AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-2485

Autonomous Landing of eVTOL Vehicles for Advanced Air Mobility via Deep Reinforcement Learning

Sabrullah Deniz*, Yufei Wu†, and Zhenbo Wang‡
The University of Tennessee, Knoxville, Tennessee, 37996, USA

The development of Advance Air Mobility (AAM) is revolutionizing air transportation by introducing autonomous aircraft, particularly electric vertical takeoff and landing (eVTOL) vehicles, for efficient movement of people and cargo in urban and remote areas. The majority of aircraft operating in the AAM system are expected to be eVTOL vehicles, which will utilize vertiports as their base of operations. However, unlike traditional airports and general aviation aircraft, the placement of vertiports within densely populated urban areas poses distinct challenges in effectively managing the air traffic flow associated with these facilities. The complexity of this challenge is amplified as the frequency of scheduled landings and take-offs increases. This paper focuses on the critical aspect of autonomous landing for eVTOLs in the context of AAM. Specifically, we propose a deep reinforcement learning (DRL) approach to enhance the reliability and safety of eVTOL autonomous landing, considering various algorithms and scenarios. By leveraging and customizing DRL algorithms for the eVTOL landing task, the accuracy and robustness of the landing operations are enhanced. To evaluate the effectiveness of our approach, extensive simulations are conducted, and the performance of our algorithm is analyzed under different scenarios and environmental conditions. We utilize the AirSim simulation environment integrated with Unreal Engine (UE), providing a realistic virtual platform for testing and validating the performance of our algorithm. Extensive simulations and evaluations conducted within the AirSim-UE environment provide compelling evidence of the exceptional landing precision and efficiency achieved by the proposed method.

I. Introduction

The exponential growth of the global population has led to increasing demands on ground infrastructure, resulting in severe traffic congestion in urban areas. To address this challenge, novel aviation concepts and technologies have been rapidly developed to provide efficient and sustainable mobility solutions. One such concept is Urban Air Mobility (UAM), which utilizes small, highly automated aircraft operating at lower altitudes to transport people and cargo within urban and suburban areas. UAM aims to utilize the airspace above cities as a practical and congestion-free transportation option [1]. Vertical take-off and landing (VTOL) aircraft, including electrically propelled VTOL (eVTOL) aircraft, are key technologies being explored to address ground traffic congestion and provide efficient transportation options.

UAM represents a revolutionary technology, especially if information-driven platforms like ride-hailing apps used by transportation network companies such as Lyft or Uber are utilized to connect service providers with immediate demand. The potential for large-scale aerial operations within our cities has never been more tangible, as demonstrated by the fact that in 2019, there were more than 1,000 test flights of full-size eVTOL aircraft [2]. Moreover, as of March 2020, at least 12 eVTOL aircraft were in the process of obtaining certification from the U.S. Federal Aviation Administration (FAA) [3]. UAM represents a paradigm shift in urban transportation, utilizing traditional helicopters, VTOL aircraft, and eVTOL aircraft as present and emerging technologies [2]. By utilizing the third dimension of the sky, UAM aims to connect people to cities and urban areas more efficiently, bypassing ground congestion and offering an innovative form of passenger transport and on-demand deliveries within urban areas [4]. However, to fully realize the potential of UAM, it is crucial to establish suitable VTOL ground infrastructures, such as vertiports and skyports, that facilitate safe and efficient operations [5].

While UAM presents exciting possibilities for urban transportation, one of the critical challenges lies in enabling precise and efficient landing of eVTOL vehicles on vertiports. As the number of eVTOLs operating in urban airspace increases, coordination and management of landing operations become crucial to ensure safety and minimize

*PhD Student, Department of Mechanical, Aerospace, and Biomedical Engineering, Student Member AIAA, sdeniz@vols.utk.edu
†PhD Candidate, Department of Mechanical, Aerospace, and Biomedical Engineering, Student Member AIAA, ywu86@vols.utk.edu
‡Assistant Professor, Department of Mechanical, Aerospace, and Biomedical Engineering, Senior Member AIAA, zwang124@utk.edu

congestion. Additionally, the landing phase itself poses unique challenges, including precise navigation in complex urban environments and the need to avoid obstacles during descent and landing [6]. Therefore, it is essential to develop advanced control strategies that can enable autonomous landing capabilities for eVTOLs in the UAM ecosystem.

Advanced control and optimization algorithms, such as convex optimization and model predictive control (MPC), have been studied in aviation and AAM applications to regulate the flight and maneuvering of aircraft [7–11]. However, these methods may fall short when it comes to more complex maneuvers that require long prediction horizons and adaptive control in uncertain environments. Landing an eVTOL vehicle on a vertiport landing mark involves precise positioning, obstacle avoidance, and compensation for environmental disturbances, making it a challenging task that demands sophisticated control strategies. To address these challenges, researchers have turned to the field of deep reinforcement learning (DRL), which combines deep neural networks with reinforcement learning techniques to enable autonomous decision-making in complex environments. DRL has shown remarkable success in various domains, including robotics, gaming, and control systems [12]. By leveraging the power of DRL, it becomes possible to develop intelligent control systems that can learn optimal landing policies through interaction with the environment, without the need for explicit human supervision.

In recent years, DRL algorithms have emerged as powerful tools for optimizing eVTOL autonomous operations [13, 14]. DRL algorithms, such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO), and Asynchronous Advantage Actor-Critic (A3C), have shown success in various autonomous systems from multiple domains and hold great potential for enhancing the performance, safety, and reliability of eVTOL landings. DQN combines deep neural networks with Q-learning to approximate the optimal action-value function, allowing eVTOLs to learn the landing strategies through interaction with the environment [12]. PPO directly optimizes the policy function to maximize cumulative reward and has demonstrated impressive performance in robotic control tasks [15]. TRPO provides stability guarantees during the policy update process and offers high sample efficiency, making it suitable for robust and reliable eVTOL landings [16]. A3C, with its parallel agent approach, enhances exploration and convergence speed, enabling efficient and reliable landings in diverse scenarios [17]. By leveraging these DRL algorithms, researchers can develop intelligent control systems that autonomously learn optimal landing policies, adapt to changing conditions, and navigate the challenges of complex urban environments.

In addition to the development of highly efficient algorithms, realistic simulation environments are also crucial to facilitate the validation, evaluation, and simulation of control strategies. AirSim, a simulation platform developed by Microsoft, provides researchers with a configurable and immersive environment for testing and validating autonomous systems [18]. When integrated with Unreal Engine (UE), AirSim enhances the capabilities for developing and fine-tuning control and perception systems specifically for eVTOLs operating in complex urban scenarios [19]. This combination of AirSim and UE enables researchers to iteratively refine and optimize eVTOL landing algorithms, ensuring safety, precision, and efficiency in real-world operations.

In this paper, we explore the significant promise and potential of DRL algorithms, including DDPG, A3C, and DQN for advancing autonomous eVTOL landing capabilities. Coupled with realistic simulation environments such as AirSim integrated with UE, we will develop and refine control strategies that optimize the landing performance and enhance the safety and reliability of eVTOL operations in complex urban environments for future UAM applications.

II. Related Work

In recent years, considerable research has been conducted regarding the design, development, and operation of autonomous Unmanned Aerial Vehicles (UAVs). The problem of autonomous landing for UAVs on both static and moving platforms has gained significant attention due to its importance in real-world applications [20][21]. Previous works have focused on various aspects of UAV autonomous landing, including perception and relative pose estimation [22], trajectory optimization [23–25], and coupled methods such as Image-Based Visual Servoing (IBVS) [26]. These studies have contributed to addressing specific challenges and enhancing the landing capabilities of UAVs.

When the relative state of vehicles is assumed to be known, various popular techniques have been employed for control maneuvers. These techniques include different types of guidance and rendezvous laws [24], sometimes supplemented with velocity controllers to expedite the approaching phase [27]. To achieve a desired meeting point, incorporating feedforward inputs allows for quicker response to track following errors [28], and optimal rendezvous trajectories can be determined, taking wind disturbances into consideration [23]. Aggressive landing from relatively short distances has been accomplished using PID controllers [27][22], while an adaptive control schema offers improved robustness [29]. In another research to address the challenging task of landing on moving inclined platforms, a discrete-time non-linear model predictive controller (MPC) was developed, optimizing both trajectories and landing

time [30].

In addition to the traditional control approaches, innovative bio-inspired strategies and intelligent control methods have shown promising results in UAV autonomous landing [31]. Bio-inspired strategies utilizing a time-to-contact (TTC) indicator have demonstrated successful performance in real-world scenarios. Machine learning-based methods, including neural network backpropagation controllers [32] and classical discrete reinforcement learning approaches [33], have also been explored. The authors in [33] applied a Least-Squares Policy Iteration (LSPI) algorithm to land a UAV on a static platform using a camera as the main sensor, demonstrating successful landing maneuvers in a simulated environment.

Deep learning strategies have shown promise in UAV indoor navigation tasks, with convolutional neural networks (CNNs) being utilized to map images to high-level behavior directives [34][35]. CNN-based methods have successfully mapped image inputs to discrete actions, enabling UAVs to perform navigation tasks [35]. Deep learning techniques have also been applied to low-level motion control, where learning control policies from imperfect sensor data have been investigated. Model predictive controllers (MPCs) and pre-trained CNN models have been adapted to enable UAVs to follow obstacle-free trajectories and navigate through environments with obstacles [36].

In the context of eVTOL autonomous landing, the application of DRL algorithms is an emerging area of research [37]. While previous works have primarily focused on UAV autonomous landing on static and moving platforms, the principles and methodologies can be extended to address the specific challenges associated with eVTOLs and vertiport landing for single-vehicle and multi-vehicle scenarios. The current operation of general aviation aircraft is handled by Air Traffic Control (ATC) [38]. A vertiport serves as an area designated for eVTOLs to perform take-off, landing, and battery charging [39]. When it comes to eVTOLs, the volume of vehicles entering or leaving a vertiport can range from hundreds to thousands per hour [40]. This poses a significant challenge in effectively controlling the landing and take-off (L/TO) of these aircraft, raising concerns about safety and regulation. The conventional First-come, First-served concept falls short in handling uncertainties and emergencies.

To address the above challenges and issues, our paper proposes a solution for regulating eVTOLs within the vertiport zone while ensuring their safety. Given the multi-dimensional nature of the problem, a novel 3D simulation environment incorporating realistic physics and path planning primitives is developed. Importantly, our simulator runs at an accelerated speed, enabling efficient data collection for learning purposes. Two distinct learning-based algorithms were trained and compared against each other. Prior works discussing the modeling of vertiports [41] and eVTOLs [42] exist. However, our focus in this paper is not on designing the vertiport or eVTOLs themselves. Instead, we consider the vertiport as a helipad and the UAV as the eVTOL, as the concept can be applied similarly. In this paper, we build upon the existing research and apply DRL algorithms to address the autonomous landing problem for eVTOL vehicles. Specifically, we investigate the effectiveness of DRL algorithms, such as DQN, DDPG, and A3C, in achieving precise and reliable landings on vertiports. We leverage the advancements in perception systems, including vision-based sensors, to provide input to the DRL algorithms and enable accurate landing maneuver generation.

Furthermore, we utilize the AirSim simulation platform with UE, to create a realistic and configurable environment for testing and evaluating the performance of our proposed DRL-based landing controller. This simulation environment allows us to conduct extensive experiments and analyze the impact of different algorithmic choices and environmental factors on the eVTOL autonomous landing performance. By combining the insights from previous works on UAV autonomous landing and the advancements in DRL techniques, we aim to contribute to the development of precise and reliable autonomous landing capabilities for eVTOLs in the context of UAM for both single-vehicle and multi-vehicle landing tasks.

III. Methodology and Preliminaries

In this section, we present the methodology and preliminaries of our approach for eVTOL autonomous landing using DRL algorithms. We begin by introducing the reinforcement learning framework and its application to robotic problems. Then, we describe the specific algorithms employed in our research, including Deep Deterministic Policy Gradient (DDPG), Actor-Critic, and Deep Q-Network (DQN).

A. Reinforcement Learning

Reinforcement learning is a framework where an agent interacts with an environment, aiming to find the best action for each state to maximize the accumulated reward [43]. The agent modifies its behavior, known as a policy, based on observed states, taken actions, and received rewards. The goal is to find an optimal policy that maximizes the expected cumulative reward over time.

In the standard reinforcement learning framework, the agent learns a policy that maps states, denoted as $s \in S$, to actions, denoted as $a \in A$, where S represents the state space and A represents the action space. The dynamics of the agent-environment interaction are represented by the transition probability model $p(s_{t+1}|s_t, a_t)$, which describes the probability of transitioning to the next state s_{t+1} given the current state s_t and action a_t. At each time step, the agent selects an action based on its policy, and observes a reward $r(s_t, a_t)$ from the environment.

The agent's objective is to find the optimal policy π^* that maximizes the value function, denoted as $V^\pi(s_t)$ or $Q^\pi(s_t, a_t)$, depending on whether the policy is deterministic or stochastic. The value function represents the expected cumulative reward starting from a state s_t under policy π. The optimal policy π^* is defined as the one that maximizes the value function, as shown in Equation 1:

$$\pi^* = \arg\max_\pi V^\pi(s_t) = \arg\max_\pi Q^\pi(s_t, a_t) \tag{1}$$

B. Actor-Critic

The Actor-Critic algorithm is a general framework that combines both policy-based and value-based methods. It consists of two components: an actor network and a critic network [44]. The actor network is responsible for selecting actions based on the current state, while the critic network evaluates the value of the selected actions. The actor network is updated using policy gradients to maximize the expected return, while the critic network is updated to minimize the temporal difference error between the estimated value and the observed reward.

The actor network updates the policy parameters θ_{actor} using the following gradient ascent rule:

$$\nabla_{\theta_{\text{actor}}} J \approx \mathbb{E}\rho \left[\nabla\theta_{\text{actor}} \log \pi(s) Q(s, a)\right] \tag{2}$$

The critic network updates the value function parameters θ_{critic} by minimizing the squared difference between the estimated value and the observed reward.

C. Deep Deterministic Policy Gradient (DDPG)

DDPG is an actor-critic algorithm designed to handle continuous control problems in reinforcement learning. It extends the standard policy gradient methods to continuous action spaces. In DDPG [45], an actor network approximates the deterministic policy $\mu(s)$, mapping states to actions, while a critic network approximates the action-value function $Q(s, a)$. The actor network is updated to maximize the expected action-value obtained from the critic network.

The DDPG algorithm updates the actor and critic networks using the following loss functions:

Actor Loss:

$$L_{\text{actor}} = -\mathbb{E}\left[Q(s, \mu(s))\right] \tag{3}$$

Critic Loss:

$$L_{\text{critic}} = \mathbb{E}\left[\left(r + \gamma Q(s', \mu'(s')) - Q(s, a)\right)^2\right] \tag{4}$$

where s is the current state, a is the action chosen by the actor network, s' is the next state, r is the reward obtained from the environment, $\mu(s)$ is the action chosen by the actor network given state s, $\mu'(s')$ is the action chosen by the target actor network given state s', γ is the discount factor, and $Q(s, a)$ is the action-value function approximated by the critic network.

During training, the actor and critic networks are updated using gradient descent optimization algorithms, such as Adam or RMSprop, to minimize the respective loss functions. To ensure stability, the DDPG algorithm utilizes a separate target network for calculating the target action-value. The target networks are periodically updated with the weights from the main networks to provide more stable value estimates.

D. Deep Q-Network (DQN)

Deep Q-Network is a value-based reinforcement learning algorithm that combines deep neural networks with Q-learning [46]. It approximates the action-value function $Q(s, a)$ using a neural network and learns from observed transitions (s, a, r, s'). The DQN algorithm updates the Q-network using the following loss function:

$$L = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{\text{target}}) - Q(s, a; \theta)\right)^2\right] \tag{5}$$

where s is the current state, a is the action chosen by the agent, r is the reward obtained from the environment, s' is the next state, θ are the parameters of the Q-network, θ_{target} are the target network parameters, and γ is the discount factor.

The target network parameters θ_{target} are periodically updated to stabilize the learning process. DQN utilizes an experience replay buffer to store and randomly sample past experiences, enabling more efficient learning from correlated data.

In our research, we adapt these algorithms and techniques to address the specific problem of eVTOL autonomous landing. By combining the power of DRL with continuous control and value-based methods, we aim to enable precise and robust autonomous landings in various environments.

E. Formulations

In the eVTOL autonomous landing problem, the state s_t represents the current state of the eVTOL, which includes variables such as position, velocity, orientation, and environmental factors like wind speed and obstacles. The action a_t represents the control inputs, such as throttle, pitch, roll, and yaw commands.

For the DDPG algorithm, the actor network approximates the deterministic policy $\mu(s)$, which maps states to actions. The critic network approximates the action-value function $Q(s, a)$. The actor network is updated using gradient ascent to maximize the expected action-value obtained from the critic network. The critic network is updated to minimize the temporal difference error between the estimated value and the observed reward.

In the DDPG formulation, the actor loss L_{actor} aims to maximize the expected action-value:

$$L_{\text{actor}} = -\mathbb{E}\left[Q(s, \mu(s))\right] \tag{6}$$

The actor network is trained to choose actions that maximize the expected cumulative reward, given the current state. By updating the actor network using gradient ascent, it learns to select actions that lead to higher expected rewards.

The critic loss L_{critic} is used to minimize the temporal difference error between the estimated value and the observed reward:

$$L_{\text{critic}} = \mathbb{E}\left[\left(r + \gamma Q(s', \mu'(s')) - Q(s, a)\right)^2\right] \tag{7}$$

where r represents the reward obtained from the environment, s' is the next state, γ is the discount factor, $\mu'(s')$ is the action chosen by the target actor network given state s', and $Q(s, a)$ is the estimated action-value obtained from the critic network.

The Actor-Critic algorithm combines policy-based and value-based methods. The actor network is trained to update the policy parameters θ_{actor} using policy gradients, aiming to maximize the expected return. The critic network is trained to update the value function parameters θ_{critic} by minimizing the squared difference between the estimated value and the observed reward.

For the DQN algorithm, the Q-network approximates the action-value function $Q(s, a)$ using a neural network. The Q-network is trained to minimize the squared difference between the estimated value and the observed reward. The target network parameters are periodically updated to stabilize the learning process. The use of an experience replay buffer allows for efficient learning from past experiences.

These formulations provide a basis for addressing the eVTOL autonomous landing problem using DRL techniques. By integrating these algorithms and adapting them to the specific requirements and challenges of eVTOL landing, we aim to achieve accurate and safe autonomous landing capabilities in diverse environments.

IV. Proposed Approach

This section provides a comprehensive overview of the simulation framework we propose for eVTOL vehicle landing using reinforcement learning algorithms, built on a foundation of AirSim, UE, and PX4. Furthermore, the formulation of the problem for landing on a specific target, given the cases of single and multiple vehicles, within the reinforcement learning paradigm, will be detailed.

A. Simulation Framework

In the past, reinforcement learning algorithms were tested and validated using traditional simulation tools such as MATLAB or Simulink [47]. However, the complexity of real-world problems and the continuous nature of state and action spaces in robotics often made these tools inadequate.

In response to this, we have seen the development of more sophisticated and realistic environments for reinforcement learning simulation, such as OpenAI Gym [48], Gazebo simulator [19] combined with the Robot Operating System (ROS) middleware [49], and most recently, AirSim [18].

AirSim is an open-source simulator for drones and cars, developed by Microsoft. It has been designed to provide a realistic environment for the training and testing of autonomous vehicles, leveraging advanced graphics and physics simulations. It integrates with the PX4 flight controller, which enables high-fidelity control of quadcopters, and allows integration with various reinforcement learning frameworks.

In our study, we extend this environment by integrating AirSim with UE, creating a highly versatile, adaptable, and realistic simulation system for testing reinforcement learning algorithms for eVTOL vehicle landing. This framework provides a seamless interface between the reinforcement learning algorithm, the environment interface, and the AirSim simulator.

The communication between the agent and the environment in our framework is implemented through standard UE and PX4 communication tools. The framework also supports pausing the simulation time for computationally expensive training steps through a shared memory communication channel. The agent in the framework, which represents the reinforcement learning agent, receives experience vectors and rewards from the environment, assisting it in deciding the optimal action at each time step. Additionally, the agent logs relevant data to track its learning progress.

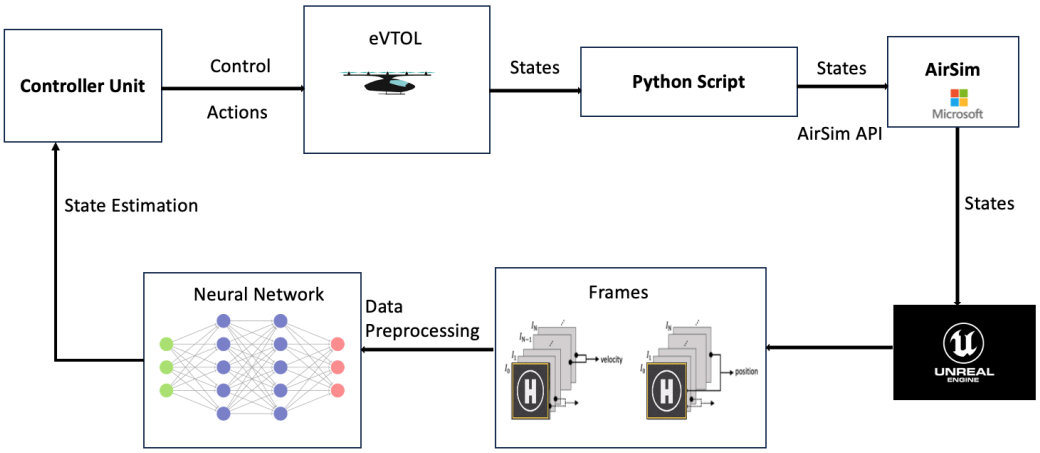

Fig. 1 Closed loop simulation environment overview showing the flow of information through the Simulation Engine and Neural Network.

Figure 1 provides a conceptual overview of our simulation framework, from which we can infer how the simulation environment using AirSim and UE could be set up for eVTOL vehicle landing problem. This framework demonstrates a closed-loop control system for precise eVTOL landings in a simulated environment. Neural networks estimate the eVTOL's state, a controller unit calculates actions, and a Python script interfaces with AirSim for execution, aligning the flight path with the desired trajectory that represents the ideal flight path for landing, which can be predefined based on mission requirements or dynamically generated to guide the eVTOL safely and precisely. The architecture of the proposed system for eVTOL vehicle landing involves various components, each having a crucial role in the functioning and performance of the system. These components, along with their primary functions, are described below:

- **Simulation Environment (AirSim):** AirSim is used as the primary simulation environment for the eVTOL vehicle. It provides highly realistic visuals and physics simulations, thereby ensuring that the learning algorithm is trained under conditions close to real-world scenarios. The environment is fully three-dimensional, crucial for eVTOL vehicles that have the ability to move freely in all directions.
- **Unreal Engine (UA):** Unreal Engine [50] serves as the core rendering and physics engine for the eVTOL vehicle simulation within AirSim. Its advanced capabilities provide a visually and physically realistic environment crucial for training and evaluating the reinforcement learning algorithm. Unreal Engine offers high-quality 3D graphics, lighting, and physics simulations, ensuring that the eVTOL vehicle interacts with its surroundings realistically. This integration with AirSim enables researchers to create immersive, true-to-life scenarios that help train the RL algorithm effectively. Unreal Engine plays a pivotal role in bridging the gap between the virtual simulation and

real-world application by closely replicating the real-world conditions in which the eVTOL vehicle will operate, thus enhancing the transferability of learned policies to the physical system.

- **Reinforcement Learning Algorithm:** The RL algorithm determines what actions the eVTOL vehicle should take based on the state information it receives. Depending on the specific algorithm used (DDPG, Actor-Critic, or DQN), it maintains an internal representation of the environment and uses it to make decisions. The goal of the RL algorithm is to learn a policy that maximizes the accumulated reward over time. In this case, the reward could be based on successful landings, maintaining stability, avoiding obstacles, and minimizing time and energy consumption.
- **Reward Function and State Representation:** These components are integral to the reinforcement learning setup. The state representation describes the current status of the eVTOL vehicle (e.g., its position, orientation, velocity, etc.) and possibly the environment (e.g., wind conditions, obstacles). The reward function provides feedback to the RL algorithm about the quality of its actions. In this case, the reward function could be designed to encourage safe and efficient landings.

Our custom eVTOL simulation environment builds upon Microsoft AirSim, leveraging Unreal Engine for realism. This environment enables comprehensive study and execution of complex missions in a controlled, safe virtual space.

B. Problem Formulation for eVTOL Vehicle Landing

Landing eVTOL vehicles precisely at a designated location, such as a vertiport, presents a significant challenge due to the complex nature of the operating environment and the inherent constraints of the vehicle dynamics. This challenge is amplified when multiple eVTOL vehicles attempt to land at the same vertiport. Therefore, it is essential to develop a systematic approach for safe and efficient landing of both single and multiple eVTOL vehicles.

In this section, we formulate the problem of landing eVTOL vehicles as a reinforcement learning problem. We employ Deep Reinforcement Learning (DRL) techniques to develop landing policies for both single and multiple eVTOL vehicles. We present the state and action space, along with the reward structure, for both single-vehicle and multi-vehicle scenarios. The objective is to find an optimal policy that can guide the eVTOL vehicles from an altitude of 100 meters to a designated vertiport while minimizing the landing error and the time taken for landing.

In the following subsections, we delve into the specifics of the problem formulation for both single-vehicle and multi-vehicle landing scenarios.

1. Single Vehicle Scenario

The single vehicle scenario involves an eVTOL vehicle that must execute a landing maneuver from an altitude of 100 meters down to a specific vertiport location. The state of the eVTOL is comprehensively represented as a 6-tuple $s = (x, y, z, \theta, \phi, \psi)$, where (x, y, z) capture the positional coordinates of the vehicle in a 3D Cartesian space, and the Euler angles (θ, ϕ, ψ) correspond to roll, pitch, and yaw of the vehicle, respectively.

In the single vehicle scenario, we denote the action taken by the eVTOL as $a = (d_x, d_y, d_z, d_\theta, d_\phi, d_\psi)$. This represents the desired direction of movement in the 3D Cartesian space (d_x, d_y, d_z) and the desired direction of change in the vehicle's orientation $(d_\theta, d_\phi, d_\psi)$.

The action space A is the set of all possible tuples, defined as:

$$A = \{(d_x, d_y, d_z, d_\theta, d_\phi, d_\psi) : -1 \leq d_i \leq 1, \quad \text{for all} \quad i\}$$

which is a 6-dimensional hypercube in the discrete case, where each d_i can take on values of -1, 0, or 1.

The vehicle commences the landing process from an initial state s_0, at an altitude of 100 meters above the vertiport. The problem then becomes one of devising an optimal policy π that can guide the eVTOL from this initial state s_0 to a desired final state s_g, representing the coordinates of the vertiport.

A cost function is thus established to enable quantifiable measurement of the efficiency and accuracy of the eVTOL landing. In this case, the cost function is given by the Euclidean distance from the current state of the vehicle to the target landing state (i.e., the vertiport).

Consequently, the objective of the reinforcement learning agent is to discover a policy π^* that effectively minimizes this cost:

$$\pi^* = \arg\min_{\pi} \left\{ \sum \left(\text{distance}(s_t, s_g) \right) \right\}$$

In this expression, $\text{distance}(s_t, s_g)$ signifies the Euclidean distance function.

*2. **Multi Vehicle Scenario***

In scenarios where multiple eVTOL vehicles (N in total) are involved, each vehicle i operates independently within the same 3D space, with its individual state represented as s_i. Therefore, the collective state of all N vehicles can be depicted as the vector $S = (s_1, s_2, ..., s_N)$.

In the multi vehicle scenario, each eVTOL i has its own action $a_i = (d_{x_i}, d_{y_i}, d_{z_i}, d_{\theta_i}, d_{\phi_i}, d_{\psi_i})$, with the desired direction of movement and orientation change given in the same way as the single vehicle scenario.

The collective action space of all N vehicles is then the tensor product of the individual action spaces:

$$A = A_1 \times A_2 \times \cdots \times A_N = \{(a_1, a_2, ..., a_N) : a_i \in A_i, \quad \text{for all} \quad i\}$$

This results in a highly multi-dimensional action space, which presents additional challenges for the learning algorithm.

Each eVTOL i is tasked with the execution of a landing maneuver onto a shared vertiport, represented by a common goal state s_g. The problem to be solved by the reinforcement learning agent is the determination of a policy π that can safely and effectively guide all vehicles to the vertiport. Importantly, when the vertiport is occupied, the incoming vehicles are expected to enter a hover mode within a specific zone until the vertiport is vacated and ready for the next landing.

In this multi-vehicle scenario, the cost function is composed of the sum of Euclidean distances from each vehicle's current state to the goal state (vertiport), augmented by an additional penalty cost associated with the time spent hovering.

Thus, the agent is tasked with finding a policy π^* that can simultaneously minimize both the total Euclidean distance from each eVTOL to the vertiport and the total hovering time:

$$\pi^* = \arg\min_{\pi} \left\{ \sum_i \left(\text{distance}(s_{i_t}, s_g) + \text{hover\_penalty}(t) \right) \right\}$$

where $\text{distance}(s_{i_t}, s_g)$ is the Euclidean distance function, while $\text{hover\_penalty}(t)$ is a function that imposes a cost that escalates with the time t spent hovering.

C. Reward Structure

In both scenarios, the reward function is designed to promote the efficient and safe landing of the eVTOL vehicle(s). A reward function that simply considers the absolute distance to the landing pad may lead to aggressive landing policies, which might not be safe. Therefore, we propose a reward structure that considers not only the distance to the vertiport, but also the velocity and angle of the eVTOL vehicle. The reward function is a real-valued function that depends on the current state and action.

*1. **Single Vehicle Scenario***

In the single vehicle scenario, the reward function $r : S \times A \to \mathbb{R}$ can be defined as:

$$r(s, a) = -\alpha \cdot \text{distance}(s_t, s_g) - \beta \cdot \text{velocity} - \gamma \cdot |\text{angle} - \text{optimal\_angle}|$$

where α, β, γ are positive weights, $\text{distance}(s_t, s_g)$ is the Euclidean distance from the current state to the goal state, velocity is the current velocity of the vehicle, angle is the current angle of the vehicle, and optimal_angle is the optimal angle for landing (which can be zero if the vehicle should land vertically). A large positive reward R is given when the vehicle successfully lands on the vertiport:

$$r(s, a) = R, \quad \text{if} \quad s_t = s_g$$

The reward and penalty structure for the single vehicle scenario is as follows:

- Distance Penalty (α): This penalty discourages the eVTOL from being too far from the vertiport. A value of $\alpha = 0.1$ could be reasonable. It means the agent receives a penalty for each unit of distance away from the vertiport.
- Velocity Penalty (β): This penalty discourages the eVTOL from moving too fast, promoting a safer landing. A value of $\beta = 0.01$ might work, meaning the agent receives a penalty for each unit of velocity.
- Angle Penalty (γ): This penalty encourages the eVTOL to maintain a suitable orientation for landing. A value of $\gamma = 0.05$ can be applied, and it represents the penalty for each degree of deviation from the optimal angle.

- Successful Landing Reward (R): A substantial positive reward, such as $R = 1000$, is given when the eVTOL successfully lands on the vertiport. This reward encourages the agent to aim for a successful landing.
- Safety Margin (ϵ): The safety margin is added to the distance penalty to define what is considered a safe landing. For example, $\epsilon = 5$ meters ensures that the agent is penalized only if it is more than 5 meters away from the vertiport.

*2. **Multi Vehicle Scenario***

In the multi vehicle scenario, the reward function $r : S \times A \to \mathbb{R}$ is defined similarly as follows:

$$r(s_i, a_i) = -\alpha \cdot \text{distance}(s_{i_t}, s_g) - \beta \cdot \text{velocity}_i - \gamma \cdot \left|\text{angle}_i - \text{optimal\_angle}\right|$$

where α, β, γ are positive weights, $\text{distance}(s_{i_t}, s_g)$ is the Euclidean distance from the current state to the goal state, velocity_i is the current velocity of the vehicle i, angle_i is the current angle of vehicle i, and optimal_angle is the optimal angle for landing. A large positive reward R is given when a vehicle successfully lands on the vertiport:

$$r(s_i, a_i) = R, \quad \text{if} \quad s_{i_t} = s_g$$

Additionally, we define a penalty function hover_penalty : $T \to \mathbb{R}$ to discourage vehicles from hovering:

$$\text{hover\_penalty}(t) = -\delta \cdot t,$$

where δ is a positive weight and t is the time spent hovering. The reward and penalty structure for the multi vehicle scenario is as follows:

- Distance Penalty (α): Maintain the same value as in the single vehicle scenario ($\alpha = 0.1$). It penalizes each unit of distance from the vertiport.
- Velocity Penalty (β): Keep the same value as in the single vehicle scenario ($\beta = 0.01$). It penalizes each unit of velocity.
- Angle Penalty (γ): Retain the same value as in the single vehicle scenario ($\gamma = 0.05$). It penalizes each degree of deviation from the optimal angle.
- Successful Landing Reward (R): Maintain the same value as in the single vehicle scenario ($R = 1000$). A substantial reward for successful landings.
- Hover Penalty (δ): Introduce a new penalty for hovering, discouraging vehicles from hovering for extended periods. A value of $\delta = 0.1$ could be used, meaning the agent receives a penalty for each unit of time spent hovering.
- Safety Margin (ϵ): Keep the same safety margin as in the single vehicle scenario ($\epsilon = 5$ meters). This ensures that the agent is penalized only if it is more than 5 meters away from the vertiport.
- Vertiport Availability Timer (τ): For each vertiport, introduce an availability timer, τ, which is set to 0 when a vehicle successfully lands on the vertiport. The vertiport becomes available for landing again after a certain time duration, in our case 3 minutes (180 seconds). When a vertiport is not available, attempts to land on it will result in a penalty.

We can now proceed to refine the reward function by integrating these essential elements:

$$r(s_i, a_i) = \begin{cases} \underbrace{-\alpha \cdot \text{distance}(s_{i_t}, s_g) - \beta \cdot \text{velocity}_i - \gamma \cdot \left|\text{angle}_i - \text{optimal\_angle}\right| + \text{hover\_penalty}(t)}_{\text{if the vertiport is available}} \\ \underbrace{-\alpha \cdot \text{distance}(s_{i_t}, s_g) - \beta \cdot \text{velocity}_i - \gamma \cdot \left|\text{angle}_i - \text{optimal\_angle}\right| + \text{hover\_penalty}(t) - \kappa}_{\text{if the vertiport is unavailable}} \end{cases} \tag{8}$$

The "penalty for unavailable vertiport" is a penalty, for example, $-\kappa$ (where κ is a positive constant), imposed when attempting to land on an unavailable vertiport. This reward structure encourages efficient landings while considering vertiport availability timers, promoting responsible landing behavior and queue management for eVTOLs. In our research, we have chosen to prioritize the efficient use of vertiports and minimize waiting times for eVTOLs by setting the penalty constant κ to a relatively low value of $\kappa = 1$. This choice encourages responsible landing behavior while allowing for flexibility in landing decisions

The agent's task is to find a policy that maximizes the sum of the rewards, which will lead to minimizing the distance and time to the landing pad, and encouraging safe landing.

D. Network Architecture

The network architecture as shown in Fig. 2 for the eVTOL vehicle landing problem in both single-vehicle and multi-vehicle scenarios is designed as follows:

- **Input Layer:** The input to the network is the state of the vehicle, represented by a 6-tuple $(x, y, z, \theta, \phi, \psi)$. For multi-vehicle scenarios, the input layer is extended to include the states of all vehicles.
- **Recurrent Layer:** The first layer of the network is a Long-Short Term Memory (LSTM) layer [51]. The LSTM layer, with 32 nodes, is designed to handle the temporal dependencies in the state sequence. LSTM layers are capable of processing sequential data and capturing long-term dependencies effectively. In the context of the eVTOL vehicle landing problem, the LSTM layer can convert unstable inputs, such as the varying positions and orientations of the vehicle(s), into a fixed-length vector representation that encodes the relevant information for decision-making.
- **Hidden Layers:** Following the LSTM layer, there are two fully connected (dense) layers [52]. The first hidden layer has 256 nodes, while the last hidden layer has 128 nodes. These hidden layers are responsible for extracting abstract features from the input data and mapping them to the appropriate output (Q-values or action values). The number of nodes in each hidden layer can be adjusted based on the complexity of the problem and the capacity of the network. In practice, it is recommended to experiment with different architectures and choose the number of nodes that achieves a good trade-off between model capacity and computational efficiency.
- **Output Layer:** The output of the network is the Q-value in the case of DQN or the action value in the case of DDPG and Actor-Critic algorithms. In DQN, the output layer is fully connected to the last hidden layer, with one output node for each possible action. In DDPG and Actor-Critic, the output layer is fully connected with a size equal to the action space, providing continuous action values. For multi-vehicle scenarios, the output layer is extended to include the actions of all vehicles.
- **Training and Optimization:** The network is trained using a reinforcement learning algorithm (DQN, DDPG, or Actor-Critic). The parameters of the network, including the weights and biases, are optimized using algorithms such as stochastic gradient descent (SGD) [53] or Adam optimizer [54]. The objective is to minimize the loss function defined by the RL algorithm and improve the overall performance of the network.

The network architecture diagram is as follows:

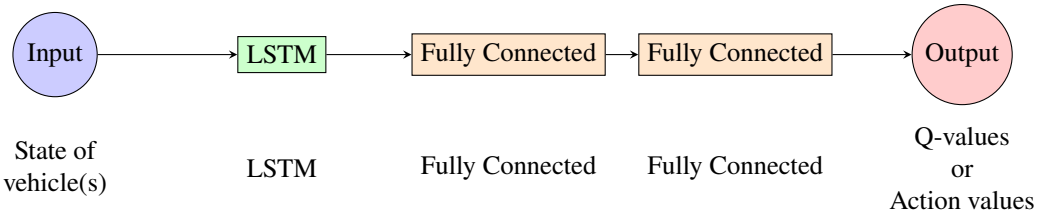

Fig. 2 Network Architecture for eVTOL Vehicle Landing.

It is important to note that the specific configuration of the network, including the number of nodes in the LSTM layer and the hidden layers, can be further optimized through experimentation and hyperparameter tuning. The chosen numbers, such as 32 nodes for the LSTM layer, 256 nodes for the first hidden layer, and 128 nodes for the last hidden layer, provide a starting point based on the given information and can be adjusted based on the specific requirements and constraints of the eVTOL vehicle landing problem.

V. Simulation Results and Discussion

In this section, we present the results and discussion of our simulations, which encompass two distinct case scenarios designed to address the challenges of safe and efficient eVTOL vehicle landings. In the first case scenario, our focus was on the precision landing of a single eVTOL vehicle on a designated vertiport. The objective was to develop a reinforcement learning policy that guides the eVTOL from an initial altitude of 100 meters to a specified landing point while minimizing landing errors and time. In the second case scenario, we extended our research to a more complex and dynamic environment, involving the simultaneous landing of multiple eVTOL vehicles at three distinct landing marks. The goal here was to devise a coordinated policy that enables multiple eVTOLs to land without collisions, optimizing landing sequences and hover times.

The visual showcases in Fig. 3 is a sophisticated urban air mobility (UAM) scenario using a state-of-the-art eVTOL

aircraft, set within a detailed urban environment rendered in Unreal Engine and operated by AirSim, a simulation platform developed by Microsoft. This scenario is designed to examine the intricacies of autonomous aerial navigation within the constraints of a densely constructed cityscape. It serves as a testbed for evaluating the eVTOL's flight control algorithms, including vertical takeoff, in-transit maneuvering, and precision landing capabilities on a designated rooftop helipad. Such simulations are vital in advancing the integration of autonomous vehicles in urban transportation networks, providing a sandbox to address the challenges of vehicle dynamics, obstacle avoidance, and AI-based navigation systems in a controlled yet realistic virtual setting. This enables rigorous testing of the UAM ecosystem's safety, reliability, and efficiency before real-world implementation.

Fig. 3 The Simulation Environment Developed using Unreal Engine [18].

In the single-vehicle landing scenario, we employed three distinct reinforcement learning algorithms: Actor-Critic, DDPG (Deep Deterministic Policy Gradient), and DQN (Deep Q-Network) to address the task of guiding a single eVTOL vehicle to a precise landing on a designated vertiport. The evaluation of these algorithms over multiple episodes provided valuable insights into their respective learning performances.

As illustrated in Fig. 4, the initial stages of training showcase a notable difference in the reward acquisition between the three algorithms. The Actor-Critic algorithm initiated with a comparatively lower reward but exhibited remarkable adaptability and rapid learning. Over the course of episodes, there was a significant upswing in the rewards received, eventually surpassing the performance of the other two algorithms. This compelling improvement highlights the superior performance of the Actor-Critic model in the context of the single-vehicle landing scenario.

In contrast, both DDPG and DQN algorithms exhibited a steady improvement in rewards as the training progressed. However, their learning curves demonstrated a tendency to plateau, failing to reach the same elevated level of rewards achieved by the Actor-Critic algorithm. This observation suggests that the Actor-Critic model, with its dual-network structure and the advantage of concurrent policy and value function optimization, excelled in striking a harmonious balance between exploration and exploitation throughout the learning process.

This adaptability and robust learning exhibited by the Actor-Critic algorithm in our single-vehicle landing scenario holds significant implications. It implies that the Actor-Critic approach is well-suited to handle the intricacies of eVTOL landing tasks, where precise maneuvering and efficient exploration of action spaces are critical. The model's ability to effectively adapt its policy based on real-time feedback leads to a more optimal policy, resulting in higher cumulative rewards.

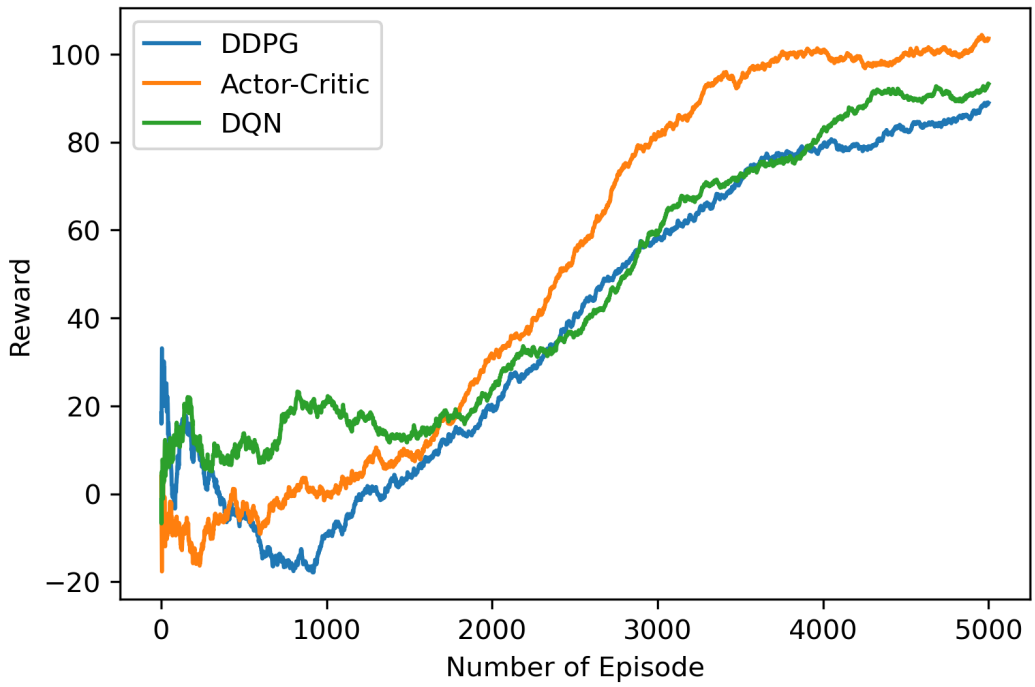

Fig. 4 Single-Vehicle Scenario: Rewards over Episodes for DDPG, Actor-Critic, and DQN

In the second case scenario, a custom simulation environment is constructed using Unreal Engine as shown in Fig. 5, featuring three vertiports designed for eVTOL landing operations. The scenario is further enriched by integrating the AirSim plugin, which introduces five eVTOLs into the simulation. This setup is intended to study and analyze the behavior and interaction of multiple autonomous eVTOLs as they navigate towards and execute landings on these dedicated vertiports. It provides a complex and dynamic framework to test algorithms for traffic management, landing scheduling, and collision avoidance in a multi-vehicle aerial environment.

In this complex scenario, multiple autonomous eVTOLs employ advanced algorithms to autonomously navigate to and land on three separate vertiports. These algorithms enable the vehicles to assess the operational landscape and communicate their positions and intentions. Decisions on landing prioritization are made through a dynamic process that considers various parameters such as distance, battery life, and vertiport occupancy. This collaborative decision-making is bolstered by conflict resolution protocols to ensure that any potential for overlapping landing claims is mitigated, allowing for a smooth and coordinated landing sequence for each eVTOL within the bustling urban airspace.

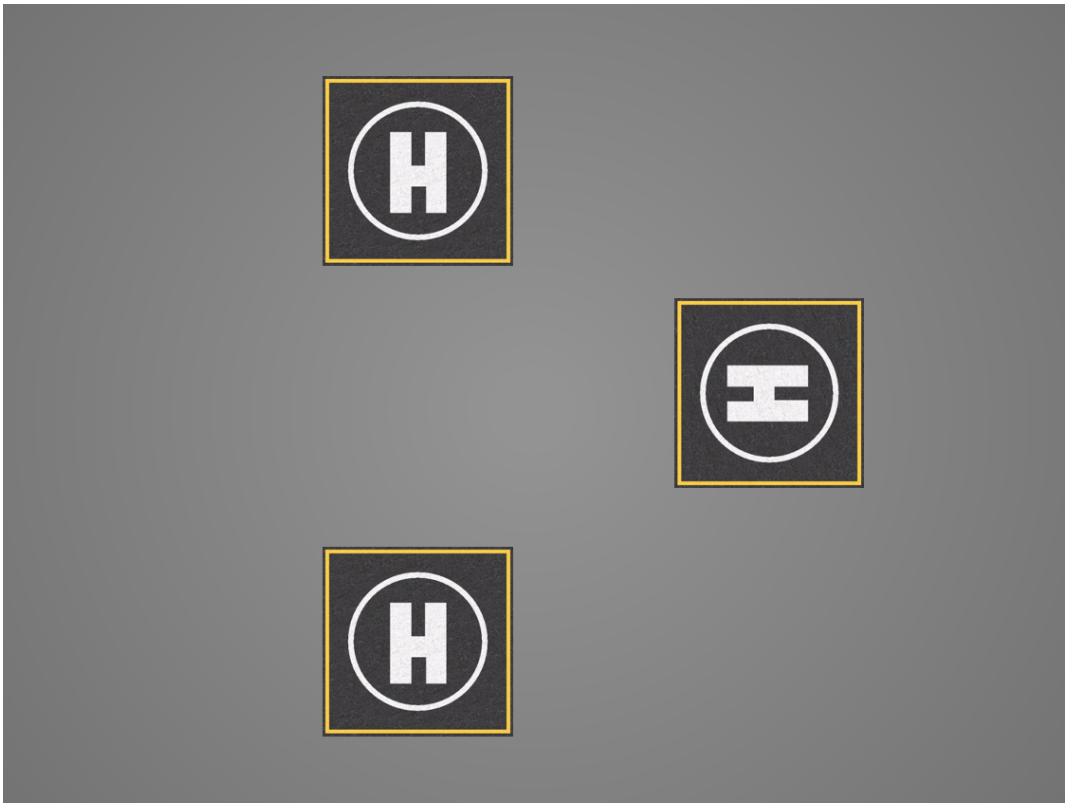

Fig. 5 A custom environment built on Unreal Engine with 3 vertiports.

In the multi-vehicle landing scenario, we observed similar trends to those in the single-vehicle scenario, as depicted in Fig. 6. Once again, the Actor-Critic algorithm demonstrated superior performance when compared to both the DDPG (Deep Deterministic Policy Gradient) and DQN (Deep Q-Network) algorithms. This consistency in the Actor-Critic model's outperformance suggests its efficacy in handling more intricate tasks, such as coordinating multiple eVTOL vehicles in a dynamic landing scenario.

It is noteworthy that DQN exhibited a lower variance in rewards, indicating higher stability in its learning process. However, this stability may come at the cost of slightly reduced effectiveness in this specific multi-vehicle landing task. In contrast, DDPG showed improved performance over DQN but was ultimately surpassed by the Actor-Critic algorithm, reaffirming the latter's dominance in this context.

The simulation results offer valuable insights into the capabilities of different DRL algorithms when applied to the challenge of multi-vehicle eVTOL landings. Notably, all three DRL algorithms—DDPG, Actor-Critic, and DQN—demonstrated the capacity to learn effective policies for eVTOL vehicle landing tasks, which is promising for their potential applications in urban air mobility and beyond.

However, the consistent superiority of the Actor-Critic algorithm across both the single-vehicle and multi-vehicle scenarios is a notable finding. This enhanced performance can be attributed to the algorithm's inherent ability to strike a balance between exploration and exploitation effectively. By adaptively adjusting its policy based on real-time feedback, the Actor-Critic model consistently achieved a more optimal policy, resulting in higher cumulative rewards.

In summary, while both DDPG and DQN exhibit promising capabilities in the context of eVTOL vehicle landings, the Actor-Critic algorithm emerges as the most effective and efficient choice for the task, particularly in scenarios involving multiple eVTOLs. However, it is crucial to recognize that these results are influenced by various factors, including algorithm parameters and architecture, task complexity, and environmental dynamics.

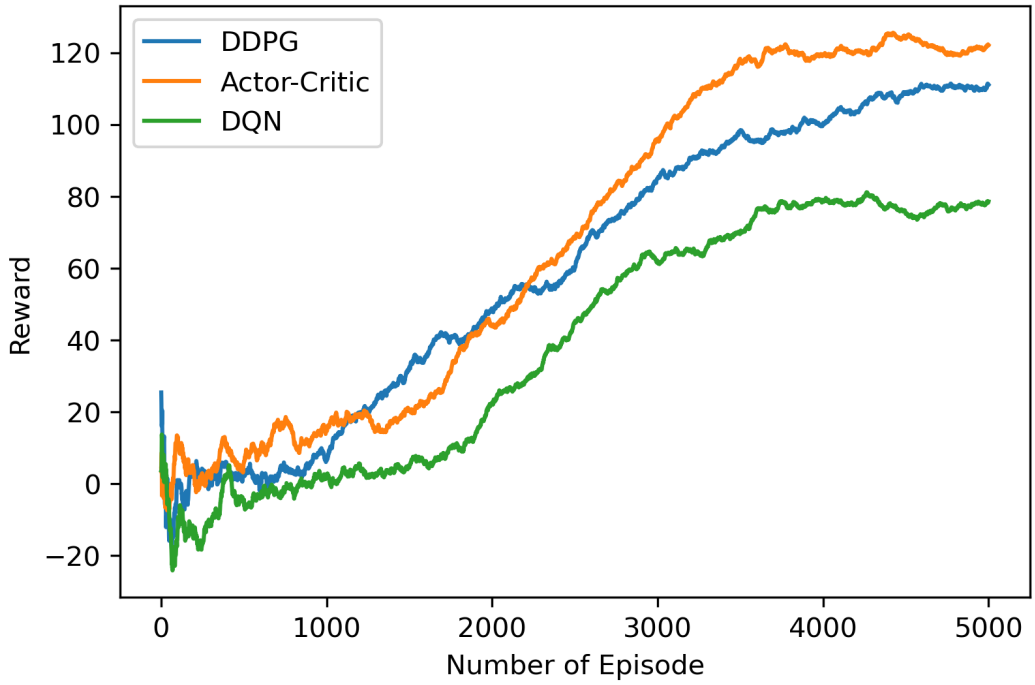

Fig. 6 Multi-Vehicle Scenario: Rewards over Episodes for DDPG, Actor-Critic, and DQN.

In the second case scenario presented in Fig. 7 highlights the intricate dynamics of a multi-agent system engaged in a reinforcement learning task, specifically employing the Advantage Actor-Critic (A2C) model. This model is instrumental in addressing the challenges of autonomous navigation and landing of five eVTOL aircraft across a limited number of vertiports, set at three. Analyzing the performance trends across 5000 episodes, we observe that the algorithm exhibits a steady and gradual improvement in performance, as evidenced by the consistent rise in successful landings over time. This trend reflects the algorithm's intrinsic capability to methodically assimilate and apply learned strategies, thereby improving its operational efficiency in a complex multi-agent environment.

As the A2C algorithm iterates through episodes, it continuously refines its policy, leveraging the cumulative experience to optimize decision-making processes. This is reflected in the gradual increase in successful landings, which plateaus as the algorithm approaches an optimal policy. The decrease in collision incidents is particularly significant, as it suggests an improvement in the eVTOLs' spatial awareness and a heightened ability to proactively manage risks. These developments are vital for ensuring safety in densely populated urban airspace environments.

In this multi-agent scenario, each of the five eVTOL aircraft operates autonomously within the dynamic environment, demonstrating the complexity of multi-agent reinforcement learning. Over the course of 5000 episodes, these eVTOLs collectively refine their decision-making processes. They learn to evaluate a multitude of factors, including their own current state, the states of other eVTOLs, vertiport availability, and potential collision risks. Through this continuous learning process, the eVTOLs develop individual landing policies that allow them to make informed decisions on when and where to land. These learned policies consider not only the immediate surroundings but also anticipate future actions of other eVTOLs, enabling them to strategically choose landing locations while avoiding collisions.

Figure 8 provides an insightful view into the cumulative average success rate of the A2C algorithm applied to the challenging eVTOL landing scenarios across 5000 iterations. The data reveals a notable pattern in the algorithm's learning journey. Initially, there is a rapid surge in performance, indicative of the algorithm swiftly acquiring fundamental navigation and landing strategies. During this initial phase, the success rate experiences a substantial boost while collision rates decline significantly, signifying the acquisition of critical skills for safe and efficient landings.

Following this initial surge, the success rate exhibits a more gradual, yet steady, upward trajectory. This phase suggests that the A2C algorithm continues to refine its decision-making processes. It strategically fine-tunes its policies and leverages the cumulative experience gained throughout the training iterations. The steady rise in success rate, especially notable beyond the 3000-iteration mark, hints at a breakthrough in strategy optimization or enhanced learning efficacy. It marks a pivotal moment in the algorithm's training, leading to a substantial enhancement in the cumulative

success rate.

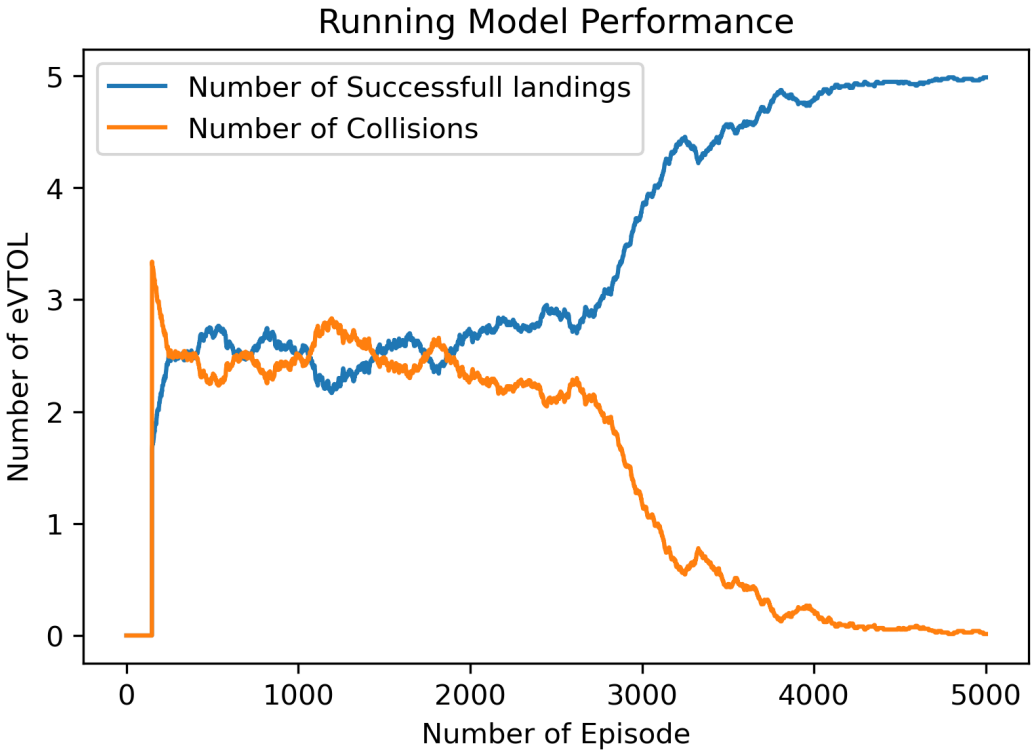

Fig. 7 Performance of A2C Reinforcement Learning in eVTOL Autonomous Landing Scenario. This graph tracks the progression of successful landings and collision incidents for five eVTOLs over 5000 training episodes in a simulated environment with three vertiports, demonstrating the A2C algorithm's adaptive learning and collision avoidance capabilities in a multi-agent system.

This performance trajectory showcases the algorithm's remarkable ability not only to rapidly acquire essential skills but also to progress into a phase of optimization and refinement. The incremental improvement observed over time underscores the algorithm's capacity to learn progressively from interactions within its environment. These findings emphasize the A2C algorithm's potential for addressing complex tasks, particularly those involving autonomous eVTOL coordination within the intricate urban airspace, where precision and safety are paramount considerations.

The Fig. 9 provides a visual presentation of the training progression of a multi-agent model aimed at safely landing five eVTOLs on three available vertiports over the course of 5000 episodes. The graph captures the cumulative results of this training, highlighting two critical performance metrics: the total number of successful landings (depicted by the blue line) and the total number of collisions (illustrated by the orange line). From the graph, it is apparent that as the number of episodes increases, the total number of successful landings (blue line) rises sharply, indicating that the model is learning and becoming more proficient at safely landing the eVTOLs. The collision count (orange line) also increases but at a slower rate, which could suggest that while the model is improving at landing eVTOLs without incident, there are still some episodes where collisions occur. The goal of the training would be to increase the number of successful landings while minimizing or ideally eliminating collisions. The trend suggests the model is becoming better at this task over time, but further refinement could be needed to reduce the collision rate more significantly.

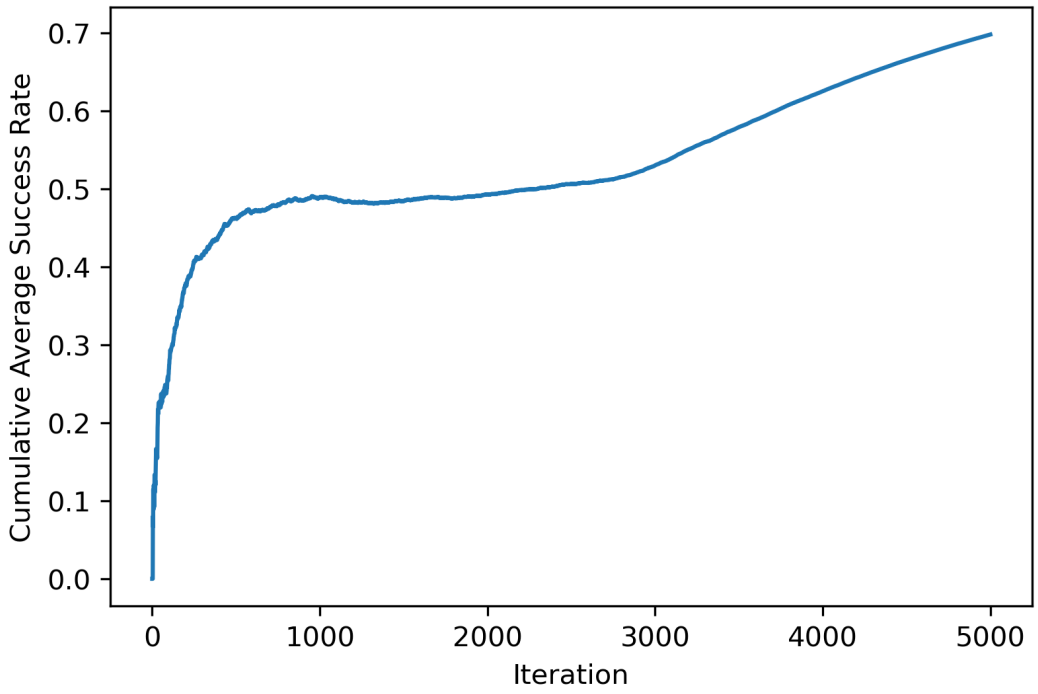

Fig. 8 Evolution of Cumulative Average Success Rate in eVTOL Autonomous Landings Over 5000 Iterations Using A2C Algorithm.

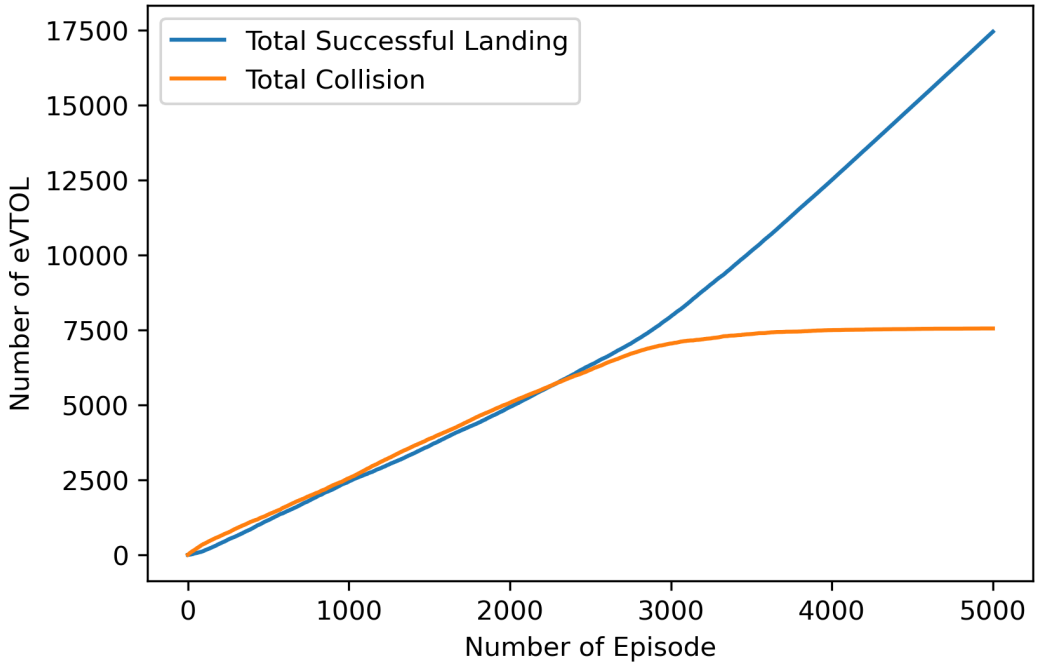

Fig. 9 Training Progression of Multi-Agent eVTOL Landing Model with Cumulative Successful Landings and Collisions Over 5000 Episodes

In sum, the findings from both single-vehicle and multi-vehicle landing scenarios underscore the efficacy of Deep Reinforcement Learning (DRL) algorithms in tackling the challenges of eVTOL landing in dynamic urban environments. The Actor-Critic algorithm consistently shines as the preferred choice, demonstrating its adaptability and superior performance in both scenarios.

Moreover, the multi-agent setting of the second case scenario highlights the A2C algorithm's ability to address complex tasks involving the coordination of multiple eVTOLs. Its gradual learning process, which leads to improved success rates and reduced collision incidents, emphasizes the algorithm's potential in scenarios demanding precision and safety, such as urban air mobility.

These results offer a promising outlook for the application of DRL algorithms in urban air mobility and similar domains, where autonomous and adaptive decision-making is critical. Future research avenues may include fine-tuning algorithms, exploring more intricate landing scenarios, and assessing adaptability to real-world conditions. This comprehensive evaluation will contribute to a deeper understanding of algorithm performance and guide their practical implementation in the emerging field of urban air mobility.

VI. Conclusion and Future Work

In this paper, we have addressed the problem of autonomous eVTOL vehicle landing using deep reinforcement learning (DRL) algorithms. Our work has focused on developing a DRL-based solution for both single-vehicle and multi-vehicle landing scenarios for the envisioned urban air mobility (UAM) missions. We have successfully formulated the problem and designed the network architecture for the landing task, considering the specific requirements of eVTOL vehicles and the availability of the vertiport. Our experiments and simulations have demonstrated the effectiveness of DRL algorithms in learning optimal landing policies and achieving efficient and safe landings. Our work provides a foundation for addressing the challenging problem of eVTOL vehicle landing using deep reinforcement learning. By continuing to explore and advance the field of DRL for autonomous landing, we can contribute to the development of efficient and safe landing strategies for eVTOL vehicles, facilitating the realization of future urban air transportation systems.

There are several promising directions that we will follow for future research in this field. First, further investigation will be conducted to improve the stability and robustness of the learned policies under various conditions and uncertainties, such as wind disturbances or sensor noise. Additionally, more advanced DRL algorithms, such as Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO), will be explored to enhance the learning performance and convergence. Furthermore, the scalability of the proposed approach to large-scale environments and a higher number of vehicles will be examined to address real-world UAM scenarios. In addition, the integration of the developed algorithms and policies with real-world eVTOL vehicles and testing in a physical environment is a crucial step towards practical implementation. This would involve deploying the learned policies on actual eVTOL platforms and evaluating their performance in real-time flight scenarios.

References

[1] FAA, "UTM Concept of Operations Version 2.0 (UTM ConOps v2.0)," https://www.faa.gov/researchdevelopment/trafficmanagement/utm-concept-operations-version-20-utm-conops-v20, 2020.

[2] Garrow, L. A., German, B. J., and Leonard, C. E., "Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research," *Transportation Research Part C: Emerging Technologies*, Vol. 132, 2021, p. 103377.

[3] Dietrich, A., and Wulff, Y., "Urban air mobility: Adding the third dimension to urban and regional transportation," *Presentation for: An Introduction to Urban Air Mobility for State and Local Decision Makers: A Virtual Workshop, sponsored by the Community Air Mobility Initiative (CAMI). Available online at https://www. communityairmobility. org/uam101*, 2020.

[4] Price, G., Helton, D., Jenkins, K., Kvicala, M., Parker, S., Wolfe, R., Miranda, F. A., Goodrich, K. H., Xue, M., Cate, K. T., et al., "Urban air mobility operational concept (OpsCon) passenger-carrying operations," 2020.

[5] Choi, J.-S., Lee, S.-H., Baek, J.-S., and Hwang, H.-W., "A study on vertiport installation standard of drone taxis (UAM)," *Journal of the Korean Society for Aviation and Aeronautics*, Vol. 29, No. 1, 2021, pp. 74–81.

[6] Yun, W. J., Jung, S., Kim, J., and Kim, J.-H., "Distributed deep reinforcement learning for autonomous aerial eVTOL mobility in drone taxi applications," *ICT Express*, Vol. 7, No. 1, 2021, pp. 1–4.

[7] Wang, Z., Wei, P., and Sun, L., "Optimal cruise, descent, and landing of eVTOL vehicles for urban air mobility using convex optimization," *AIAA Scitech 2021 Forum*, 2021, p. 0577.

[8] Wu, Y., Wang, Z., Benedikter, B., and Zavoli, A., "A Convex Approach to Multi-phase Trajectory Optimization of eVTOL Vehicles for Urban Air Mobility," *AIAA SCITECH 2022 Forum*, 2022, p. 2159.

[9] McDonald, L., Wu, Y., Deniz, S., and Wang, Z., "Real-Time Generation of Comfort-Optimal Flight Trajectories for Urban Air Mobility Missions," *AIAA SCITECH 2022 Forum*, 2022, p. 2157.

[10] Wu, Y., Deniz, S., Shi, Y., and Wang, Z., "A Convex Optimization Approach to Real-Time Merging Control of eVTOL Vehicles for Future Urban Air Mobility," *AIAA AVIATION 2022 Forum*, 2022, p. 3319.

[11] Wang, Z., "A Survey on Convex Optimization for Guidance and Control of Vehicular Systems," *arXiv preprint arXiv:2311.05115*, 2023.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[13] Deniz, S., and Wang, Z., "A Multi-Agent Reinforcement Learning Approach to Traffic Control at Future Urban Air Mobility Intersections," *AIAA SCITECH 2022 Forum*, 2022, p. 1509.

[14] Deniz, S., Wu, Y., Shi, Y., and Wang, Z., "A Multi-Agent Reinforcement Learning Approach to Traffic Control at Merging Point of Urban Air Mobility," *AIAA AVIATION 2022 Forum*, 2022, p. 3912.

[15] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P., "Trust region policy optimization," *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

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

[18] Shah, S., Dey, D., Lovett, C., and Kapoor, A., "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and Service Robotics: Results of the 11th International Conference*, Springer, 2018, pp. 621–635.

[19] Koenig, N., and Howard, A., "Design and use paradigms for gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3, IEEE, 2004, pp. 2149–2154.

[20] Cantelli, L., Mangiameli, M., Melita, C. D., and Muscato, G., "UAV/UGV cooperation for surveying operations in humanitarian demining," *2013 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, IEEE, 2013, pp. 1–6.

[21] Rodriguez-Ramos, A., Sampedro, C., Bavle, H., De La Puente, P., and Campoy, P., "A deep reinforcement learning strategy for UAV autonomous landing on a moving platform," *Journal of Intelligent & Robotic Systems*, Vol. 93, 2019, pp. 351–366.

[22] Araar, O., Aouf, N., and Vitanov, I., "Vision based autonomous landing of multirotor UAV on moving platform," *Journal of Intelligent & Robotic Systems*, Vol. 85, 2017, pp. 369–384.

[23] Rucco, A., Sujit, P., Aguiar, A. P., De Sousa, J. B., and Pereira, F. L., "Optimal rendezvous trajectory for unmanned aerial-ground vehicles," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 2, 2017, pp. 834–847.

[24] Gautam, A., Sujit, P., and Saripalli, S., "Application of guidance laws to quadrotor landing," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 372–379.

[25] Wang, Z., and McDonald, S. T., "Convex relaxation for optimal rendezvous of unmanned aerial and ground vehicles," *Aerospace Science and Technology*, Vol. 99, 2020, p. 105756.

[26] Lee, D., Ryan, T., and Kim, H. J., "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 971–976.

[27] Borowczyk, A., Nguyen, D.-T., Phu-Van Nguyen, A., Nguyen, D. Q., Saussié, D., and Le Ny, J., "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *Ifac-Papersonline*, Vol. 50, No. 1, 2017, pp. 10488–10494.

[28] Ling, K., Chow, D., Das, A., and Waslander, S. L., "Autonomous maritime landings for low-cost vtol aerial vehicles," *2014 Canadian Conference on Computer and Robot Vision*, IEEE, 2014, pp. 32–39.

[29] Hu, B., Lu, L., and Mishra, S., "Fast, safe and precise landing of a quadrotor on an oscillating platform," *2015 American Control Conference (ACC)*, IEEE, 2015, pp. 3836–3841.

[30] Vlantis, P., Marantos, P., Bechlioulis, C. P., and Kyriakopoulos, K. J., "Quadrotor landing on an inclined platform of a moving ground vehicle," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2202–2207.

[31] Kendoul, F., and Ahmed, B., "Bio-inspired taupilot for automated aerial 4d docking and landing of unmanned aircraft systems," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 480–487.

[32] Ananthakrishnan, U., Akshay, N., Manikutty, G., and Bhavani, R. R., "Control of quadrotors using neural networks for precise landing maneuvers," *Artificial Intelligence and Evolutionary Computations in Engineering Systems: Proceedings of ICAIECES 2016*, Springer, 2017, pp. 103–113.

[33] Shaker, M., Smith, M. N., Yue, S., and Duckett, T., "Vision-based landing of a simulated unmanned aerial vehicle with fast reinforcement learning," *2010 International Conference on Emerging Security Technologies*, IEEE, 2010, pp. 183–188.

[34] Kim, D. K., and Chen, T., "Deep neural network for real-time autonomous indoor navigation," *arXiv preprint arXiv:1511.04668*, 2015.

[35] Sadeghi, F., and Levine, S., "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[36] Zhang, T., Kahn, G., Levine, S., and Abbeel, P., "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 528–535.

[37] Deniz, S., Wu, Y., Shi, Y., and Wang, Z., "Autonomous Landing of eVTOL Vehicles via Deep Q-Networks," *AIAA AVIATION 2023 Forum*, 2023, p. 4499.

[38] Vascik, P. D., and Hansman, R. J., "Constraint identification in on-demand mobility for aviation through an exploratory case study of los angeles," *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3083.

[39] Daskilewicz, M., German, B., Warren, M., Garrow, L. A., Boddupalli, S.-S., and Douthat, T. H., "Progress in vertiport placement and estimating aircraft range requirements for eVTOL daily commuting," *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 2884.

[40] Guerreiro, N. M., Hagen, G. E., Maddalon, J. M., and Butler, R. W., "Capacity and throughput of urban air mobility vertiports with a first-come, first-served vertiport scheduling algorithm," *AIAA Aviation 2020 Forum*, 2020, p. 2903.

[41] Preis, L., and Hornung, M., "Vertiport operations modeling, agent-based simulation and parameter value specification," *Electronics*, Vol. 11, No. 7, 2022, p. 1071.

[42] Bacchini, A., and Cestino, E., "Electric VTOL configurations comparison," *Aerospace*, Vol. 6, No. 3, 2019, p. 26.

[43] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement learning: A survey," *Journal of artificial intelligence research*, Vol. 4, 1996, pp. 237–285.

[44] Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R., "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, No. 6, 2012, pp. 1291–1307.

[45] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[46] Van Hasselt, H., Guez, A., and Silver, D., "Deep reinforcement learning with double q-learning," *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30, 2016.

[47] Chaturvedi, D. K., *Modeling and simulation of systems using MATLAB and Simulink*, CRC press, 2017.

[48] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[49] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al., "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, Vol. 3, Kobe, Japan, 2009, p. 5.

[50] Sanders, A., *An introduction to Unreal engine 4*, CRC Press, 2016.

[51] Yu, Y., Si, X., Hu, C., and Zhang, J., "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation*, Vol. 31, No. 7, 2019, pp. 1235–1270.

[52] Albawi, S., Mohammed, T. A., and Al-Zawi, S., "Understanding of a convolutional neural network," *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[53] Bottou, L., "Stochastic gradient descent tricks," *Neural Networks: Tricks of the Trade: Second Edition*, 2012, pp. 421–436.

[54] Zhang, Z., "Improved adam optimizer for deep neural networks," *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, Ieee, 2018, pp. 1–2.