Autonomous Robot Navigation Using TD3 in Unknown Environments

***Abstract*—This work introduces a navigation framework designed for autonomous robotic systems to explore unknown environments while being directed towards a specified goal. The system uses deep reinforcement learning (DRL) to determine optimal navigation strategies. Points of interest (POI) within the environment are identified, evaluated, and chosen as waypoints to guide the robot towards the goal. The approach combines reactive local motion planning through DRL with a global navigation strategy, mitigating local optimization issues in navigation. The framework does not rely on pre-existing maps or prior knowledge. Results demonstrate the method's capability to outperform similar exploration techniques in both static and dynamic scenarios.**

I. Introduction and Motivation

Robotic exploration has seen significant advancements in simultaneous localization and mapping (SLAM), traditionally operated with human intervention. In SLAM, operators manually decide exploration priorities, which may include connecting locations or mapping routes. However, manual operation is not always feasible due to cost, resource limitations, or hazardous environments. This has fueled interest in fully autonomous systems capable of goal-oriented exploration. Such systems face dual challenges: determining the most promising POI to achieve a global goal and navigating uncertain environments effectively.

Modern advancements in DRL have enabled precise decision-making capabilities for autonomous agents. Despite their effectiveness, DRL systems can be hindered by their reactive nature, often leading to suboptimal navigation when tackling large-scale tasks. Addressing these limitations, this study proposes a fully autonomous exploration framework that integrates POI evaluation and selection with DRL-based motion planning to enable goal-driven navigation in unknown terrains.

Key contributions include:

- A global strategy for waypoint selection and navigation.
- Development of a neural network architecture based on the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm.
- Integration of DRL motion policies with global strategies to address local optimum problems.

II. Problem Statement

Robotic exploration has leveraged a wide range of methodologies, including SLAM systems, low-cost sensors, and neural network-based navigation. Traditional SLAM relies on human operators or predefined plans, extracting POI from map edges to explore free space. However, these methods often assume prior map availability. With neural networks, robots have learned to navigate using sensor inputs, albeit often restricted to modular tasks or local planning.

Advanced DRL techniques have shown promise in navigation tasks, enabling robots to avoid obstacles and make decisions in dynamic environments. Despite this, such methods face challenges like limited generalizability and local optimum traps. To address these gaps, this study combines DRL-based local motion control with a comprehensive global navigation strategy to enable goal-oriented exploration and mapping without requiring pre-existing maps.

A. Global Navigation

Global navigation involves selecting intermediate waypoints from identified POI. Without prior environmental knowledge, optimal paths cannot be precomputed. Therefore, the robot explores its surroundings, storing and evaluating POI based on criteria such as distance and accessibility. Two techniques are used for POI identification:

1) Detecting gaps between sequential laser readings exceeding a predefined threshold.
2) Identifying free space from non-numerical sensor readings.

An Information-based Distance Limited Exploration (IDLE) evaluation method scores POI, integrating distance, proximity to the goal, and map information. The POI with the lowest score is chosen as the next waypoint. The scoring formula is given as:

$$h(c_i) = \tanh\left(\frac{e^{\left(\frac{d(p_t,c_i)}{l_2-l_1}\right)^2}}{e^{\left(\frac{l_2}{l_2-l_1}\right)^2}}\right) l_2 + d(c_i,g) + e^{I_{i,t}}, \tag{1}$$

where:

- $d(p_t, c_i)$: Euclidean distance between the robot's position p_t and candidate POI c_i.
- l_1, l_2: Distance limits.
- $d(c_i, g)$: Distance between the candidate POI and the global goal g.
- $I_{i,t}$: Map information score, calculated using:

$$I_{i,t} = \frac{\sum_{w=-\frac{k}{2}}^{\frac{k}{2}} \sum_{h=-\frac{k}{2}}^{\frac{k}{2}} c_{(x+w)(y+h)}}{k^2}, \tag{2}$$

where k is the kernel size.

III. Proposed Solution

A. Local Navigation

The local navigation layer replaces traditional planners with a neural network-based motion policy. A TD3 algorithm trains the robot to navigate in continuous action spaces using simulated environments. The robot processes laser sensor inputs and waypoint coordinates to calculate linear and angular velocities. The policy function is defined as:

$$a = \left[v_{\max} \left(\frac{a_1+1}{2} \right), \omega_{\max} a_2 \right], \tag{3}$$

where:

- a_1, a_2: Action parameters representing linear and angular velocities.
- $v_{\max}, \omega_{\max}$: Maximum linear and angular velocities.

The policy gradient ideology is formulated as:
The policy gradient optimizes the policy π by maximizing the expected reward:

$$\max_{\theta} J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \pi_\theta \right]$$

where $\gamma \in [0, 1]$ is the discount factor, and θ represents the policy parameters.

The gradient is computed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^\pi, a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s, a) \right]$$

Rewards are calculated as:

$$r(s_t, a_t) = \begin{cases} r_g, & \text{if } D_t < \eta_D, \\ r_c, & \text{if collision}, \\ v - |\omega|, & \text{otherwise}, \end{cases} \tag{4}$$

where:

- D_t: Distance to the goal.
- η_D: Threshold for reaching the goal.
- r_g, r_c: Goal reward and collision penalty.

B. Exploration and Mapping

As the robot traverses waypoints, it simultaneously maps its surroundings. Using laser sensors and odometry data, an occupancy grid is constructed. Waypoints are dynamically updated to ensure progress towards the goal while avoiding previously explored or unreachable areas. The overall exploration algorithm is summarized as:

Algorithm 1 Goal-Driven Autonomous Exploration

```
Set global goal $g$ and threshold $\delta$.
**while** reachedGlobalGoal $\neq$ True **do**
    Read sensor data and update the map.
    Extract POI.
    Evaluate POI using the IDLE method and select the optimal waypoint.
    Use the TD3 policy to calculate and execute actions.
**end while**=0
```

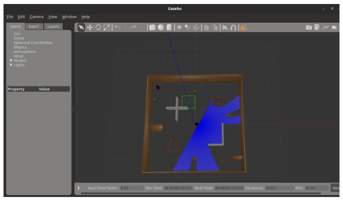


Fig. 1. gazebo-simulation

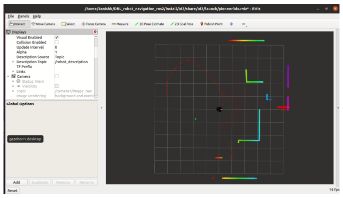
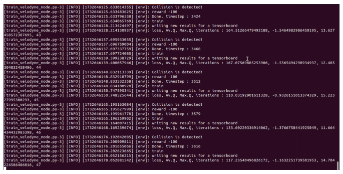


Fig. 2. rviz-visualization

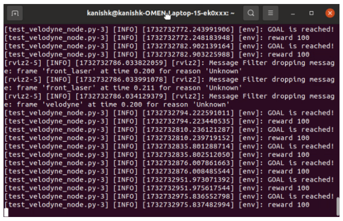
C. Experimental Setup

Experiments were conducted in simulation and real-world scenarios using a Pioneer P3-DX robot equipped with laser sensors. Local navigation training was performed in Gazebo with ROS integration, and real-world testing used a mini-PC onboard the robot. Metrics such as travel distance, time, and mapping accuracy were recorded to compare the proposed framework against existing methods.



```
[train_velodyne_node.py-3] [INFO] [1732646125.633014355] [env]: Collision is detected!
[train_velodyne_node.py-3] [INFO] [1732646125.633483625] [env]: reward -100
[train_velodyne_node.py-3] [INFO] [1732646125.633796538] [env]: Done. timestep : 3424
[train_velodyne_node.py-3] [INFO] [1732646125.634065769] [env]: train
[train_velodyne_node.py-3] [INFO] [1732646128.213424497] [env]: writing new results for a tensorboard
[train_velodyne_node.py-3] [INFO] [1732646128.214130937] [env]: loss, Av.Q, Max.Q, iterations : 164.51266479492188, -1.5464982986450195, 13.627
458572387695, 43
[train_velodyne_node.py-3] [INFO] [1732646137.695933035] [env]: Collision is detected!
[train_velodyne_node.py-3] [INFO] [1732646137.696759004] [env]: reward -100
[train_velodyne_node.py-3] [INFO] [1732646137.697337759] [env]: Done. timestep : 3468
[train_velodyne_node.py-3] [INFO] [1732646137.697724080] [env]: train
[train_velodyne_node.py-3] [INFO] [1732646139.399238729] [env]: writing new results for a tensorboard
[train_velodyne_node.py-3] [INFO] [1732646139.400057946] [env]: loss, Av.Q, Max.Q, iterations : 167.07569885253906, -1.5565494298934937, 12.485
304832458496, 44
[train_velodyne_node.py-3] [INFO] [1732646148.832113339] [env]: Collision is detected!
[train_velodyne_node.py-3] [INFO] [1732646148.832918799] [env]: reward -100
[train_velodyne_node.py-3] [INFO] [1732646148.833532358] [env]: Done. timestep : 3512
[train_velodyne_node.py-3] [INFO] [1732646148.834100928] [env]: train
[train_velodyne_node.py-3] [INFO] [1732646150.747595341] [env]: writing new results for a tensorboard
[train_velodyne_node.py-3] [INFO] [1732646150.748525644] [env]: loss, Av.Q, Max.Q, iterations : 110.03192901611328, -0.9326151013374329, 15.223
27995300293, 45
[train_velodyne_node.py-3] [INFO] [1732646165.195163084] [env]: Collision is detected!
[train_velodyne_node.py-3] [INFO] [1732646165.195627999] [env]: reward -100
[train_velodyne_node.py-3] [INFO] [1732646165.195961778] [env]: Done. timestep : 3579
[train_velodyne_node.py-3] [INFO] [1732646165.196259902] [env]: train
[train_velodyne_node.py-3] [INFO] [1732646168.164007415] [env]: writing new results for a tensorboard
[train_velodyne_node.py-3] [INFO] [1732646168.169239674] [env]: loss, Av.Q, Max.Q, iterations : 133.60220336914062, -1.3766758441925049, 11.664
434432983398, 46
[train_velodyne_node.py-3] [INFO] [1732646176.192042085] [env]: Collision is detected!
[train_velodyne_node.py-3] [INFO] [1732646176.200909811] [env]: reward -100
[train_velodyne_node.py-3] [INFO] [1732646176.201655066] [env]: Done. timestep : 3616
[train_velodyne_node.py-3] [INFO] [1732646176.202623665] [env]: train
[train_velodyne_node.py-3] [INFO] [1732646178.052156215] [env]: writing new results for a tensorboard
[train_velodyne_node.py-3] [INFO] [1732646178.052881542] [env]: loss, Av.Q, Max.Q, iterations : 117.23540496826172, -1.1632251739501953, 14.704
224586486816, 47
```

Fig. 3. results before training



```
[test_velodyne_node.py-3] [INFO] [1732732772.243991906] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732772.248183948] [env]: reward 100
[test_velodyne_node.py-3] [INFO] [1732732782.902139164] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732782.903225988] [env]: reward 100
[rviz2-5] [INFO] [1732732786.033822059] [rviz2]: Message Filter dropping messag
e: frame 'front_laser' at time 0.200 for reason 'Unknown'
[rviz2-5] [INFO] [1732732786.033991078] [rviz2]: Message Filter dropping messag
e: frame 'front_laser' at time 0.211 for reason 'Unknown'
[rviz2-5] [INFO] [1732732786.034129379] [rviz2]: Message Filter dropping messag
e: frame 'velodyne' at time 0.200 for reason 'Unknown'
[test_velodyne_node.py-3] [INFO] [1732732794.222591011] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732794.223440535] [env]: reward 100
[test_velodyne_node.py-3] [INFO] [1732732810.236121287] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732810.239719152] [env]: reward 100
[test_velodyne_node.py-3] [INFO] [1732732835.801288714] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732835.802512050] [env]: reward 100
[test_velodyne_node.py-3] [INFO] [1732732876.007861663] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732876.008485544] [env]: reward 100
[test_velodyne_node.py-3] [INFO] [1732732951.973071392] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732951.975617544] [env]: reward 100
[test_velodyne_node.py-3] [INFO] [1732732975.836552798] [env]: GOAL is reached!
[test_velodyne_node.py-3] [INFO] [1732732975.837482994] [env]: reward 100
```

Fig. 4. results after training

IV. Results

A. Quantitative Results

Table I summarizes the results comparing GD-RL and NF based on average distance traveled (Av. Dist.) and time taken (Av. T.) over 5 goal destinations. It is visible from the experiment that the conventional approach cheaper in terms of cost but it is to be noted that DRL proved to make the robot reach the goal 5/5 times whereas NF stood 4/5.

Method	Av. Dist. (m)	Av. T. (s)
DRL	77.02	171.82
NF	40.74	109.11

TABLE I
QUANTITATIVE COMPARISON OF GD-RL AND NF.

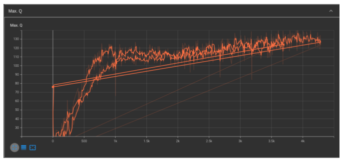


Fig. 5. maximum Q convergence

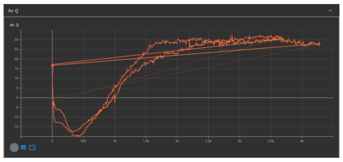


Fig. 6. average Q convergence

B. Qualitative Observations

Simulations validated the framework's ability to navigate cluttered and dynamic environments. In scenarios involving local optima or unknown obstacles, the robot demonstrated adaptive behavior, successfully escaping traps and navigating to the goal while building accurate maps. The experimental results indicate that while the conventional Nearest Frontier (NF) approach is more cost-effective in terms of average distance traveled and time taken, it lacks reliability in goal attainment. The Deep Reinforcement Learning (DRL)-based method ensures consistent performance, achieving 5/5 successful goal completions compared to 4/5 for NF. This highlights the trade-off between efficiency and reliability, with DRL demonstrating superior robustness in achieving exploration goals even in challenging environments.

V. Conclusion

This study presents a DRL-based autonomous exploration system capable of navigating unknown environments without prior knowledge or human supervision. By combining global navigation strategies with DRL-driven local motion policies, the framework addresses key challenges such as local optima and mapless navigation. Experimental results highlight the system's efficiency, reliability, and adaptability. Future work will focus on generalizing the framework for diverse robotic platforms and integrating advanced neural architectures to further enhance decision-making capabilities.

References

[1] Brilliant.org. "A* Search — Brilliant Math Science Wiki." Brilliant.org, 2016, brilliant.org/wiki/a-star-search/.

[2] Kumar, Rajesh. "The A* Algorithm: A Complete Guide." Datacamp.com, DataCamp, 7 Nov. 2024, www.datacamp.com/tutorial/a-star-algorithm.

[3] Reinis Cimurs, Il Hong Suh, Jin Han Lee, "Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning," *arXiv*, 2021.