A Survey on Deep Reinforcement Learning Methods for Autonomous Navigation Systems

Palveenraj Uma Kandan, Saad Harous

***Abstract*—Autonomous navigation systems are cutting-edge innovations that automate the locomotion of artificial agents to maximize efficiency and productivity. However, the dynamic nature of the environment and unpredictability in encountering obstacles, serve as a barrier to effective navigation. In response, deep reinforcement learning is gaining prominence within the field of artificial intelligence for offering solutions to navigate real-world conditions autonomously. Deep reinforcement learning seeks to enhance traditional reinforcement learning through integration of deep learning to automatically extract features from real-world input state and train artificial agents to generate a relevant action according to that. This survey begins with an overview of conventional methods utilized for autonomous navigation to shed light on their limitations. Then, it delves into basic principles of deep reinforcement learning, followed by a comprehensive review of state-of-the-art deep reinforcement learning methods widely applied to solve various challenges associated with autonomous navigation. By reviewing these methods, valuable insights are gained into the advancements shaping the future of autonomous navigation systems.**

***Index Terms*—Autonomous navigation systems, deep reinforcement learning, survey**

I. Introduction

AUTONOMOUS navigation involves systems or devices navigating and functioning on their own without direct human input or control [1]. It involves determining paths based on one's current position relative to the target destination, and selecting a path that avoids the obstacles along the way. Autonomous navigation systems are widely used in various fields, including robots such as mobile robots [2] and industrial robots [3], self-driving cars [4], controlling Unmanned Aerial Vehicles (UAV) [5], and Autonomous Underwater Vehicles (AUV) [6].

There are several benefits that explain why autonomous navigation systems are widely adopted. One advantage is that autonomous systems are used to minimize human errors, resulting in enhanced safety and reliability of operations. [1]. Second, autonomous vehicles optimize resource usage by selecting optimal routes that minimize fuel consumption and wear and tear on vehicles. This could help save costs and improve resource management. Furthermore, the utilization of these systems are primarily driven by its ability to enhance time efficiency. They execute task autonomously, without delays caused by human limitations [7].

Despite their numerous advantages, autonomous navigation systems also encounter several challenges. One such challenge is when multiple agents navigate within an unknown environment with limited information about their surroundings [8]. In addition, the dynamic nature of the environment presents a major problem for agents, as they may struggle when objects appear at random point of time [2]. Also, industrial robots may encounter difficulty in movement due to the dynamic nature of humans. Moreover, localization and mapping are issues where an agent's current position in the environment needs to be determined (localization) as well as a detailed map of the surrounding (mapping) needs to be created for effective navigation [9]. Obstacle avoidance poses a significant challenge for agents that need to reach their target destination safely without colliding with other agents and objects in their path [4].

To address these challenges, Deep Reinforcement Learning (DRL) techniques are employed. DRL utilizes the decision-making capabilities of Reinforcement Learning (RL) and feature extraction capabilities of Deep Learning (DL) from raw input information, to enable agents to learn optimal strategies by receiving rewards or penalties based on the performed actions. The aim is for the agents to learn a policy that maximizes cumulative rewards over time [10]. In the case of autonomous navigation systems, optimal navigation policies is learned through trial and error, aiming to adapt to diverse and dynamic environments. Therefore, allowing autonomous navigation systems to make informed decisions and enhancing their ability to navigate complex environment efficiently.

A number of surveys, ranging from 2019 to 2023, have explored in specific domains like visual navigation [10], multi-robot navigation [11] and motion planning for autonomous vehicles [12], all utilizing deep reinforcement learning (DRL). Yet, there is a gap in research regarding the diverse applications of autonomous navigation systems using DRL as a whole. Thus, in this paper, a survey on the applications of autonomous navigation systems leveraging DRL is provided. Additionally, a comprehensive research of the DRL based methods and their evaluation metrics are discussed. This facilitates researches and practitioners enhance, or expand upon existing techniques in a timely fashion.

The rest of the paper is outlined as follows: Section II describes the systematic process of collecting related literature. Section III presents the traditional methods employed to solve the challenges. The basics of RL, DL and DRL are covered in section IV. In section V, various applications of autonomous navigation system, leveraging DRL is provided (Fig. 1) Section VII describes various evaluation metrics used to asses the performance of DRL methods. Section VIII concludes the studies.

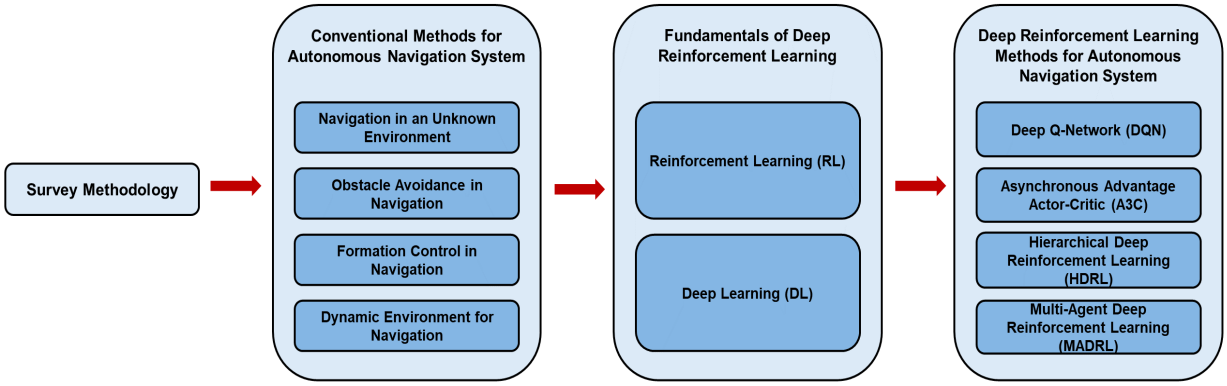

Fig. 1. Organization of the survey paper

II. Survey Methodology

To conduct a thorough survey of recent applications of deep reinforcement learning in autonomous navigation, a meticulous methodology was devised. Initially, a set of key terms was carefully selected, including ”reinforcement learning,” ”deep learning,” ”deep reinforcement learning,” ”autonomous navigation,” ”multi-agent reinforcement learning,” ”deep Q-network,” ”hierarchical deep reinforcement learning,” and ”asynchronous advantage actor-critic.” These keywords formed the basis for searches conducted across three prominent academic databases: IEEE Xplore, Google Scholar, and Springer. The search scope was limited to papers published between 2020 and 2024 to ensure relevance to current advancements. Each database was queried using combinations of the selected keywords to yield a comprehensive set of potential papers. Subsequently, a stringent selection process was implemented, initially screening papers based on their title and abstract to gauge relevance to the topic. Full-text assessments were then performed on promising candidates to determine their suitability for inclusion. Only those papers deemed highly relevant to recent applications of deep reinforcement learning in autonomous navigation were included in the final selection.

III. Conventional Methods for Autonomous Navigation Systems

A. Navigation in an Unknown Environment

A method known as centralized planning was used to assign multiple agents to reach a certain destination (target assignment) in an unknown environment. In this approach, each agent is assigned to a target before or during path planning. However, this can be time consuming and impractical for real-time applications, especially if the number of agents are significant. Distributed planning method involves modifying target assignment in response to ongoing communication between agents. Therefore, it is computationally efficient. However, this approach does not include collision avoidance algorithm. To address this problem, local planning techniques were implemented for each individual agent. Due to absence of global target assignment and comprehensive path planning, distributed planning approaches often lack optimality [8]. Traditional map-based technique like Rapidly Exploring Random Tress (RRT) and A* are algorithms used for exploration of the environments. Although these approaches are widely used, they necessitate an initial exploration stage. When dealing with complex environmental scenarios or input spaces with numerous dimensions, these methods continue to experience slow computational performance [13].

B. Obstacle Avoidance in Navigation

Prior studies employed different Passive-Avoidance Strategies (PAS), allowing the agents to avoid humans and obstacles passively without active engagement in the environment. Passive avoidance involves using sensors, algorithms, or predefined rules to detect and avoid potential collisions or conflicts while maintaining a safe distance from humans or other objects in its environment. However, the agents lacks adaptability due to rapidly changing environment where real-time adjustments are necessary [14], [15]. Other technique such as non-learning based techniques such as perception and avoidance were utilized to avoid collision and navigate the environment by moving the agents in the other direction. Due to the reactive nature of this technique, they could not anticipate and proactively address complex scenarios. In addition, RGB-D cameras were integrated to the agents as a visual input, which emits infrared light to measure the distance from each agent to the obstacles. However, RGB-D cameras has limited range to detect obstacles and sensitivity to environmental conditions [5]. A method known as traditional decentralized navigation systems were employed where each agent make independent decision based on the input states of the neighboring agents, potentially solving the issue of colliding with humans and object in real-world. Nevertheless, this method has two limitations. First, every agent within the simulated environment must share velocity data with the nearby agents. Second, conventional velocity obstacle algorithms for agents involve numerous adjustable parameters and require intricate designs for fine-tuning these parameters to suit various situations [16]. Other collision avoidance algorithm such as path planning priority was calculated for each agent

to avoid collision between other agents in the environment. The effectiveness of this algorithm is limited to specific environments because each environment has a unique arrangement of objects. As the environment changes or when agents are relocated to a different setting, the algorithm's effectiveness decreases because it's not familiar with the new environment. In this case, dynamic path planning is looked for, to re-evaluate the agents based on the assumption of the updated or the new environment [8].

C. Formation Control in Navigation

Existing studies have applied multiple standard techniques on Autonomous underwater vehicles (AUVs), to synchronize the movement and arrangement of multiple AUVs in relation to one another (formation control), accomplishing specific tasks. The method called Lagrangian mechanics principles were used to establish formation control in marine surface craft, using constraint functions as control laws. It adapts seamlessly for single vessel position control, showcasing versatility and robustness in simulations [17]. A robust control scheme for achieving time-varying formation of multiple AUVs was introduced. It incorporates a body-fixed coordinate transformation, additional control terms for AUV limitations, and a disturbance observer within Dynamic Surface Control (DSC) to maintain formation despite uncertainties [18]. Classic control methods work well, but they rely on certain assumptions like simplifying the system's dynamics and accurately knowing the model or its parameters. These methods often involve complex math, making them difficult to use with AUVs that have unpredictable factors like non linearity and unknown disturbances in unfamiliar environments [6].

D. Dynamic Environment for Navigation

A path planning algorithm for active simultaneous localization and mapping was introduced to enhance the agent's localization accuracy as it navigates through a dynamic environment by strategically revisiting previously observed positions. This is achieved by continuously refining the agent's localization while ensuring smooth, uninterrupted movement towards a specified goal position. This approach leverages the D* shortest path graph search algorithm, incorporating negative edge weights to account for localization uncertainty. Thus, enables the agent to calculate the shortest feasible path considering its evolving localization accuracy [19]. This approach has two drawbacks, including the time taken to generate the map increases as more data is needed and frequently updating the map to include all areas the agents need to navigate, especially if the configuration of the environment changes [20]. Other path planning methods that rely on static maps for navigation, leads to inefficiencies and potential failures as they do not account for real-time changes in the environment [2].

IV. Fundamentals of Deep Reinforcement Learning

A. Reinforcement Learning (RL)

Deep reinforcement learning is a revolutionary domain of machine learning that combines reinforcement learning and deep learning, to enhance the decision-making ability of an artificial intelligent agent. Reinforcement learning (Fig. 2) involves training the agent to optimize its performance by interacting with and receiving feedback from a dynamic environment. During reinforcement learning, the environment first provides its current state to the agent. Based on the current state, the agent takes an action which could be considered favorable or unfavorable by the environment, and given rewards or penalties respectively. This process continues until the agent achieves a desired performance through accumulation of rewards. Despite the constant interaction between the environment and agent, facilitate optimization, traditional reinforcement learning systems tend to perform poorly for tasks requiring high computational complexity which arise from high-dimensional input data. This hinders widespread application of reinforcement learning to solve complex real-world problems [21].

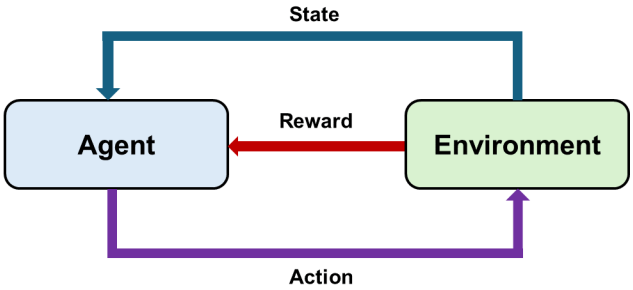

Fig. 2. Reinforcement Learning Workflow

B. Deep Learning (DL)

Deep reinforcement learning overcomes the limitations of traditional reinforcement learning by integrating deep learning to learn representations of large amount of raw data automatically, without prior feature engineering. Therefore, deep learning is considered as a powerful approach for tasks related to robotics, computer vision and natural language processing. The general architecture of a deep learning model is a multi-layer neural network that generally consists of an input layer, several hidden layers and an output layer (Fig. 3). The hidden layers contain hundreds of neurons which are the basic units of a neural network. The neurons are interconnected across different layers to transmit parameters and build a comprehensive feature representation of the input data. On top of that, this architecture facilitates the fine-tuning of the neural network through supervised backpropagation. This process is unique to neural networks, and is involved in updating the model parameters during training, to minimize the deviation between the actual and predicted data [10].

In recent years, deep learning has undergone tremendous advancement, hence accelerating significant progress in deep reinforcement learning. Unlike traditional reinforcement learning, deep reinforcement learning is capable of extracting meaningful information from high-dimensional state and action spaces efficiently and apply them to solve tasks in an unpredictable complex environment. The following section delves

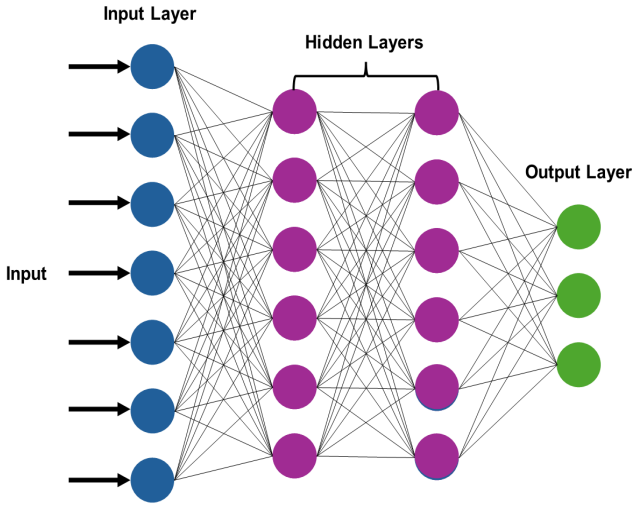

Fig. 3. Deep Learning Architecture

into current state-of-the-art deep reinforcement algorithms for autonomous navigation.

V. Deep Reinforcement Learning Methods for Autonomous Navigation Systems

A. Deep Q-Network (DQN)

DQN is a pioneering deep reinforcement learning method developed by Mnih et al. [22] that melds deep convolutional network with Q-learning. Conventional Q-learning method uses a Q table to store Q-values which are estimates of the expected cumulative reward an agent will receive by taking a particular action from a given state and following a certain policy thereafter. In the context of reinforcement learning, each state-action pair has an associated Q-value, which represents the utility or desirability of taking that action from that state. The Q-value function Q(s, a) maps a state-action pair (s, a) to a real number, indicating the expected total reward if the agent starts in state s, takes action a, and then continues to act optimally. However, as the state and action spaces grow larger, maintaining a Q-table becomes impractical due to the sheer size of the table. For instance, in problems with high-dimensional state and action spaces, the Q-table would become prohibitively large, requiring enormous memory resources. This limitation is particularly problematic in real-world applications where the state and action spaces are continuous or have a very large number of possible values. In such cases, storing and updating Q-values in a tabular form becomes infeasible.

To address this challenge, DQN leverages deep convolutional networks to approximate Q-values instead of maintaining a Q-table. The architecture of a DQN typically consists of three convolutional layers followed by one or more fully connected layers. This network is trained using a combination of experience replay and a target network to stabilize learning and improve sample efficiency. Experience replay stores agent experiences in a buffer, from which batches are randomly sampled for training, reducing correlation between updates. The target network, periodically updated with parameters from the main network, provides stable target Q-values during training, aiding convergence. This approach allows DQN to handle high-dimensional state spaces and action spaces, as the network can generalize across similar states and actions (Fig. 4). As a result, DQN could learn Q-values directly from raw sensory inputs, such as images or sensor data, thereby tackling complex real-world problems that are beyond the capabilities of traditional Q-learning methods [23].

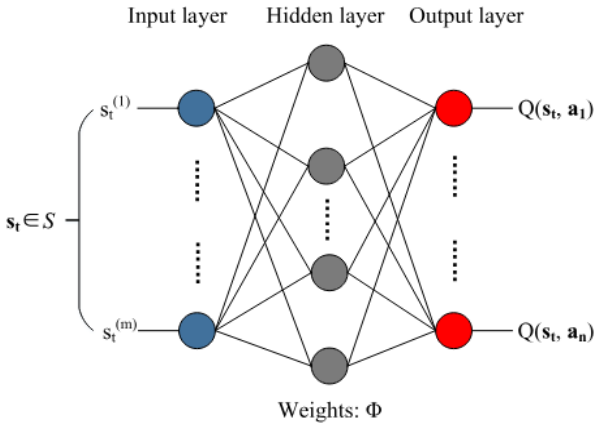

Fig. 4. Deep Q-Network Architecture [24]

Manikandan et al. [4] applied different DQN algorithms as a decision-making technique for the self-driving vehicle. The agent (vehicle) receives input from the RGB-D depth sensor, processes it through lane detection and obstacle detection algorithms, and then feeds this information into the DQN algorithms. The DQN algorithms then outputs action commands for steering, accelerating, and braking based on the current state of the environment and the learned Q-values. The results obtained from comparing different variants of the DQN algorithms (such as DQN, DDQN, and D3DQN) showed that the D3DQN variant had the fewest negative rewards. This suggests that the D3DQN algorithm performed better at avoiding collisions and navigating complex scenarios compared to the other variants.

Yang et al. [25] stated the challenge of enabling UAVs to navigate in changing environments, especially in congested airspace where capturing real-time global environment information such as the determining the geolocation of other UAVs and navigating through steep terrain. Previous approaches like traditional Q-learning have shown feasibility in static environments but face issues of overestimating action values in dynamic environments. To solve this, a framework based on a double deep Q-network with priority experience replay (DDQN-PER) have been proposed. This framework utilizes convolutional neural networks (CNNs) to process raw pixel inputs from the local environment and estimate future rewards for different actions. By setting up experimental scenarios that involves both stationary and dynamic obstacles, spanning from individual-agent to multi-agent navigation tasks, the model is trained and tested to demonstrate successful navigation and obstacle avoidance in dynamic environments. The ex-

perimental findings show that the proposed models enable UAVs to successfully reach their goals in unfamiliar dynamic environments.

Naranjo et al. [26] addressed the limitations of existing control systems for mobile robots, mainly on using predetermined routes that lacks real-time capabilities for avoiding obstacles and adjusting trajectories. To overcome these challenges, DQN method was utilized. This method involves training the agent in a simulated environment, where it learns to navigate and avoid obstacles in real-time while optimizing rewards. Through experimentation and parameter tuning, the proposed method performed well, showcasing its ability to enhance autonomous learning in mobile robots. This approach leverages randomized training conditions within the simulated environment, allowing the DQN network to compute complex functions compared to traditional methods, thereby significantly enhancing the autonomous capabilities of mobile robots.

Wang et al. [27] stated the challenge of enhancing autonomous navigation and collision avoidance of UAVs using on-board cameras when radar and communication are unavailable or disrupted. To solve this, current DQN method was optimized by introducing a Faster R-CNN model. The images captured from on-board cameras are passed to the proposed framework as input, to extract valid information that can help direct the agents in avoiding obstacles. Additionally, due to alterations in scenario features, the experience replay method within DQN is no longer effective, leading to the agent's inability to achieve improved training outcomes. Therefore, Data Deposit Mechanism (DDM) was proposed to further enhance the training process of the proposed model. This approach shows that FRDDM-DQN surpasses other methods such as FRDQN, FR-DDQN, FR-Dueling DQN, YOLO-based YDDM-DQN, and FR-ODQN regarding autonomous navigation, avoiding collisions, and adapting to various scenarios. This highlights its efficacy for UAV operations in difficult conditions.

B. Asynchronous Advantage Actor-Critic (A3C)

A3C is an advanced reinforcement learning algorithm also introduced by Mnih et al. [28] to address the limitations of conventional policy gradient methods. These traditional methods typically update policies in an on-policy manner, which can lead to slow convergence and low data efficiency. To overcome these challenges, A3C employs asynchronous training with multiple threads. In A3C, the agent interacts with the environment simultaneously in multiple threads, each thread exploring its own trajectory of interactions. This asynchronous data collection process significantly speeds up sample collection, as each thread explores different parts of the environment. After collecting samples, each thread independently completes training and updates the global model parameters asynchronously. This decentralized training approach allows for parallelization and efficient use of computational resources [21].

Additionally, A3C incorporates the advantage function to measure the quality of the policy, which helps balance bias and variance in learning [10]. The advantage function measures the advantage of taking a particular action in a given state compared to the average value of that state. When the value of the advantage function is positive, it indicates that the action has a higher expected return than the average value of the state, suggesting that the action is advantageous. In this case, the probability of selecting that action is increased, encouraging the agent to explore actions that have shown promise in the past. Conversely, the function value is negative, it means that the action has a lower expected return than the average value of the state, suggesting that the action is disadvantageous. In this scenario, the probability of selecting that action is decreased, discouraging the agent from choosing actions that are less likely to lead to favorable outcomes.

Furthermore, A3C incorporates the policy's entropy into the training process [10]. Entropy measures the uncertainty or randomness in the probability distribution of actions. By incorporating entropy into the advantage function, A3C encourages exploration by penalizing overly deterministic policies. A higher entropy implies a more uncertain policy, which prevents the algorithm from prematurely converging to a sub-optimal policy. Therefore, A3C seeks to strike a balance between maximizing expected returns (through the advantage function) and exploring diverse actions (through entropy regularization), ultimately facilitating efficient and effective reinforcement learning.

Caruso et al. [29] and Kastner et al. [30] employed two RL algorithms, DQN and A3C which enable save navigation for robots in crowded environments. DQN uses Q-function that estimates the average discounted return, representing the sum of future rewards obtained by following a policy, starting from the initial state to the goal state. This method iteratively improves the estimated Q-function, converging towards the optimal policy. This is achieved by approximating the Q-function using a neural network and updating its weights to minimize the loss function. On the other hand, the A3C algorithm is a policy gradient method, mainly focused on optimizing the policy by computing gradients to update its weights and maximizing the objective function. The objective function represents the average return achieved by the policy through sampled trajectories from the environment. A3C helps lower variation by looking at the benefit of an action compared to a baseline given by a critic function, which estimates the value of a state for the policy. A3C estimates this value in real-time without considering previous rewards.

Wang et al. [31] utilized A3C algorithm to solve collision avoidance problem for Unmanned Underwater vehicle (UUVs) in unknown environments. A part of A3C known as asynchronous approach was adopted that allows multiple threads to work simultaneously without synchronizing their data generation. The method collects experiences learned

TABLE I
SUMMARY OF THE PAPERS

Reference	DRL Methods	Applications
[4], [25], [26], [27]	Deep Q-Network	self-driving vehicle, UAVs, mobile-robots
[29], [30], [31]	A3C	mobile-robots, UUVs
[8], [5], [32]	HADRL	UAVs, mobile-robot
[16], [33]	MADRL	mobile-robots

by each sub-model (thread-specific Actor-Critic models) and updates its parameters accordingly. Furthermore, the algorithm utilizes multi-threading for training. Each thread corresponds to different training environments, ensuring that they do not interfere with each other. Different training environments for each thread accelerate learning by exposing the models to diverse experiences. After interacting with the environment, each thread updates and shares the learned experiences to the model. Thus, improving the learning efficiency.

C. Hierarchical Deep Reinforcement Learning (HDRL)

Hierarchical Reinforcement Learning (HRL) is designed to tackle challenges where rewards are sparse and require exploration. Existing HRL techniques includes dividing complex tasks into simple sub-tasks, eventually completing the original task by addressing these sub-tasks. They relied on manual selection of sub-tasks which required domain expertise. However, with the integration of deep learning into HRL, significant advancements have been made, enhancing the capability of HRL to handle large state and action spaces.

HDRL has emerged as a prominent area of research within the broader field of Deep Reinforcement Learning (DRL). HDRL leverages the deep learning to create hierarchical structures that aid in solving complex tasks more effectively. By combining hierarchical decomposition with deep learning's learning capabilities, HDRL has the potential to tackle a wide range of challenging problems in various domains, especially in autonomous navigation systems [21].

Jin et al. [8] tackles Multiagent Navigation to Unassigned Multiple targets (MNUM), a complex scenario where multiple agents need to reach unassigned targets autonomously by avoiding the obstacles at the same time, minimizing the time spent. Due to the vast number of policies and the dynamic environment caused by concurrent learning, traditional MADRL methods struggle with MNUM. Thus, HIST-MADRL framework was introduced to solve this problem using a hierarchical policy setup and a stable MADRL approach.

Hierarchical policy setup consists of two types of policies, high-level policy and low-level policy. The high-level policy dynamically selects the target destinations by observing all the positions, including target destinations and agents (global observations), whereas low-level policy determines the movement direction of the agents based on the distance between the agents and the surrounding obstacles (local observations). In this approach, target selection comes first, followed by policy learning. Agents move within a defined range toward their selected target unless they encounter obstacles that require them to deviate from their path. Therefore, reducing the number of policies greatly.

Stable MADRL method was integrated to each agent that estimates the action states of other agents to improve policy learning stability. Once an agent estimates the possible actions of other agents, it can incorporate this information into its own policy learning process. This allows the agent to consider how other agents might behave and adjust its own actions accordingly. Considering the potential actions of others allows the agent to develop a policy that remains stable even as the environment changes due to other agents' learning.

Liu et al. [5] addressed the limited performance in different environments and the need for improved long-term task handling. To tackle these issues, they proposed a HDRL framework called HTARADrQ. This framework comprises a master policy and multiple sub-policies. The master policy selects sub-tasks for the primary navigation task, guiding the UAV through a hierarchical structure of learned rules. Each sub-policy handles a specific sub-task, enabling the agent to break down complex navigation into manageable steps. The recurrent network and temporal attention mechanism enhance algorithm performance, while the averaged estimation function improves robustness.

Zhu et al. [32] proposed a HDRL framework, consisting of a low-level policy that is responsible for quick motion and obstacle avoidance, a high-level policy aimed at enhancing safety further. The framework utilizes a sub-goal, selected as a way point along the path to the final goal, to reduce the state space and avoid sparse rewards. By incorporating deep neural networks for feature extraction, the framework achieves a more efficient representation of the surroundings. The reward function encourages safe navigation and efficient progress towards the target. The high-level policy maintains motion efficiency by learning to avoid collisions.

VI. Multi-Agent Deep Reinforcement Learning (MADRL)

MADRL is a field that focuses on developing algorithms and techniques for training multiple agents to interact and collaborate in complex environments. Unlike traditional reinforcement learning, which typically involves a single agent learning to maximize its own reward, MADRL involves multiple agents learning to optimize their collective performance while considering the actions and strategies of other agents. Deep reinforcement learning (DRL) methods are employed in MADRL to handle high-dimensional state and action spaces, enabling agents to learn representations of their environments and make decisions based on these learned representations [34].

Xue et al. [16] addressed the safe navigation of multiple unmanned aerial vehicles (UAVs) in complex and unknown 3D environments, where traditional decentralized navigation systems struggle due to partial observation and environmental instability. To solve the problem, Multi-Agent Recurrent Deterministic Policy Gradient (MARDPG) algorithm was proposed, which builds upon the deterministic policy gradient algorithm. MARDPG combines decentralized execution with centralized training, allowing each UAV to estimate policies of others while training centrally, thereby overcoming slow convergence caused by unstable environments. The algorithm utilizes a shared centralized critic network to estimate joint action-values, enabling coordination without direct communication between UAVs during execution. By incorporating LSTM networks for historical trajectory information and optimizing the deterministic policy through a gradient descent method, MARDPG enhances decision-making and navigation in multi-UAV scenarios, demonstrating superior convergence and efficacy over existing deep reinforcement learning approaches.

Tan et al. [33] introduced Macro Action Decentralized Exploration Network(MADE-Net) using multi-agent deep reinforcement learning (DRL). MADE-Net is designed to address the communication dropouts by considering high-level teammate intentions during action selection. The architecture comprises Centralized Training and Decentralized Execution sub-systems, aiming to learn team behaviors centrally while executing exploration locally. The Macro Observation module processes sensory data into high-level representations, including localization, teammate detection, mapping, and goal extraction. The Centralized Exploration Policy (CEP) and Decentralized Exploration Policy (DEP) use Deep Double Recurrent Q Networks (DDRQN) for decision-making, focusing on spatial context learning and incorporating decentralized planning. Policy Optimization involves training both policies simultaneously with centralized and decentralized loss functions, reward functions, and experience replay.

VII. Evaluation Metrics for Deep Reinforcement Learning

Success rate is a metric used to evaluate the effectiveness of an agent in accomplishing its task within a specified criterion. It represents the percentage of episodes or trials in which the agent successfully achieves its goal, such as reaching a certain score or completing a task. Success criteria, multiple episodes or trials of the environment needs to be defined to calculate the success rate. Moreover, improving the success rate over time is essential for assessing the progress of the DRL algorithm and making adjustments to enhance its performance [8].

Collision rate metric refers to the frequency at which agents or objects collide with each other within a simulated environment. This metric is crucial for assessing the effectiveness and safety of an agent's policies, especially in domains like autonomous driving or robotics where collisions can have significant consequences. The collision rate is often used as a performance measure to evaluate the robustness and reliability of the agent's behavior. It can be calculated by dividing the total number of collisions that occur during a simulation by the total simulation time, yielding collisions per unit time. This rate gives insights into how frequently the agent encounters collisions, helping researchers and developers fine-tune algorithms and policies to minimize such incidents and improve overall performance [32].

VIII. Conclusion

Deep reinforcement learning is a promising approach to improve autonomous navigation systems in terms of training speed, performance and use of computational resources. Unlike conventional methods, deep reinforcement learning enables autonomous navigation systems to respond properly to unexpected and complex scenarios. This paper presented the principles and applications of several deep reinforcement learning methods in autonomous navigation. In addition, recent literature proposing variants of deep reinforcement learning methods for further enhancement was highlighted. Overall, this paper serves as a pivot for researchers to gain a comprehensive understanding of current advancements in deep reinforcement learning for various domains within autonomous navigation.

References

[1] D. P. Watson and D. H. Scheidt, "Autonomous systems," *Johns Hopkins APL Technical Digest*, vol. 26, no. 4, pp. 368–376, 2005.

[2] A. N. Kumaar and S. Kochuvila, "Mobile service robot path planning using deep reinforcement learning," *IEEE Access*, 2023.

[3] M. Shamout, R. Ben-Abdallah, M. Alshurideh, H. Alzoubi, B. al Kurdi, and S. Hamadneh, "A conceptual model for the adoption of autonomous robots in supply chain and logistics industry," *Uncertain Supply Chain Management*, vol. 10, no. 2, pp. 577–592, 2022.

[4] N. S. Manikandan, G. Kaliyaperumal, and Y. Wang, "Ad hoc-obstacle avoidance-based navigation system using deep reinforcement learning for self-driving vehicles," *IEEE Access*, 2023.

[5] Z. Liu, Y. Cao, J. Chen, and J. Li, "A hierarchical reinforcement learning algorithm based on attention mechanism for uav autonomous navigation," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[6] B. Hadi, A. Khosravi, and P. Sarhadi, "Adaptive formation motion planning and control of autonomous underwater vehicles using deep reinforcement learning," *IEEE Journal of Oceanic Engineering*, 2023.

[7] P. J. Antsaklis, K. M. Passino, and S. J. Wang, "An introduction to autonomous control systems," *IEEE Control Systems Magazine*, vol. 11, no. 4, pp. 5–13, 1991.

[8] Y. Jin, S. Wei, J. Yuan, and X. Zhang, "Hierarchical and stable multiagent reinforcement learning for cooperative navigation control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 1, pp. 90–103, 2021.

[9] E. E. Montero, H. Mutahira, N. Pico, and M. S. Muhammad, "Dynamic warning zone and a short-distance goal for autonomous robot navigation using deep reinforcement learning," *Complex and Intelligent Systems*, vol. 10, no. 1, pp. 1149–1166, 2024.

[10] F. Zeng, C. Wang, and S. S. Ge, "A survey on visual navigation for artificial agents with deep reinforcement learning," *IEEE Access*, vol. 8, pp. 135 426–135 442, 2020.

[11] J. Orr and A. Dutta, "Multi-agent deep reinforcement learning for multi-robot applications: A survey," *Sensors*, vol. 23, no. 7, p. 3625, 2023.

[12] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 740–759, 2020.

[13] L. Zhang, Z. Hou, J. Wang, Z. Liu, and W. Li, "Robot navigation with reinforcement learned path generation and fine-tuned motion control," *IEEE Robotics and Automation Letters*, 2023.

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

[15] M. Kamezaki, R. Ong, and S. Sugano, "Acquisition of inducing policy in collaborative robot navigation based on multiagent deep reinforcement learning," *IEEE Access*, vol. 11, pp. 23 946–23 955, 2023.

[16] Y. Xue and W. Chen, "Multi-agent deep reinforcement learning for uavs navigation in unknown complex environment," *IEEE Transactions on Intelligent Vehicles*, 2023.

[17] I.-A. F. Ihle, J. Jouffroy, and T. I. Fossen, "Formation control of marine surface craft: A lagrangian approach," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 922–934, 2006.

[18] J. Li, J. Du, and W.-J. Chang, "Robust time-varying formation control for underactuated autonomous underwater vehicles with disturbances under input saturation," *Ocean Engineering*, vol. 179, pp. 180–188, 2019.

[19] I. Maurović, M. Seder, K. Lenac, and I. Petrović, "Path planning for active slam based on the d* algorithm with negative edge weights," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 8, pp. 1321–1331, 2017.

[20] E. Palacios-Morocho, S. Inca, and J. F. Monserrat, "Multipath planning acceleration method with double deep r-learning based on a genetic algorithm," *IEEE Transactions on Vehicular Technology*, 2023.

[21] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064–5078, 2024, iD: 1.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[23] Z. He, K. P. Tran, S. Thomassey, X. Zeng, J. Xu, and C. Yi, "Multi-objective optimization of the textile manufacturing process using deep-q-network based multi-agent reinforcement learning," *Journal of Manufacturing Systems*, vol. 62, pp. 939–949, 2022.

[24] J. Zhao, Y. Zhang, Y. Nie, and J. Liu, "Intelligent resource allocation for train-to-train communication: A multi-agent deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 8032–8040, 2020.

[25] Y. Yang, K. Zhang, D. Liu, and H. Song, "Autonomous uav navigation in dynamic environments with double deep q-networks," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–7.

[26] J. Escobar-Naranjo, G. Caiza, P. Ayala, E. Jordan, C. A. Garcia, and M. V. Garcia, "Autonomous navigation of robots: Optimization with dqn," *Applied Sciences*, vol. 13, no. 12, p. 7202, 2023.

[27] W. Fei, Z. Xiaoping, Z. Zhou, and T. Yang, "Deep-reinforcement-learning-based uav autonomous navigation and collision avoidance in unknown environments," *Chinese Journal of Aeronautics*, vol. 37, no. 3, pp. 237–257, 2024.

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[29] M. Caruso, E. Regolin, F. J. C. Verdù, S. A. Russo, L. Bortolussi, and S. Seriani, "Robot navigation in crowded environments: A reinforcement learning approach," *Machines*, vol. 11, no. 2, p. 268, 2023.

[30] L. Kästner, Z. Shen, C. Marx, and J. Lambrecht, "Autonomous navigation in complex environments using memory-aided deep reinforcement learning," in *2021 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2021, pp. 170–175.

[31] H. Wang, W. Gao, Z. Wang, K. Zhang, J. Ren, L. Deng, and S. He, "Research on obstacle avoidance planning for uuv based on a3c algorithm," *Journal of Marine Science and Engineering*, vol. 12, no. 1, p. 63, 2023.

[32] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962–4971, 2022.

[33] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep reinforcement learning for decentralized multi-robot exploration with macro actions," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 272–279, 2022.

[34] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 895–943, 2022.