Deep Reinforcement Learning for Mobile Robot Navigation in Dynamic Environments

1st Ahmed Yesuf Nurye
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology
Warsaw, Poland
https://orcid.org/0009-0004-6372-3190

2nd Elżbieta Jarzębowska
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology
Warsaw, Poland
elzbieta.jarzebowska@pw.edu.pl

***Abstract*—We present a framework for mobile robot navigation in dynamic environments using Deep Reinforcement Learning (DRL) and the Robot Operating System (ROS). Traditional navigation methods often lack the real-time adaptability required in highly dynamic settings. To address this, we leverage the TD7 algorithm—an extension of the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm incorporating state and state-action embeddings—to directly map raw sensor inputs to control actions. These embeddings, trained to minimize the mean squared error (MSE) between the encoded state-action representation and the transition-predicted next state, enhance the system's ability to model environment dynamics and improve navigation performance.**

Extensive simulations were conducted in custom Gazebo environments of increasing complexity, ranging from open spaces to scenarios with static obstacles and moving actors. Performance was evaluated based on navigation success rate, time to goal, path efficiency, and collision rate. Results indicate that this approach consistently improves navigation performance, particularly in highly dynamic environments.

***Index Terms*—Mobile Robot Navigation, Deep Reinforcement Learning, ROS, Gazebo**

I. Introduction

We are now in an era where robots have demonstrated remarkable efficiency in controlled environments, such as factories and warehouses, where they perform repetitive tasks with high precision [1], [2]. These successes, however, underscore a significant challenge when considering deployment in dynamic, real-world settings [3], [4]. Despite substantial advances in robotics, their widespread use in home assistance, healthcare, and collaborative industrial environments remains limited. The primary obstacle is the unpredictability and complexity of these real-world settings, where human behavior and rapidly changing conditions create a highly variable and uncertain landscape [5].

In such environments, robots must not only navigate around obstacles and adapt to evolving terrains but also interact seamlessly with humans, who often behave unpredictably. These tasks demand a level of adaptability and learning that exceeds the capabilities of traditional robotic systems. Therefore, developing robust navigation frameworks capable of handling uncertainty and variability in real-time is essential for the successful deployment of robots in dynamic settings.

Motivated by these challenges, this paper presents a navigation framework for dynamic environments based on the TD7 algorithm [6], which integrates state-action representation learning, policy checkpoints, and the Twin Delayed Deep Deterministic Policy Gradient (TD3) method. Unlike conventional social navigation approaches that employ multiple networks for separate tasks, our method is computationally efficient. A key insight of this work is that incorporating next-state prediction significantly improves navigation performance, allowing robots to better anticipate and respond to environmental changes.

II. Related works

A. Conventional Navigation Approach

In robot navigation, two critical questions must be addressed: *"Where am I and who is around me?"* related to mapping and localization; *"Where am I going and how should I get there?"* involving path planning and obstacle avoidance.

1) Simultaneous Localization and Mapping: SLAM is a process in which robots simultaneously construct a map of an unknown environment while deducing their own location within that map [7]. This problem is mostly considered solved for static environments [8]. However, real-world applications often involve dynamic elements that can degrade SLAM performance.

Leveraging recent advancements in deep learning, there has been progress in handling dynamic environments. DynaSLAM [9] handles dynamic environments by leveraging Mask R-CNN for semantic segmentation. It can identify and segment dynamic objects, removing them from the SLAM pipeline to prevent them from affecting the map. However, the reliance on computationally intensive deep learning models like Mask R-CNN poses challenges for real-time performance, particularly on devices with limited processing power. DynaSLAM II [10], unlike its predecessor, which focused on removing dynamic elements, integrates the tracking of these objects into the SLAM process, allowing for mutual improvements in tracking accuracy and map consistency.

In summary, almost all theories and implementations of current SLAM approaches are built on the static world assumption, treating any moving objects in the environment as outliers to the static model, which are intentionally ignored during tracking and mapping [11], [12]. This idealized setup, therefore, can only handle a small number of dynamic elements and is not suitable for many real-world applications, particularly in environments where humans are present and constant changes occur.

2) Path Planning: Planning is one of the fundamental and most studied problems in robotics [13]. The basic motion planning problem is a geometric problem of finding a collision-free path for a robot among rigid static obstacles [14]. Several extensions of the basic problem have been studied, where, for instance, kinematic constraints (such as joint limits and linkage configurations) and dynamic constraints (such as forces, torques, and inertia) limit the robot's motions, multiple robots have to be coordinated, and moving obstacles have to be considered. Therefore, the objective of path planning is to guide the robot from an initial starting point to a target goal while adhering to the robot's motion constraints [15]. Often, path planning is divided into two stages [16], [17]: global path planning, which involves generating an overall route considering the entire environment, and local path planning, which focuses on real-time navigation and obstacle avoidance based on immediate sensor data.

There have been various algorithms proposed for navigation in dynamic environments. These include methods such as the social force model [18], originally used to describe pedestrian dynamics, where motion is influenced by social forces comprising several components, including acceleration toward the desired velocity, distance from other pedestrians and boundaries, and a term modeling attractive forces. The reciprocal velocity obstacles method for real-time multi-agent navigation [19] introduces a local, reactive collision avoidance approach that implicitly assumes other agents use similar strategies to generate safe and oscillation-free motions. However, most navigation strategies encounter the freezing robot problem [3], which limits their applicability in highly dynamic environments.

B. Deep Reinforcement Learning in Mobile Robot Navigation

There are, however, deep reinforcement learning (DRL) based social navigation strategies that have been proposed over the years. Reference [20] introduces a decentralized multi-agent collision avoidance algorithm. The approach involves training a pair of agents to navigate around each other to learn a value network that encodes the expected time to goal and then generalizing this network in a principled way to handle multi-agent scenarios. Crowd-Robot Interaction [21] presents an approach to improve robot navigation in crowded environments by explicitly modeling interactions between humans and the robot as well as between humans themselves using a self-attention mechanism.

Unlike these methods, our approach is simpler and focuses on improving navigation performance in dynamic environments by predicting the next environmental state.

III. Methods

A. Formulation of the Problem of Mobile Robot Navigation as a Reinforcement Learning Problem

1) State Space: The state space must encompass all relevant information about the agent's environment and its current status. The environment is represented by laser scanner readings, which are grouped into 20 bins. Each bin captures the minimum distance to an obstacle within its field of view (FOV). This method of binning reduces the complexity of the raw sensor data while retaining essential information needed for navigation. Fig. 1 illustrates how these laser scans are aggregated into bins to construct the environment state.

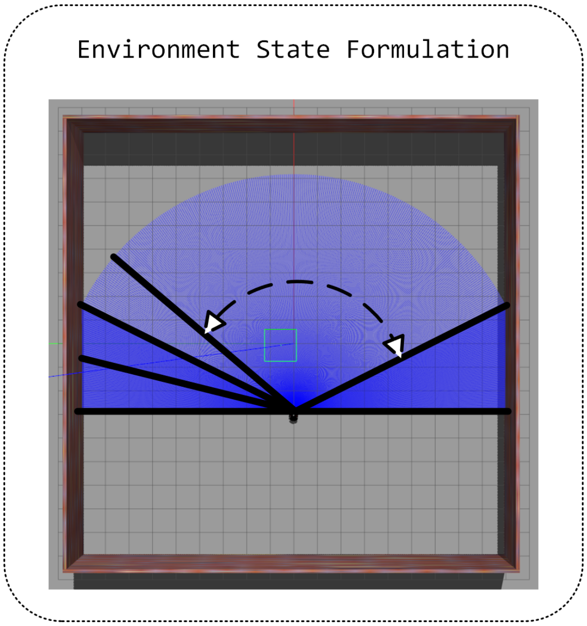

Figure 1: Environment state formulated by selecting the shortest range from each bin as a representative.

In addition to the environment state, the agent's current status is defined by four values: the distance from the goal d, the relative heading to the goal (θ), and the linear (v) and angular (ω) velocities that were executed in the previous time step as depicted in Fig. 2. These parameters are crucial as they provide the agent with information about its current trajectory and position relative to the target goal.

Combining these elements results in a state space ($\mathcal{S}$) with 24 dimensions. This state space is designed to be sufficiently rich to capture the necessary details of both the environment and the agent's status, enabling the reinforcement learning algorithm to make informed decisions during navigation.

2) Action Space: The agent, Pioneer P-3DX differential drive mobile robot, can perform actions by altering its forward velocity (v) and angular velocity (ω), resulting in a continuous action space with 2 dimensions. The range for these actions extends from $[-1m/s, -1rad/s]$ to $[1m/s, 1rad/s]$. For practical reasons, we have constrained the agent to only execute forward motions by clipping negative linear velocities. This

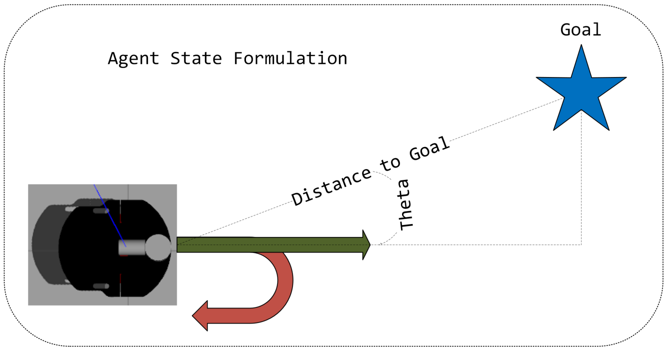

Figure 2: Formulation of the agent state based on target goal and actions taken.

restriction is due to the laser scan's 180° FOV, which makes backward movement impractical as it would leave the agent blind to obstacles behind it.

3) Reward Formulation: The reward function is designed to guide the agent in achieving its navigation goals by encouraging the selection of optimal paths while avoiding dangerous or sub-optimal actions. To this end, it is composed of three key components. The first component provides positive reinforcement for reaching the target goal. The second component imposes a penalty for collisions with obstacles. The third component offers an immediate reward to discourage the agent from coming too close to obstacles and from spending excessive time to reach the goal. These components collectively ensure that the agent navigates efficiently and safely towards its goal.

$$R = \begin{cases} r_t, & \text{if } d_t < \text{GOAL\_THRESHOLD}, \\ r_c, & \text{if } d_c < \text{COLLISION\_THRESHOLD}, \\ r_i, & \text{otherwise.} \end{cases} \tag{1}$$

Where r_t is the reward given when the agent reaches the target goal, r_c is the penalty incurred when the agent collides with an obstacle, r_i is the reward for other intermediate cases, d_t distance to target goal, and d_c is distance to the closest obstacle.

$$r_i = \alpha(v - |\omega|) - c \tag{2}$$

r_i is directly proportional to the difference between v and $|\omega|$, which helps in achieving smooth motion. Furthermore, a small constant negative reward $(-c)$ is provided to encourage the agent to reach the target in a shorter time period. The specific values of each parameter used in the reward formulation are given in Table I.

Table I: Reward parameters.

r_t	r_c	α	c
100	−100	0.5	0.001

B. Solution to the Problem of Mobile Robot Navigation in a Dynamic Environment

The proposed solution leverages a DRL approach that is grounded in the TD3 algorithm [22] and its recent advancement, TD7 [6]. TD3 is recognized for its effectiveness in handling continuous action spaces and enhances training stability through key techniques such as twin Q-networks and delayed policy updates. Building on this, TD7 introduces state-action learned embeddings, which facilitate the modeling of environmental dynamics in a latent space. This enhancement enables the network to better comprehend and adapt to changes in the environment, thereby improving navigation performance in complex, dynamic settings.

1) Network Architecture: To effectively capture dynamic actors in the environment, enabling the policy to make more informed actions, it is essential to predict the next state of the environment accurately. TD7 employs a pair of encoders (f, g). The encoder $f(s)$ transforms the state s into a state embedding z^s, while $g(z^s, a)$ jointly encodes both the state embedding z^s and action a into a state-action embedding z^{sa}. This encoding process is designed to capture the relevant structures within the observation space and to model the transition dynamics of the environment effectively. The encoders (f, g) are jointly trained to predict the next state embedding, which is decoupled from the training of the value function and policy.

Algorithm 1 Online TD7 Algorithm [6]

```
Initialize:                                              ▷ Before training
- Policy $\pi_{t+1}$, value function $Q_{t+1}$, encoders $(f_{t+1}, g_{t+1})$.
- Target policy $\pi_t$, target value function $Q_t$, fixed encoders
$(f_t, g_t)$, target fixed encoders $(f_{t-1}, g_{t-1})$.
- Checkpoint policy $\pi_c$, checkpoint encoder $f_c$.
for episode $= 1$ to final_episode do
                                                         ▷ Data collection
    Collect and store transitions using current policy, $\pi_{t+1}$.
    if checkpoint_condition then                         ▷ Checkpointing
        if actor $\pi_{t+1}$ outperforms checkpoint $\pi_c$ then
            Update checkpoint $\pi_c \leftarrow \pi_{t+1}$, $f_c \leftarrow f_t$.
        end if
                                                         ▷ Training
        for $i = 1$ to timesteps_since_training do
            Sample transitions from LAP replay buffer.
            Train encoder, value function, and policy.
            if target_update_frequency steps passed then
                Update target networks.
            end if
        end for
    end if
end for
```

The embeddings are designed to capture the underlying structure of the environment [6]. However, they may not include all relevant information needed by the value function and policy, such as features related to the reward, current policy, or task horizon. To address this, TD7 concatenate the

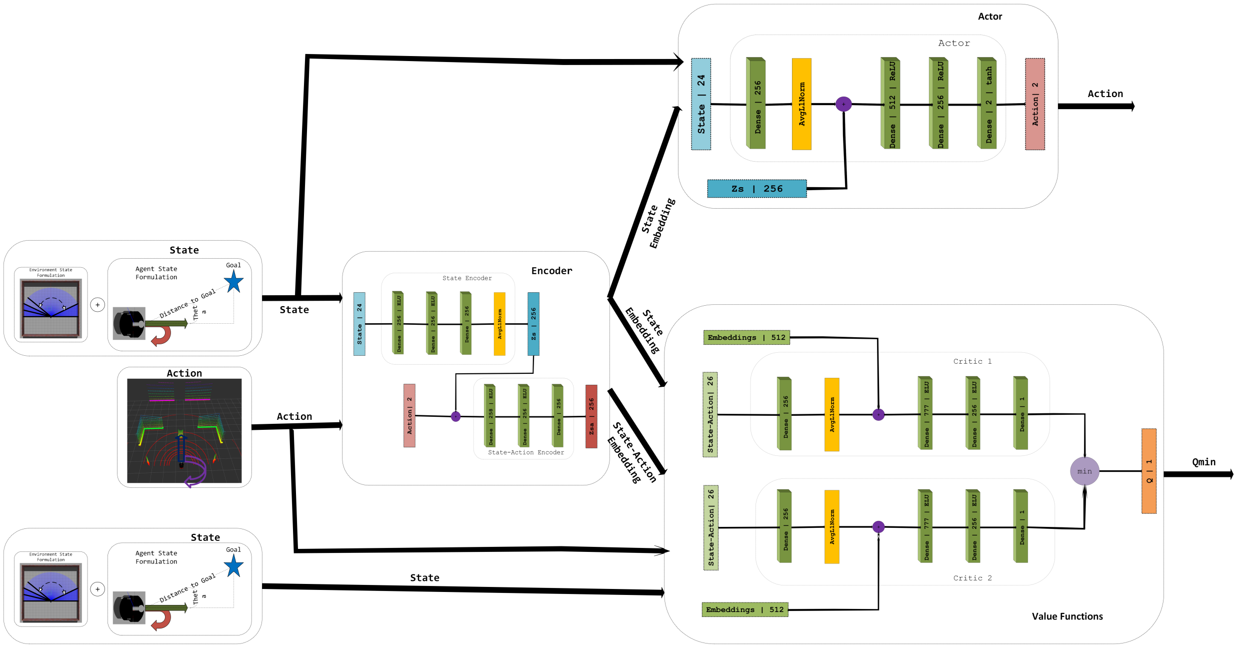

Figure 3: TD7 network architecture used in the implementation.

embeddings with the original state and action, as shown in Fig. 3. This approach allows the value and policy networks to learn the necessary internal representations for their respective tasks.

2) Agent Training: The training process was conducted using a laptop, which is equipped with a 12th Gen Intel® Core^{TM} i5-12500H processor featuring 16 cores. This setup also includes an NVIDIA GeForce RTX 4050 Laptop GPU and 16GB of RAM. The DRL agent was trained using the TD7 algorithm, which is detailed in Algorithm 1.

IV. Experimental Results

We have tested our system in a range of environments that vary in complexity to assess its robustness and effectiveness. For comparison purposes, we evaluated our system against a baseline: *Goal-Driven Autonomous Exploration Through Deep Reinforcement Learning* (GDAE) [23].

A. Test Environment 1: Environment with No Obstacle

To comprehensively assess the system's performance, we conducted *five* independent test runs. The results of our experiments are summarized in Table II, where we compare the performance of our method, against the baseline. In this relatively simple environment, both methods were able to guide the agent to the target goal with a 100% success rate. The baseline method, however, exhibited slightly better performance in terms of the average time taken to reach the goal and the average distance traveled.

Table II: Environment one, result comparison.

Method	Avg. Time(sec)	Avg. Distance m	Success	Collision
Ours	11.5874	10.2191	1.0	0.0
Baseline	10.9508	9.4772	1.0	0.0

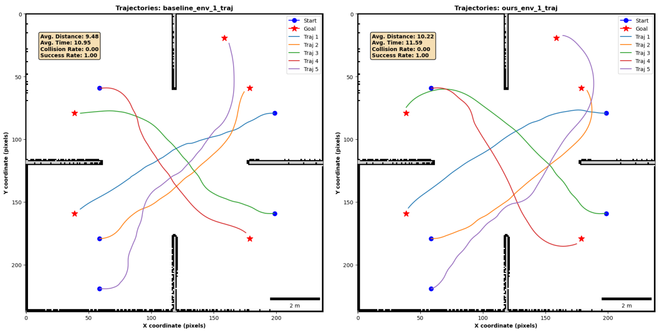

Figure 4: Agent trajectories for test environment one overlaid on the map of the environment.

B. Test Environment 2: Environment with Static Obstacles

Similar to test environment 1, we conducted *five* test runs in this environment. The data presented in Table III clearly demonstrate that our method outperforms the baseline, particularly in terms of success rate and collision rate. Our method consistently achieved a 100% success rate across all test runs, successfully navigating to the goal without any collisions. To ensure an accurate comparison, trajectories where the target goal was not reached were excluded from the calculation of average time and average distance traveled.

Table III: Environment two, result comparison.

Methods	Avg. Time (sec)	Avg. Distance(m)	Success	Collision
Ours	12.3035	9.9404	1.0	0.0
Baseline	10.5075	8.9495	0.6	0.4

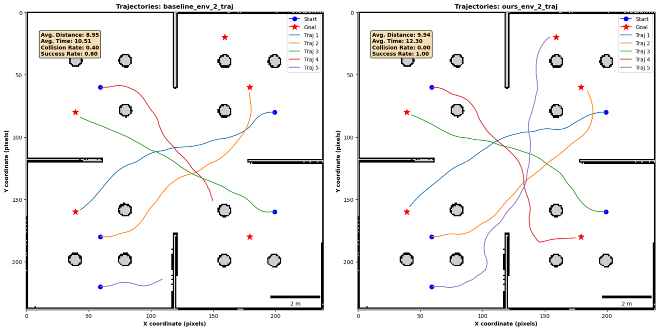

Figure 5: Agent trajectories for test environment one overlaid on the map of the environment.

C. *Test Environment 3: Environment Shared with Other Actors*

We conducted *five* test runs in this environment, and the quantitative comparison is provided in Table IV. As the data in this table indicates, our method outperformed the baseline method across all metrics. However, while our method performed well, it did not achieve a 100% success rate, indicating that there is still room for improvement.

Table IV: Environment three, result comparison.

Methods	Avg. Time(sec)	Avg. Distance m	Success	Collision
Ours	6.0847	5.0845	0.8	0.2
Baseline	6.6669	5.6258	0.6	0.4

D. *Summary of Performance Comparison with Baseline*

This section summarizes the comparison of results with the baseline across all environments. Fig. 6 presents a summary of the performance comparison. The vertical axis represents the three environments, while the horizontal axis displays the values of the respective test metrics: meters for average distance and percentage for success rate. In test environment 1, both our method and the baseline performed well, achieving an average success rate of 100%. However, in test environments 2 and 3, our method outperformed the baseline, particularly in terms of success rate.

V. Conclusions

In this paper, we have presented a new framework for mobile robot navigation in dynamic environments using DRL. The proposed approach leverages the TD7 algorithm, an enhancement of TD3 with state-action embeddings, to enable efficient and reliable navigation in environments characterized by both static and dynamic obstacles. The framework was implemented using ROS and validated through extensive simulations in the Gazebo environment.

The experimental results from the three test cases demonstrate that the proposed method outperforms the baseline approach, particularly in complex environments, in terms of success rate and collision avoidance. In simpler environments without obstacles, both our method and the baseline achieved a 100% success rate. However, as the environment's complexity increased—with the introduction of static obstacles and dynamic actors—our method consistently outperformed the baseline.

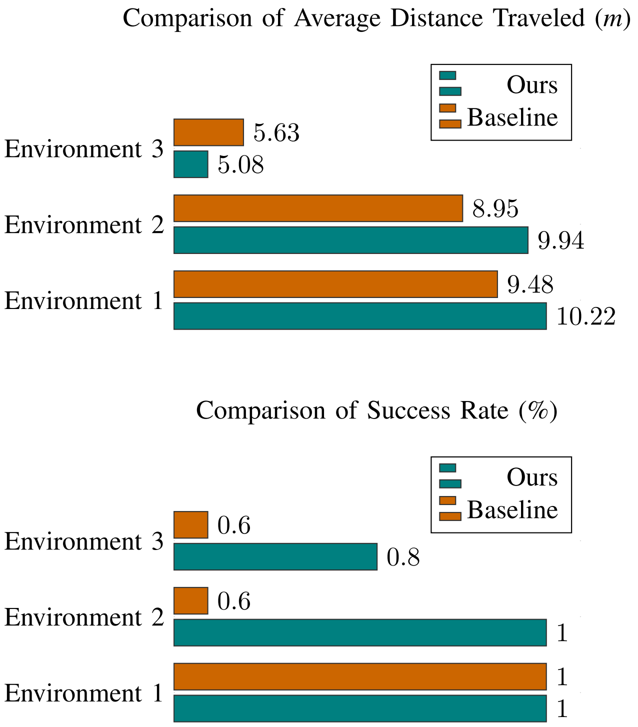

Figure 6: Summary of performance comparison with baseline.

While the proposed framework demonstrates strong performance, several challenges remain. Future work could explore embedding social norms into the navigation policy to improve human-robot interaction in shared spaces. Additionally, incorporating explicit human intent prediction and adapting to diverse social contexts would further enhance the system's real-world applicability.

References

[1] A. Grau, M. Indri, L. L. Bello, and T. Sauter, "Industrial robotics in factory automation: From the early stage to the internet of things," in *IECON 2017-43rd annual conference of the IEEE industrial electronics society*, IEEE, 2017, pp. 6159–6164.

[2] A. Grau, M. Indri, L. Lo Bello, and T. Sauter, "Robots in industry: The past, present, and future of a growing collaboration with humans," *IEEE Industrial Electronics Magazine*, vol. 15, no. 1, pp. 50–61, 2021. DOI: 10.1109/MIE.2020.3008136.

[3] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 797–803. DOI: 10.1109/IROS.2010.5654369.

[4] M. B. Alatise and G. P. Hancke, “A review on challenges of autonomous mobile robot and sensor fusion methods,” *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020. DOI: 10.1109/ACCESS.2020.2975643.

[5] C. Mavrogiannis, F. Baldini, A. Wang, *et al.*, “Core challenges of social robot navigation: A survey,” *J. Hum.-Robot Interact.*, vol. 12, no. 3, Apr. 2023. DOI: 10.1145/3583741. [Online]. Available: https://doi.org/10.1145/3583741.

[6] S. Fujimoto, W.-D. Chang, E. J. Smith, S. S. Gu, D. Precup, and D. Meger, “For sale: State-action representation learning for deep reinforcement learning,” in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS ’23, New Orleans, LA, USA: Curran Associates Inc., 2023.

[7] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part i,” *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[8] X. Gao, T. Zhang, Y. Liu, and Q. Yan, *14 Lectures on Visual SLAM: From Theory to Practice*. Publishing House of Electronics Industry, 2017.

[9] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, “Dynaslam: Tracking, mapping, and inpainting in dynamic scenes,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018. DOI: 10.1109/LRA.2018.2860039.

[10] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, “Dynaslam ii: Tightly-coupled multi-object tracking and slam,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021. DOI: 10.1109/LRA.2021.3068640.

[11] Y. Sun, M. Liu, and M. Q.-H. Meng, “Improving rgb-d slam in dynamic environments: A motion removal approach,” *Robotics and Autonomous Systems*, vol. 89, pp. 110–122, 2017, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2016.11.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015302232.

[12] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, “Mid-fusion: Octree-based object-level multi-instance dynamic slam,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5231–5237. DOI: 10.1109/ICRA.2019.8794371.

[13] M. Elbanhawi and M. Simic, “Sampling-based robot motion planning: A review,” *IEEE Access*, vol. 2, pp. 56–77, 2014. DOI: 10.1109/ACCESS.2014.2302442.

[14] J.-C. Latombe, “Motion planning: A journey of robots, molecules, digital actors, and other artifacts,” *The International Journal of Robotics Research*, vol. 18, no. 11, pp. 1119–1128, 1999.

[15] K. Cai, C. Wang, J. Cheng, C. W. de Silva, and M. Q.-H. Meng, “Mobile robot path planning in dynamic environments: A survey,” *ArXiv*, vol. abs/2006.14195, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:220055835.

[16] S. Li, X. Xu, and L. Zuo, “Dynamic path planning of a mobile robot with improved q-learning algorithm,” in *2015 IEEE International Conference on Information and Automation*, 2015, pp. 409–414. DOI: 10.1109/ICInfA.2015.7279322.

[17] A. Loganathan and N. S. Ahmad, “A systematic review on recent advances in autonomous mobile robot navigation,” *Engineering Science and Technology, an International Journal*, vol. 40, p. 101 343, 2023, ISSN: 2215-0986. DOI: https://doi.org/10.1016/j.jestch.2023.101343. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098623000204.

[18] D. Helbing and P. Molnár, “Social force model for pedestrian dynamics,” *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, May 1995, ISSN: 1095-3787. DOI: 10.1103/physreve.51.4282. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.51.4282.

[19] J. van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935. DOI: 10.1109/ROBOT.2008.4543489.

[20] Y. F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292. DOI: 10.1109/ICRA.2017.7989037.

[21] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022. DOI: 10.1109/ICRA.2019.8794134.

[22] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International Conference on Machine Learning*, 2018, pp. 1582–1591.

[23] R. Cimurs, I. H. Suh, and J. H. Lee, “Goal-driven autonomous exploration through deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 730–737, 2022. DOI: 10.1109/LRA.2021.3133591.