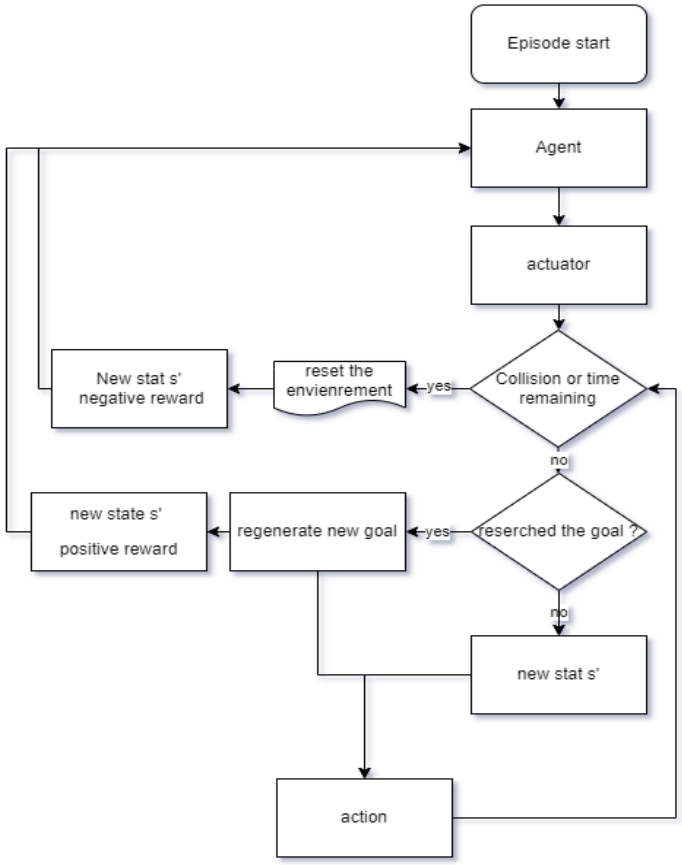

Fig. 2. Training flow of the TD3+RRT agent illustrating the interaction cycle between the agent, environment, and goal regeneration

4. Experiment

4.1. Simulation Setup and Platform

The experiments were carried out using ROS Noetic together with the Gazebo 11 simulator [44]. Gazebo provides a physics engine, realistic 3D rendering, and diverse obstacle scenarios, which make it a suitable platform for evaluating robot navigation in dynamic environments [45]. Training deep reinforcement learning agents directly on physical robots can be time-consuming and risky, especially during early exploration when collisions are frequent [46]. Simulation offers a safe and repeatable environment that accelerates development while avoiding hardware damage [47], [48]. Moreover, the modular nature of ROS makes it easy to integrate navigation components, and models trained in Gazebo have shown good transferability to real robots in prior studies. The node-based structure of ROS also facilitates communication through topics and services, which mirrors how the system would operate in real deployments [49]. The robot model selected for the tests is the TurtleBot3 Burger, equipped with a 180-degree LiDAR for obstacle detection and a front RGB camera for visual perception [50]. The robot's kinematics were limited to a maximum linear velocity of 0.5 m/s and an angular velocity between −2 and 2 rad/s. These limits reflect the real capabilities of the TurtleBot3 and help maintain realistic navigation behavior in simulation.

The TD3 model used in this study follows an actor–critic design. The actor network includes three fully connected layers: two hidden layers with 256 and 128 neurons activated by ReLU, and an output layer that generates continuous velocity commands. The critic network shares a similar structure and estimates the Q-value of each state–action pair. Training was performed with a learning rate of 0.0003 for both actor and critic, a discount factor γ of 0.99, a replay buffer of one million transitions, and a batch size of 256. Policy noise of 0.2 was added and clipped at 0.5. The policy was updated every two steps, and the target network update rate τ was set to 0.005.

To accelerate convergence, transfer learning was applied. The TD3 agent was first trained for 50,000 iterations in a static environment, then refined in a dynamic one using the pretrained weights. By contrast, the RRT global planner does not require any prior learning phase, since it is non-parametric. All evaluations are averaged over 5 runs, each with different random seeds to account for stochasticity. All simulations were executed on a laptop running Ubuntu 20.04, equipped with an Intel Core i7-1355U processor at 1.70 GHz and 32 GB of RAM.

4.2. Simulation Scenarios

Two distinct environments were developed to assess the robustness of the algorithms tested. The first is a static environment measuring (4 × 4 meters), featuring fixed obstacles; it was mainly used for the initial training of the TD3 algorithm and for collecting parameters for transfer learning. Fig. 3 presents the static environment.

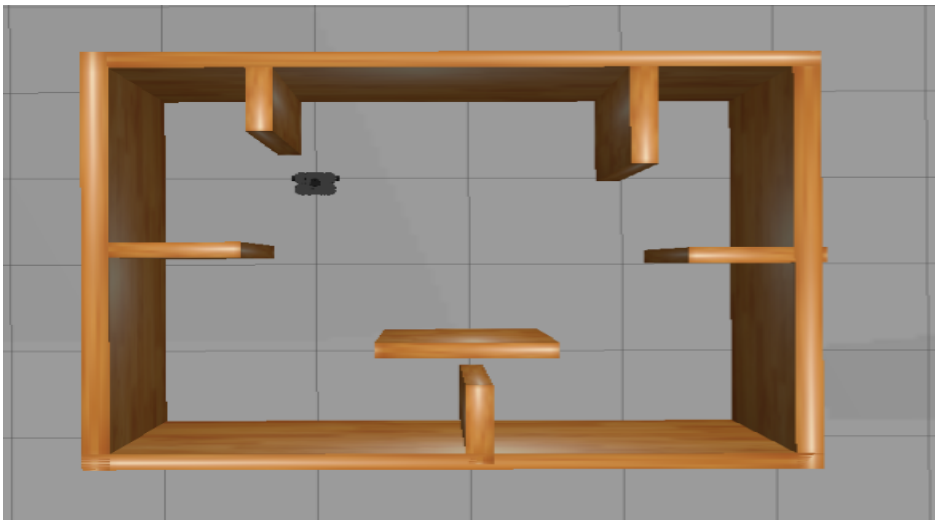

Fig. 3. Static environment

The second, larger course (8 × 8 metres) features the same fixed obstacles as the first, but also includes four rotating cylinders, simulating moving obstacles, as well as blue boxes that appear randomly to increase the complexity of the environment. Fig. 4 present the dynamic environment.

The diversity of these environments enables the testing of the proposed algorithm's adaptability under realistic and variable conditions, while maintaining a controlled experimental framework for comparative analysis.

4.3. Evaluation Metrics

In order to ensure the relevance and fairness of the comparison, we compared our TD3+RRT hybrid method with three other configurations: TD3 used alone, SAC, and DDPG. All algorithms were trained under the same experimental conditions, each over a total of 300,000 steps, to ensure a rigorous and consistent comparative evaluation. To assess and compare the performance of the different navigation strategies, a diverse set of quantitative and qualitative metrics was used. These indicators provide a complete view of how each method learns, plans, and behaves during navigation.

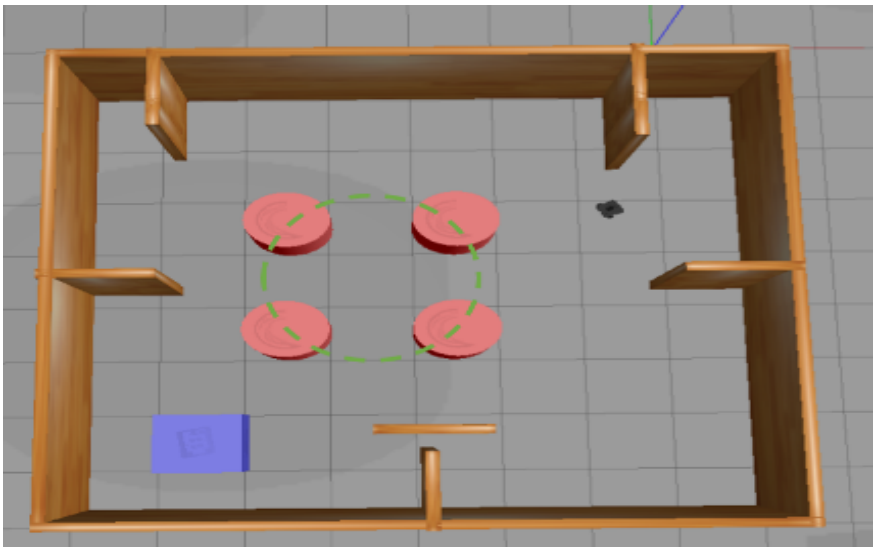

Fig. 4. Dynamic environment

The first group of metrics focuses on learning quality and stability. The cumulative reward measures the total reward gained in each episode and reflects how well the policy helps the robot reach its goal while avoiding penalties from collisions or timeouts. A higher cumulative reward indicates better performance. The reward variance, measured during the final part of training, shows how stable the learning process is. A lower variance means that the algorithm has reached consistent behavior. The mean squared error evaluates how accurately the value function is estimated over time, with lower values indicating better learning consistency. Convergence speed is also observed through the shape of the learning curves, where steady and early stabilization reflects efficient and reliable training.

The second group evaluates navigation success and safety. The success rate represents the percentage of episodes in which the robot reaches its goal within the allowed time without colliding with obstacles. In contrast, the collision rate measures how often the robot comes into contact with static or moving objects. A lower collision rate means safer and more reliable navigation.

A third group focuses on path efficiency and execution cost. The average path length indicates how far the robot travels to reach the goal in each successful run. Shorter paths suggest more direct and efficient trajectories. The computation time per episode measures how long the system takes to complete both planning and execution, showing how well the algorithm can perform in real time. Planning time specifically refers to the duration required by the global planner to generate a valid path, where shorter times are preferable for fast decision-making.

The final group concerns motion smoothness and trajectory quality. Smoothness is measured by analyzing variations in angular velocity throughout the movement. Lower values indicate fluid and natural motion, while large fluctuations suggest unstable or jerky trajectories. In addition to numerical evaluation, visual inspection of trajectories provides valuable qualitative insight. Observing how the robot moves, avoids obstacles, and follows the path helps confirm whether the behavior appears natural and consistent across different trials. Together, these metrics offer a balanced evaluation of learning performance, safety, efficiency, and motion quality, allowing a clear comparison of how each method behaves under dynamic navigation conditions.

5. Results

This section evaluates the proposed hybrid navigation strategy that combines TD3 and RRT. The goal is to measure how well it performs in different aspects, such as reward gain, collision avoidance, learning stability, and path quality. We compare our method with three common deep reinforcement learning algorithms, TD3, SAC, and DDPG, using the same training and testing conditions. To make

the results reliable, each experiment is repeated several times, and the averages are reported with confidence intervals and standard deviations. The evaluation includes both numerical results and visual analysis of trajectories in static and dynamic environments, giving a complete view of the system's behavior.

5.1. Learning Progression and Reward Acquisition

Fig. 5 presents the evolution of cumulative rewards for the four algorithms over 300,000 timesteps. The curves show the average performance, with shaded regions representing the standard deviation across five runs using different random seeds. The hybrid approach TD3 + RRT achieves the highest cumulative reward, reaching around 1850 ± 50 by the end of training. It also rises more quickly after about 100,000 timesteps, showing that the agent begins to learn effective behaviors earlier. The standard TD3 model follows with a final score near 1600 ± 70, while DDPG stabilizes around 980 ± 65, and SAC remains the lowest at roughly 570 ± 80. The early stages of training show greater variability for TD3 and DDPG, suggesting that these models are less stable during exploration. In contrast, SAC maintains more consistent but slower progress, while TD3 + RRT shows both steady growth and higher final performance.

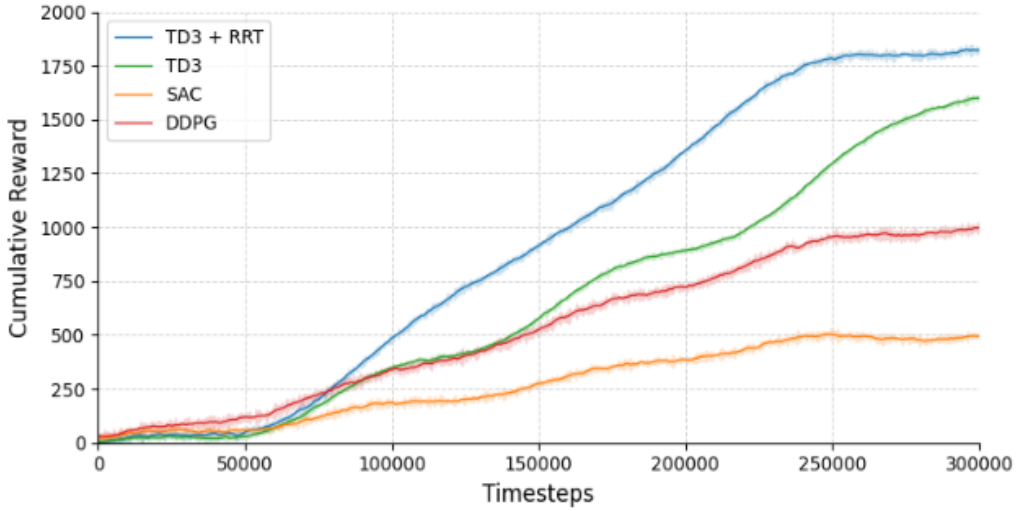

Fig. 5. Mean cumulative reward ± std for TD3 + RRT and baseline methods. The proposed method

5.2. Collision Rate Evaluation

Fig. 6 shows how the collision rate changes during training for the four algorithms. Each curve represents the average of five runs, and the shaded areas show the variability. The hybrid TD3 + RRT keeps the lowest collision rate through the entire training process, settling near 8%. The standard TD3 follows with about 13%, while DDPG reaches around 27%, and SAC records the highest rate, exceeding 40%. These results highlight clear differences in how each method handles obstacle avoidance under the same conditions.

5.3. Convergence Evaluation

Fig. 7 shows the evolution of the value function during training for the four algorithms. Each curve represents the average of five runs, with shaded areas indicating the standard deviation. The TD3 + RRT and TD3 models display a clear and steady upward trend, showing that both agents improve their value estimation over time. DDPG rises more slowly, suggesting delayed convergence and less stable learning. SAC, on the other hand, stays almost flat, which indicates weak progress in value estimation and poor adaptation to the training environment. These patterns reflect the different convergence behaviors of each method under identical conditions, with TD3 + RRT reaching stability faster and maintaining higher value estimates.

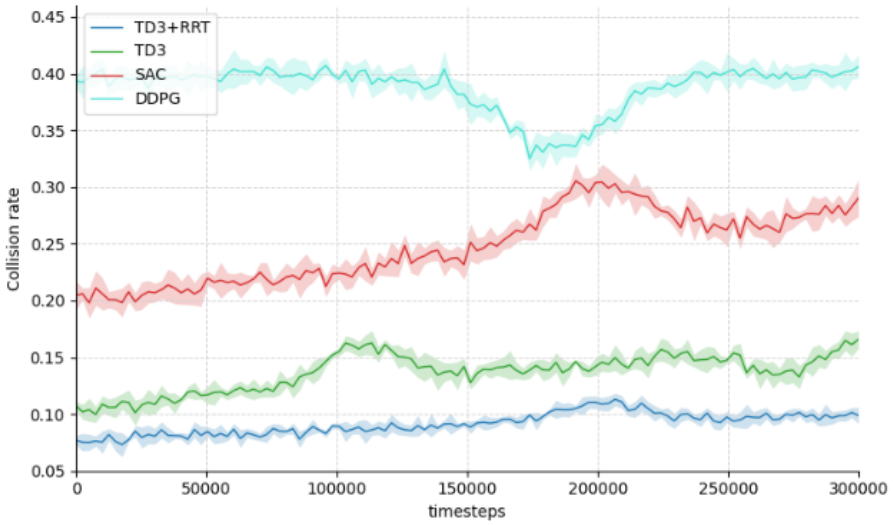

Fig. 6. Collision rate ± std over 300k timesteps for all evaluated algorithms

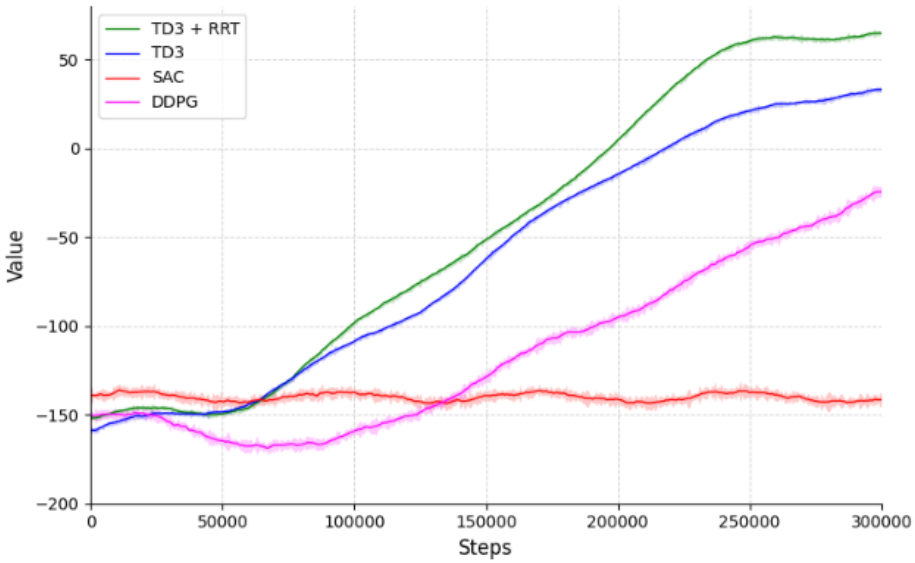

Fig. 7. Estimated value function over training steps for all methods

The results in Table 1 confirm these observations. The hybrid TD3 + RRT achieves the lowest mean square error (0.002) and the most stable performance, as shown by its low reward variance (±45). The standard TD3 follows closely, but with slightly higher variance and error. DDPG and SAC present larger errors and higher variability, which point to unstable convergence and weaker learning consistency.

Table 1. Convergence metrics based on reward variance and MSE over final training steps

Algorithm	Reward Variance (last 50k steps)	MSE
TD3 + RRT	**1800 ± 45**	**0.002**
TD3	1600 ± 70	0.008
DDPG	980 ± 65	0.021
SAC	570 ± 80	0.033

5.4. Transfer Learning Impact

Fig. 8 compares the training reward curves of agents initialized with and without transfer learning. The transferred agent starts with higher performance and quickly improves, reaching a peak reward of approximately 600 within the first 80,000 steps. In contrast, the agent trained from scratch progresses slowly, remaining below -100 for most of the training period. This result quantifies the acceleration and performance gain obtained through transfer initialization.

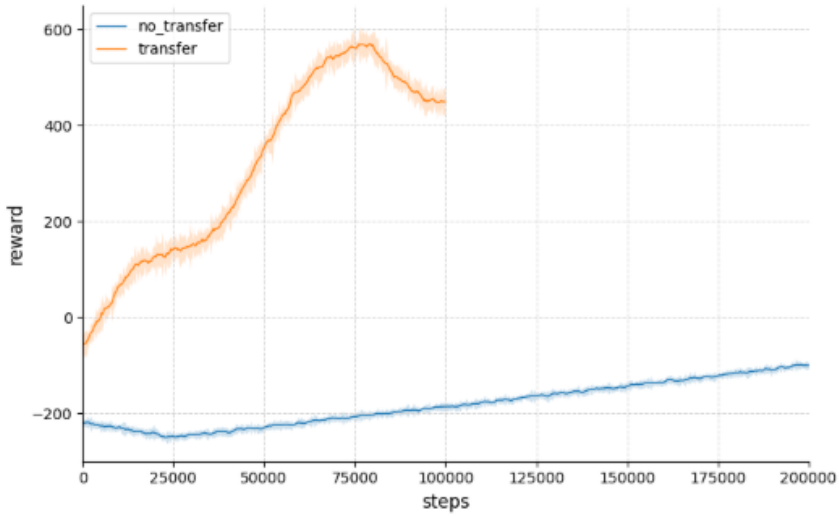

Fig. 8. Reward progression with and without transfer learning across 200k steps

5.5. Qualitative Trajectory Analysis

Fig. 9 shows sample trajectories obtained from three navigation methods: TD3 + RRT in red, TD3 in yellow, and SAC in green. The robot starts from the top-right corner of the map and moves toward the goal, which is represented by a fixed red circle. The test takes place in a dynamic environment that includes rotating cylinders, moving obstacles, and a shifting blue box. Each colored path represents one trial carried out under the same conditions. Both the red and yellow trajectories, corresponding to TD3 + RRT and TD3, successfully reach the red goal, while the green trajectory from SAC becomes trapped in the center area. The figure provides a clear visual comparison of how each algorithm navigates and adapts to a changing environment.

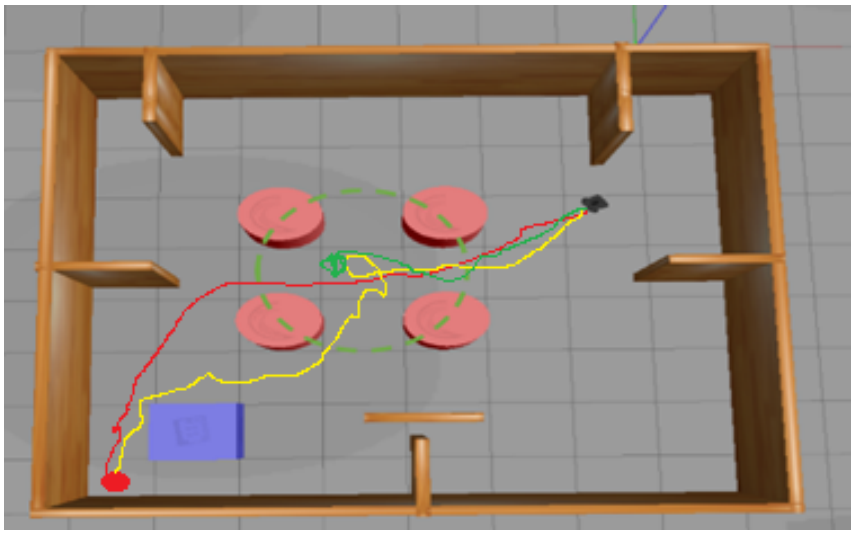

Fig. 9. Trajectories of TD3+RRT (red), TD3 (yellow), and SAC (green) toward the goal in a dynamic environment

5.6. Overall Comparison

Table 2 presents a summary of key performance metrics for all tested algorithms. Table 3 compares the impact of different global planners when combined with the same TD3 local policy.

Table 2. Overall performance comparison

Metric	TD3+RRT	TD3	DDPG	SAC	*p*-value (vs TD3)
Success rate (%)	**92 ± 3**	81 ± 5	68 ± 4	49 ± 6	0.008
Avg. path length (m)	**5.3**	5.9	6.8	7.4	0.012
Computation time (s/episode)	1.8 ± 0.2	1.3 ± 0.1	1.1 ± 0.1	**1.0 ± 0.1**	0.016

Table 3. Impact of global planner choice on navigation performance using TD3

Global Planner	Path Length (m)	Planning Time (s)	Success Rate (%)	Smoothness ↓
A*	4.8	**0.3**	76	0.42
RRT	5.1	0.5	**92**	**0.18**
PRM	**4.6**	0.8	68	0.35

6. Discussion

6.1. Main Findings of the Present Study

The experiments confirm the effectiveness of the proposed hybrid navigation framework that combines TD3 with RRT. Among all tested methods, TD3 with RRT gives the best overall performance. It reaches a cumulative reward of about 1850 ± 50, maintains a low collision rate of 8 percent, and shows stable convergence with little reward variation and a mean squared error of 0.002. The success rate also increases to 92 percent compared with 81 percent for TD3 alone. The average path length remains efficient at 5.3 meters, even with the addition of a global planner. The RRT component helps the robot plan longer routes, while TD3 manages short-term adjustments in real time. Together they produce safe, stable, and efficient navigation in dynamic environments.

6.2. Comparison with Other Studies

Earlier hybrid methods, such as A* combined with SAC, achieved good results in static settings but struggled to adapt to dynamic ones. In contrast, TD3 with RRT shows stronger reactivity and faster convergence when obstacles move or the environment changes. Previous studies, such as that of Ibarz et al. [23], noted that deep reinforcement learning can be unstable when models are transferred from simulation to real-world conditions. The use of transfer learning in our system helps overcome this problem and allows faster reward growth, reaching about 600 within the first 80,000 steps. When compared with global planners such as PRM or A*, the RRT planner achieves a better balance between planning time, path smoothness, and success rate. Gao et al.[16] Introduced a hybrid planner that combined the Probabilistic Roadmap Method with the TD3 algorithm to improve indoor navigation in static environments. Their PRM, combined with TD3 achieved good results in terms of path feasibility and training efficiency but remained limited to simple and largely static settings. In contrast, the hybrid framework proposed in this study extends the use of TD3 to dynamic and uncertain environments. The integration of RRT as the global planner allows faster adaptation to environmental changes and provides smoother paths while maintaining a higher success rate. This improvement confirms that incremental sampling-based planning aligns better with the exploratory nature of TD3, especially when operating under real-time constraints.

6.3. Implications and Explanation of Findings

The success of the hybrid approach comes from the complementary strengths of both components. RRT creates a global path that prevents the robot from becoming trapped in local dead ends. TD3 refines the movement at a local level by reacting to real-time sensor data. This combination improves sample efficiency and adaptability, especially in environments that change over time. The

low collision rate and stable learning behavior suggest that the agent generalizes well to new situations. The waypoints generated by RRT give stability and direction to TD3's learning, which reduces erratic motion. Faster convergence and lower error values show a smoother and safer learning process. The good results obtained with transfer learning also demonstrate that pre-trained models can adapt to new environments with little retraining, reducing the time and cost of development.

6.4. Strengths and Limitations

A major strength of this study is the use of a wide range of evaluation criteria. The analysis includes learning performance, convergence, safety, efficiency, and motion smoothness. Each metric is supported by several experimental trials and statistical measures, which reinforce the validity of the results. The qualitative study of trajectories further confirms the benefits of the hybrid method, showing consistent goal achievement and smooth motion in dynamic environments. Some limitations remain. The experiments were carried out with a single robot, the TurtleBot3, and in two handcrafted environments. This may reduce the generalization of the findings to other robotic platforms or more complex settings with people and unpredictable obstacles. The dynamic elements used in simulation, such as rotating or shifting obstacles, provide variety but still lack the randomness and speed changes found in real conditions.

7. Conclusion

This study introduced a hybrid navigation framework that combines the global planning ability of Rapidly Exploring Random Trees with the adaptive learning behavior of Twin Delayed Deep Deterministic Policy Gradient. The method was tested in both static and dynamic environments using several quantitative and qualitative metrics, including reward variance, convergence, collision rate, success rate, and path efficiency. The results show that TD3 with RRT delivers the best overall performance, reaching a success rate of 92 percent, a collision rate of only 8 percent, and a mean squared error of 0.002, with a clear improvement in learning stability compared with TD3, DDPG, and SAC.

Beyond the numerical results, this work provides theoretical insight into the complementarity between model-based planning and model-free reinforcement learning in navigation. The hybrid design uses RRT to ensure global feasibility and TD3 to make adaptive local decisions under uncertainty. This interaction improves training stability, generalization, and path safety, which remain major challenges in deep reinforcement learning for robotics.

Some limitations must be recognized. The experiments were conducted entirely in simulation, which does not fully reflect the imperfections of real environments such as sensor noise, actuator delay, or communication latency. These factors can reduce the reliability of the policy when applied to physical robots. The computational demand of running both a global planner and a learning-based controller is also high, particularly in large or dynamic scenes. The method depends on the quality of the map used by RRT, which may lose efficiency in highly cluttered or high-dimensional spaces. In addition, the benefits of transfer learning can decrease when the difference between the source and target environments becomes large, emphasizing the need for improved domain adaptation techniques.

Future research will aim to overcome these challenges. Planned directions include the integration of online SLAM to update maps in real time, optimization of computational cost using incremental RRT variants or lighter neural models, and hardware-in-the-loop testing with real robots. Other extensions will explore multi-robot coordination, the inclusion of safety constraints during policy optimization, and strategies such as domain randomization and noise modeling to close the gap between simulation and reality.

In summary, this work demonstrates that hybrid reinforcement learning architectures can achieve both adaptability and stability in robot navigation. By combining a sampling-based planner with a deep policy, the study contributes to a better understanding of how to design scalable and reliable

navigation systems. The results and remaining challenges outlined here provide a solid foundation for future progress toward the deployment of learning-driven autonomous robots in real-world environments.

Conflict of interest: On behalf of all authors, the corresponding author states that there is no conflict of interest.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: The authors did not receive support from any organization for the submitted work.

Financial interests: The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] H. B. Roummane and C. Daoui, "Localization and navigation of ROS-based autonomous robot in hospital environment," in *Business Intelligence*, R. El Ayachi, M. Fakir, and M. Baslam, Eds. Cham: Springer Nature Switzerland, 2023, pp. 159–172, https://doi.org/10.1007/978-3-031-37872-0_12.

[2] N. Liu, Z. Hu, M. Wei, P. Guo, S. Zhang, and A. Zhang, "Improved A* algorithm incorporating RRT* thought: A path planning algorithm for AGV in digitalised workshops," *Computers & Operations Research*, vol. 177, p. 106993, May 2025, https://doi.org/10.1016/j.cor.2025.106993.

[3] J. Choi, G. Lee, and C. Lee, "Reinforcement learning-based dynamic obstacle avoidance and integration of path planning," *Intelligent Service Robotics*, vol. 14, no. 5, pp. 663–677, Nov. 2021, https://doi.org/10.1007/s11370-021-00387-2.

[4] B. Tao and J.-H. Kim, "Deep reinforcement learning-based local path planning in dynamic environments for mobile robot," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 10, p. 102254, Dec. 2024, https://doi.org/10.1016/j.jksuci.2024.102254.

[5] A. N. A. Rafai, N. Adzhar, and N. I. Jaini, "A review on path planning and obstacle avoidance algorithms for autonomous mobile robots," *Journal of Robotics*, vol. 2022, no. 1, p. 2538220, Jan. 2022, https://doi.org/10.1155/2022/2538220.

[6] X. Lu, H. Woo, A. Faragasso, A. Yamashita, and H. Asama, "Robot navigation in crowds via deep reinforcement learning with modeling of obstacle uni-action," *Advanced Robotics*, vol. 37, no. 4, pp. 257–269, 2023, https://doi.org/10.1080/01691864.2022.2142068.

[7] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, Oct. 2021, https://doi.org/10.26599/TST.2021.9010012.

[8] H. Le, S. Saeedvand, and C.-C. Hsu, "A comprehensive review of mobile robot navigation using deep reinforcement learning algorithms in crowded environments," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 4, p. 158, Nov. 2024, https://doi.org/10.1007/s10846-024-02198-w.

[9] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," *Journal of Intelligent Manufacturing*, vol. 33, no. 2, pp. 387–424, Feb. 2022, https://doi.org/10.1007/s10845-021-01867-z.

[10] Y. Zhu, W. Z. Wan Hasan, H. R. Harun Ramli, N. M. H. Norsahperi, M. S. Mohd Kassim, and Y. Yao, "Deep reinforcement learning of mobile robot navigation in dynamic environment: A review," *Sensors*, vol. 25, no. 11, p. 3394, Jun. 2025, https://doi.org/10.3390/s25113394.

[11] A. N. Atiyah, N. Adzhar, and N. I. Jaini, "An overview: On path planning optimization criteria and mobile robot navigation," in *Journal of Physics: Conference Series*, vol. 1988, no. 1, p. 012036, 2021, https://doi.org/10.1088/1742-6596/1988/1/012036.

[12] E. S. Low, P. Ong, and C. Y. Low, "A modified Q-learning path planning approach using distortion concept and optimization in dynamic environment for autonomous mobile robot," *Computers & Industrial Engineering*, vol. 181, p. 109338, Jul. 2023, https://doi.org/10.1016/j.cie.2023.109338.

[13] J. Kim and G.-H. Yang, "Improvement of dynamic window approach using reinforcement learning in dynamic environments," *International Journal of Control, Automation and Systems*, vol. 20, no. 9, pp. 2983–2992, Sep. 2022, https://doi.org/10.1007/s12555-021-0462-9.

[14] R. Singh, J. Ren, and X. Lin, "A review of deep reinforcement learning algorithms for mobile robot path planning," *Vehicles*, vol. 5, no. 4, pp. 1423–1451, 2023, https://doi.org/10.3390/vehicles5040078.

[15] H. Liu, Y. Shen, C. Zhou, Y. Zou, Z. Gao, and Q. Wang, "TD3 based collision free motion planning for robot navigation," *arXiv preprint,* arXiv:2405.15460, 2024, https://doi.org/10.1109/CISCE62493.2024.10653233.

[16] J. Gao, W. Ye, J. Guo, and Z. Li, "Deep reinforcement learning for indoor mobile robot path planning," *Sensors*, vol. 20, no. 19, p. 5493, Oct. 2020, https://doi.org/10.3390/s20195493.

[17] V. D. Sharma, J. Lee, M. Andrews, and I. Hadžić, "Hybrid classical/RL local planner for ground robot navigation," *arXiv preprint,* arXiv:2410.03066, 2024, https://doi.org/10.48550/arXiv.2410.03066.

[18] S. Dey, A. Sadek, G. Monaci, B. Chidlovskii, and C. Wolf, "Learning whom to trust in navigation: dynamically switching between classical and neural planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 5235–5242, https://doi.org/10.1109/IROS55552.2023.10342308.

[19] Y. Liu, C. Wang, C. Zhao, H. Wu, and Y. Wei, "A soft actor-critic deep reinforcement-learning-based robot navigation method using LiDAR," *Remote Sensing*, vol. 16, no. 12, p. 2072, Jun. 2024, https://doi.org/10.3390/rs16122072.

[20] P. Li, D. Chen, Y. Wang, L. Zhang, and S. Zhao, "Path planning of mobile robot based on improved TD3 algorithm in dynamic environment," *Heliyon*, vol. 10, no. 11, Jun. 2024, https://doi.org/10.1016/j.heliyon.2024.e32167.

[21] B. Huang, J. Xie, and J. Yan, "Inspection robot navigation based on improved TD3 algorithm," *Sensors*, vol. 24, no. 8, p. 2525, Aug. 2024, https://doi.org/10.3390/s24082525.

[22] M. Park, S. Y. Lee, J. S. Hong, and N. K. Kwon, "Deep deterministic policy gradient-based autonomous driving for mobile robots in sparse reward environments," *Sensors*, vol. 22, no. 24, p. 9574, Dec. 2022, https://doi.org/10.3390/s22249574.

[23] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4–5, pp. 698–721, Apr.–May 2021, https://doi.org/10.1177/0278364920987859.

[24] H. Tu, Y. Deng, Q. Li, M. Song, and X. Zheng, "Improved RRT global path planning algorithm based on Bridge Test," *Robotics and Autonomous Systems*, vol. 171, p. 104570, Jan. 2024, https://doi.org/10.1016/j.robot.2023.104570.

[25] H. Jiang, M. A. Esfahani, K. Wu, K.-w. Wan, K.-k. Heng, H. Wang, and X. Jiang, "iTD3-CLN: Learn to navigate in dynamic scene through deep reinforcement learning," *Neurocomputing*, vol. 503, pp. 118–128, 2022, https://doi.org/10.1016/j.neucom.2022.06.102.

[26] Y. Yin, Z. Chen, G. Liu, J. Yin, and J. Guo, "Autonomous navigation of mobile robots in unknown environments using off-policy reinforcement learning with curriculum learning," *Expert Systems with Applications*, vol. 247, p. 123202, Aug. 2024, https://doi.org/10.1016/j.eswa.2024.123202.

[27] Y. Cao, N. Mohamad Nor, and Z. Samad, "An assistant algorithm model for a mobile robot to pass through a concave obstacle area," *SN Applied Sciences*, vol. 5, no. 8, p. 219, Jul. 2023, https://doi.org/10.1007/s42452-023-05433-5.

[28] J. Jermyn, "A comparison of the effectiveness of the RRT, PRM, and novel hybrid RRT-PRM path planners," *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. 12, pp. 600–611, Dec. 2021, https://doi.org/10.22214/ijraset.2021.39297.

[29] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari, "Path planning algorithms in the autonomous driving system: a comprehensive review," *Robotics and Autonomous Systems*, vol. 174, p. 104630, Apr. 2024, https://doi.org/10.1016/j.robot.2024.104630.

[30] J. Wittmann, F. Ochsenfarth, V. Sonneville, and D. Rixen, "Centralized vs. decoupled dual-arm planning taking into account path quality," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 4, p. 141, Sep. 2024, https://doi.org/10.1007/s10846-024-02175-3.

[31] C. Yuan, G. Liu, W. Zhang, and X. Pan, "An efficient RRT cache method in dynamic environments for path planning," *Robotics and Autonomous Systems*, vol. 131, p. 103595, Sep. 2020, https://doi.org/10.1016/j.robot.2020.103595.

[32] J. Jayan, "Enhancing robotic control: a TD3-based approach for planar continuum robots," in *2024 1st International Conference on Trends in Engineering Systems and Technologies (ICTEST)*, 2024, pp. 1–6, https://doi.org/10.1109/ICTEST60614.2024.10576194.

[33] Z. B. Hazem, F. Saidi, N. Guler, and A. H. Altaif, "Reinforcement learning-based intelligent trajectory tracking for a 5-DOF Mitsubishi robotic arm: comparative evaluation of DDPG, LC-DDPG, and TD3-ADX," *International Journal of Intelligent Robotics and Applications*, Jul. 2025, https://doi.org/10.1007/s41315-025-00475-x.

[34] H. A. Neamah and O. A. Mayorga Mayorga, "Optimized TD3 algorithm for robust autonomous navigation in crowded and dynamic human-interaction environments," *Results in Engineering*, vol. 24, p. 102874, Dec. 2024, https://doi.org/10.1016/j.rineng.2024.102874.

[35] R. Huang, H. He, Q. Su, M. Härtl, and M. Jaensch, "Enabling cross-type full-knowledge transferable energy management for hybrid electric vehicles via deep transfer reinforcement learning," *Energy*, vol. 305, p. 132394, Oct. 2024, https://doi.org/10.1016/j.energy.2024.132394.

[36] P. Sadhukhan and R. R. Selmic, "Proximal policy optimization for formation navigation and obstacle avoidance," *International Journal of Intelligent Robotics and Applications*, vol. 6, no. 4, pp. 746–759, 2022, https://doi.org/10.1007/s41315-022-00245-z.

[37] H. Wang, Y. Ye, J. Zhang, and B. Xu, "A comparative study of 13 deep reinforcement learning based energy management methods for a hybrid electric vehicle," *Energy*, vol. 266, p. 126497, Mar. 2023, https://doi.org/10.1016/j.energy.2022.126497.

[38] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros, and D. F. T. Gamarra, "Soft actor-critic for navigation of mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, p. 31, May 2021, https://doi.org/10.1007/s10846-021-01367-5.

[39] N. Botteghi, K. Alaa, M. Poel, B. Sirmacek, C. Brune, A. Mersha, and S. Stramigioli, "Low dimensional state representation learning with robotics priors in continuous action spaces," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 190–197, https://doi.org/10.1109/IROS51168.2021.9635936.

[40] J. C. Choton, J. Woods, and W. Hsu, "Efficient environment design for multi-robot navigation via continuous control," *arXiv e-prints*, arXiv:2508.14105, 2025, https://doi.org/10.48550/arXiv.2508.14105.

[41] S. Krishnan, B. Boroujerdian, W. Fu, A. Faust, and V. J. Reddi, "Air learning: a deep reinforcement learning gym for autonomous aerial robot visual navigation," *Machine Learning*, vol. 110, no. 9, pp. 2501–2540, Sep. 2021, https://doi.org/10.1007/s10994-021-06006-6.

[42] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, "Transfer learning: a friendly introduction," *Journal of Big Data*, vol. 9, no. 1, p. 102, 2022, https://doi.org/10.1186/s40537-022-00652-w.

[43] Y. Liu, H. Xu, D. Liu, and L. Wang, "A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102365, Dec. 2022, https://doi.org/10.1016/j.rcim.2022.102365.

[44] B. Abbyasov, T. Gamberov, V. Zhukova, T. Tsoy, E. A. Martínez-García, and E. Magid, "A tutorial on modelling a real office environment in Gazebo simulator," *Journal of Robotics, Networking and Artificial Life*, vol. 10, no. 2, pp. 166–169, 2023, https://doi.org/10.57417/jrnal.10.2_166.

[45] A. Farley, J. Wang, and J. A. Marshall, "How to pick a mobile robot simulator: a quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion," *Simulation Modelling Practice and Theory*, vol. 120, p. 102629, Nov. 2022, https://doi.org/10.1016/j.simpat.2022.102629.

[46] J. Bessler, G. B. Prange-Lasonder, L. Schaake, J. F. Saenz, C. Bidard, I. Fassi, M. Valori, A. B. Lassen, and J. H. Buurke, "Safety assessment of rehabilitation robots: a review identifying safety skills and current knowledge gaps," *Frontiers in Robotics and AI*, vol. 8, p. 602878, 2021, https://doi.org/10.3389/frobt.2021.602878.

[47] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, Sep. 2021, https://doi.org/10.1007/s10994-021-05961-4.

[48] S. Azadi, I. C. Green, A. Arnold, M. Truong, J. Potts, and M. A. Martino, "Robotic surgery: the impact of simulation and other innovative platforms on performance and training," *Journal of Minimally Invasive Gynecology*, vol. 28, no. 3, pp. 490–495, 2021, https://doi.org/10.1016/j.jmig.2020.12.001.

[49] J. Platt and K. Ricks, "Comparative analysis of ROS-Unity3D and ROS-Gazebo for mobile ground robot simulation," *Journal of Intelligent & Robotic Systems*, vol. 106, no. 4, p. 80, Dec. 2022, https://doi.org/10.1007/s10846-022-01766-2.

[50] F. H. Martinez, "TurtleBot3 robot operation for navigation applications using ROS," *Tekhnê*, vol. 18, no. 2, pp. 19–24, 2021, https://share.google/GlcH3xabrK1J4MfTp.