



**Politecnico  
di Torino**

# Noise-Aware UAS Flight Path Planning in Urban Environments with Reinforcement Learning

MASTER'S DEGREE THESIS IN  
MECHANICAL ENGINEERING

Author: **Shahin Sarhan**

Student ID:	297715
Advisor:	Prof. Stefano Primatesta
Co-advisor:	Prof. Giorgio Guglieri
Co-advisor:	Ing. Marco Rinaldi
Academic Year:	2023-24



## Abstract

This thesis presents a comprehensive approach to mitigating noise pollution from Unmanned Aerial Systems (UAS) in urban environments through flight path planning using reinforcement learning (RL). The study focuses on Turin, Italy, leveraging its diverse urban architecture to develop a comprehensive model. A detailed 3D occupancy grid map, based on OpenStreetMap data, was created to represent buildings' locations and heights while a population density map was developed to account for demographic variances. The research develops a dynamic noise source model that adjusts noise emission levels based on UAV velocity, ensuring realistic noise impact predictions. Acoustic ray tracing techniques are utilized to simulate noise propagation, accounting for atmospheric absorption and reflections from urban structures, providing a detailed analysis of noise distribution. The core of this work is the application of the Deep Deterministic Policy Gradient (DDPG) algorithm within the RL framework. The algorithm is tailored to optimize flight paths by minimizing noise impact while balancing other factors like travel distance and energy efficiency. The RL agent learns to navigate complex urban landscapes, integrating penalties for idling, excessive distance, and abrupt maneuvers to refine its path planning strategy.

Key findings reveal that the RL-based approach effectively reduces noise impact in urban settings, making it a viable solution for integrating UAVs into urban air mobility systems. The methodology is scalable and adaptable, with potential applications in various urban environments globally. This research contributes to the development of sustainable urban air mobility by addressing the critical issue of noise pollution, enhancing public acceptance and regulatory compliance.

**Key-words:** UAS, noise mitigation, reinforcement learning, acoustic ray tracing, DDPG.

# Contents

<b>Abstract.....</b>	<b>i</b>
<b>Contents .....</b>	<b>ii</b>
<b>Introduction.....</b>	<b>1</b>
<b>1. State of the Art.....</b>	<b>3</b>
1.1 Noise Mitigation Techniques .....	4
1.1.1 Trajectory Optimization .....	4
1.1.2 UAV Fuselage and Rotor Blade Design .....	8
1.1.3 Active Noise Cancellation.....	9
1.2 Path Planning Algorithms .....	9
1.2.1 Sampling Based Algorithms .....	10
1.2.2 Node-Based Optimal Algorithms .....	12
1.2.3 Mathematical Model-Based Algorithms.....	14
1.2.4 Bio-Inspired Algorithms.....	15
1.2.5 Machine Learning Based Algorithms.....	16
1.3 Noise Propagation .....	18
1.3.1 Ray Tracing .....	18
1.3.2 Gaussian Beam Tracing .....	19
<b>2. Methodology .....</b>	<b>21</b>
2.1 Study Area.....	21
2.2 Obstacles Map .....	21
2.3 Population Density Map.....	23
2.4 Noise Source Modelling .....	24
2.5 Noise Propagation Analysis.....	26
2.6 Ray Tracing.....	27
2.7 Reinforcement Learning .....	33

2.7.1	Approaches in Model-free RL .....	35
2.7.2	Interpolating Between Policy Optimization and Q-learning.....	36
2.7.3	Markov Decision Process .....	37
2.7.4	Policy and Value Functions .....	39
2.7.5	Experience Replay .....	40
2.8	Deep Deterministic Policy Gradient .....	40
<b>3.</b>	<b>Implementation and Results.....</b>	<b>45</b>
3.1	State Representation.....	46
3.2	Action Space .....	47
3.3	Reward Function .....	48
3.4	Training and Hyperparameter Tuning .....	52
3.4.1	Training Setup .....	52
3.4.2	Hyperparameter Configuration.....	53
3.5	Simulation and Results .....	54
3.6	Model Validation .....	65
3.7	Application .....	68
<b>4.</b>	<b>Conclusion and Future Development.....</b>	<b>75</b>
4.1	Conclusion .....	75
4.2	Scope for Future Work.....	76
	<b>Bibliography.....</b>	<b>79</b>
	<b>List of Figures.....</b>	<b>85</b>
	<b>List of Tables .....</b>	<b>87</b>



# Introduction

The advent of Unmanned Aerial Vehicles (UAVs), commonly known as drones, has ushered in a new era of possibilities in various sectors such as surveillance, search and rescue, delivery services and environmental monitoring. The rapid advancement of UAV technology has paved the way for Urban Air Mobility (UAM), a concept that envisions the use of airspace above urban areas for the transportation of goods and passengers. UAM aims to alleviate ground traffic congestion and provide efficient, on-demand transportation options. However, as the use of drones becomes more widespread, several challenges have emerged, particularly concerning their noise impact on urban environments.

UAM represents a significant leap forward in the transportation sector, promising to revolutionize how we move within cities. Drones, as a key component of UAM, offer various applications that enhance urban living. For instance, in surveillance, UAVs provide real-time monitoring capabilities that improve security and disaster response. In delivery services, drones can reduce delivery times and costs while minimizing the environmental impact compared to traditional vehicular delivery methods. Environmental monitoring through drones enables efficient data collection on air quality, vegetation health, and wildlife, contributing to better urban planning and environmental conservation.

The social acceptance of UAVs is intertwined with their perceived benefits and nuisances. While drones offer substantial advantages, their acceptance by the public is hindered by concerns over privacy, safety, and noise pollution. UAVs produce distinct noise signatures characterized by high-frequency tones that can be more annoying and perceptible than other sources of urban noise. This issue is exacerbated in densely populated areas where the ambient noise levels are relatively low, making drone noise more prominent and bothersome. Therefore, addressing the noise impact of drones is crucial for the successful implementation and social acceptance of UAM.

Addressing these concerns requires a comprehensive approach that combines technological innovation with policy and community engagement. Public education on the benefits of drones, along with transparent communication about noise mitigation efforts, can help build trust and acceptance. Furthermore, regulations that

enforce noise standards and operational guidelines are crucial for ensuring that UAV operations do not adversely affect urban residents. Current research on UAV noise primarily focuses on understanding its characteristics and impact. However, there is a notable gap in developing effective strategies for noise mitigation. The absence of standardized methods for evaluating UAV noise further complicates efforts to address this issue. To bridge this gap, our research introduces a novel approach to UAV path planning aimed at minimizing noise impact using Reinforcement Learning (RL).

In conclusion, our research contributes to the growing body of knowledge on UAV noise mitigation and offers a practical solution for integrating drones into urban environments. By minimizing noise impact through intelligent path planning, we can unlock the full potential of drones in various applications while ensuring they remain a welcome addition to urban landscapes.



# 1. State of the Art

The future utilization of Unmanned Aerial Systems (UAS) in urban air transportation holds significant potential for the sustainable development of cities [1], [2], [3]. UAS can revolutionize transportation by reducing traffic congestion and carbon emissions associated with traditional modes of transportation [4]. By efficiently navigating through the urban airspace, UAS can significantly decrease travel time and enhance the overall mobility of packages, such as food and express parcels. However, the integration of UAS into urban airspace also raises concerns regarding the potential negative impact of noise pollution. The noise generated by low-altitude UAS operations, especially in densely populated areas, can disrupt the tranquillity of urban environments and potentially disturb residents and wildlife [5].

To address these concerns, it is essential to implement comprehensive and environmentally conscious low-noise flight strategies for UAS. Such strategies can effectively mitigate noise disturbances, thereby preserving the tranquillity of urban environments [6]. This commitment to minimizing noise pollution aligns perfectly with the overarching objective of creating sustainable and liveable cities, where residents can enjoy a high quality of life while minimizing the ecological footprint of urban transportation systems [7], [8]. By embracing sustainable practices in UAS operations, we can pave the way for a future where cities thrive harmoniously with their surroundings, fostering a greener, quieter, and more sustainable urban landscape.

Urban areas are particularly sensitive to noise pollution, and UAVs contribute significantly to this problem due to their proximity to the ground and the density of human activity. Studies have shown that UAV noise can be more disturbing than ground vehicle noise because of its unique acoustic signature, characterized by higher frequency tones and irregular sound patterns. Research has indicated that prolonged exposure to UAV noise can lead to increased stress levels, sleep disturbances, and adverse effects on cardiovascular health. These findings underscore the importance of developing UAV path planning strategies that minimize noise impact, particularly in densely populated areas.

## 1.1 Noise Mitigation Techniques

Despite the significant advancements in UAV technology and its applications, there are surprisingly few studies focused on UAV path planning in order to minimize noise impact. The existing research on this topic is very recent, indicating an emerging field that has not yet been extensively explored. Additionally, there is currently no standardized method for evaluating UAV noise, which poses a challenge for researchers attempting to develop and assess noise mitigation strategies. The lack of a standard evaluation framework means that studies may use different metrics and approaches, making it difficult to compare results and build on previous work. This gap in the literature highlights the importance and novelty of this study, which aims to fill this void by applying Machine Learning techniques to optimize UAV flight paths with a specific focus on reducing noise pollution.

It's worthy to point out that the impact of noise is not solely determined by the distance it propagates. In fact, factors such as the flight state [9], the distribution of buildings, vegetation, and terrain characteristics [10], [11] influence the noise impact. This complexity becomes particularly apparent when considering low-flying UASs, as the impact of noise becomes three-dimensional and multifaceted. There is a clear opportunity for improvement in addressing the noise factor in a more comprehensive and accurate manner. Studies focusing on noise reduction are relatively scarce due to the challenges in efficiently evaluating the UAS noise impact on the environment. In response to this, previous efforts have been made to establish an efficient framework that allows for the estimation of drone noise impact on the 3D urban environment [11].

As it can be deduced, mitigating UAV noise is of great interest, given that most of the backlash it causes to all the scenarios they could be employed, limiting its applicability. The focus will now shift to presenting an overview of current approaches to noise mitigation. From what could be gathered from the literature, there seem to be three types of mitigation techniques which are discussed here.

### 1.1.1 Trajectory Optimization

Noise mitigation can be achieved by optimizing the flight paths of UAVs to reduce noise impact, given that sound pressure levels (SPL) diminish with distance. Several strategies have been proposed to mitigate UAV noise, focusing on both operational and technological aspects. Increasing the flight altitude of UAVs can significantly reduce ground-level noise due to the dispersion of sound waves and atmospheric absorption. However, in urban environments, altitude adjustments must be managed

carefully to avoid airspace restrictions and obstacles. Advanced path planning algorithms can minimize noise impact by avoiding densely populated areas and sensitive zones such as hospitals and schools. Heuristic and optimization-based algorithms can be adapted to include noise cost functions, prioritizing paths that reduce exposure to high population densities and minimize the noise footprint.

There are two primary approaches to mitigating aircraft noise [12]. For aircraft that can operate at altitudes where noise becomes negligible, noise mitigation efforts are focused on the arrival and departure phases. This involves considering factors such as airport location, terrain, elevation, and demographic distribution when designing flight paths to minimize noise during these critical phases. Conversely, for aircraft such as UAVs that cannot achieve sufficient altitude to mitigate noise, the entire flight path must be optimized. This approach requires prior knowledge of land use to ensure that noise-affected areas are minimized.

Most of the noise mitigation strategies found in literature focus on optimizing the entire trajectory. A notable example is the study in [13] which uses the iNoise software [14] to simulate and calculate noise (as per ISO 9613 standard). They provided three flight paths using a rough estimate of land use from a selected town. However, all three paths exceeded the World Health Organization's LDEN value, even with a small quadcopter. The study in [15] limits the flying space to a specific environment, overlays a grid, and designates obstacles and "quiet zones" (areas that should be noise-free). The A\* algorithm is then used to create various paths, each with a different level of 'quietness'. The study in [16] uses agent-based modelling to guide UAVs away from paths heavily used by other UAVs, which are assumed to be noisy. The study in [11] shares similarities with [15] but uses the ENVironmental Acoustic Ray-tracing Code (EnvARC) to simulate noise in a more sophisticated manner. They proposed some paths in a given flight space, but did not present any trajectory optimization.

It's important to note that a significant limitation of these noise mitigation techniques is the need for prior knowledge of the location of quiet zones. This can be challenging in scenarios where human activity varies throughout the day, such as in an industrial park or a university. As established in [16], UAVs increase noise activity when present in an acoustic environment but decrease it once they leave. Therefore, trajectory optimization requires up-to-date knowledge of an environment that can be highly dynamic and may not always be available.

The study in [17] integrates advanced noise assessment techniques and heuristic path planning algorithms to mitigate urban noise pollution from UAS. Utilizing a virtual flight simulator and a heuristic algorithm, it optimizes flight paths for low noise impact. Noise assessment involves sound source modelling for practical UAS models

and Gaussian beam tracing to simulate outdoor noise propagation. The path planning incorporates a noise-based objective function into the A\* algorithm, balancing obstacle avoidance, distance minimization, and noise metrics. Extensive urban simulations validate the noise-aware path planning algorithm, showing significant noise pollution reductions and underscoring its potential for quieter, eco-friendly UAS operations.

The study in [18] addresses the aviation industry's environmental impact through a RL approach. By utilizing high-fidelity sociological and topological data, the authors optimize spatial and temporal departure trajectories, modelling the problem as a continuous Markov decision process. The Soft Actor-Critic (SAC) architecture identifies optimal trajectories, revealing a notable trade-off between noise reduction and fuel consumption. This research presents a precise and sophisticated method for departure trajectory optimization, promoting environmental sustainability in aviation and potentially informing UAV path planning strategies to reduce noise impact.

Authors in [19] introduce a novel method to reduce noise pollution from low-altitude UAS flights in urban areas. By leveraging a noise assessment platform to generate noise maps of discretized airspace blocks, the method accounts for sound reflection, absorption, diffraction, and atmospheric attenuation. An improved cost-based A\* algorithm is used to find the optimal path with minimal sound exposure levels. Virtual flight simulations demonstrate the method's effectiveness in reducing en-route noise exposure. This cost-effective and efficient approach offers significant benefits for air traffic planning and management by mitigating noise pollution and could enhance UAV path planning strategies.

The study in [20] addresses public concerns about noise exposure in urban air mobility. It presents a comprehensive evaluation of flight noise under various conditions and proposes strategies for low-noise path planning. Two sound source evaluation methods are used: a semi-empirical model for rotor noise prediction and a boundary element method for fuselage noise scattering, along with high-fidelity computational aeroacoustics simulations for full-scale vehicles. The study employs a Gaussian beam method to calculate far-field sound propagation, considering sound refraction, atmospheric attenuation, and obstacle reflection. A heuristic method optimizes flight paths in urban scenarios, contributing to green aerial transportation by reducing noise impact.

Authors in [21] use a point-source spherical propagation noise model to simulate UAV noise, assuming isotropic noise distribution. This simplifies noise calculations while accurately representing real-world scenarios. For path planning, a simulated annealing algorithm is used, optimizing UAV flight paths to minimize urban noise

impact. The algorithm iteratively adjusts routes to balance noise reduction and energy efficiency, prioritizing noise abatement in residential zones by incorporating noise sensitivity levels. This approach effectively reduces noise pollution while maintaining operational efficiency for UAV delivery fleets.

The study in [22] introduces a novel approach that integrates noise restrictions and power management. Formulated as a Mixed Integer Linear Program (MILP), the path planning problem is discretized and solved akin to the shortest path problem. An exact dynamic programming-based labelling algorithm is employed, offering superior computational speed over traditional methods. Noise restrictions are incorporated by designating certain path edges where the gasoline generator cannot operate, thereby reducing noise in sensitive areas. This solution allows UAVs, equipped with both a battery and a gasoline generator, to operate quietly on electric power when necessary. The study effectively balances power management and noise mitigation for hybrid fuel UAVs.

Authors in [23] employ two primary methods to address noise reduction and path planning: Source Noise Model Integration and Observer Noise Model for Trajectory Optimization. The Source Noise Model adjusts flight parameters to directly modify the aircraft's noise emission, reducing the sound power level during flight. The Observer Noise Model estimates the noise experienced by an observer and optimizes the flight path to avoid noise-sensitive areas. The source noise model focuses on reducing the sound power level generated by the UAS, while the observer noise model optimizes the flight path to minimize noise impact on the ground. These methods create an acoustically-aware vehicle, balancing control effort and travel time while mitigating noise effects. The study highlights the integration of acoustics into the path planning process to develop optimized, noise-reducing trajectories.

The study in [24] presents an integrated cost assessment model that addresses third-party risks in metropolitan areas such as fatality risk, property damage risk, and noise impact. A novel flight path optimization method was developed to minimize the overall risk cost. The study uniquely considers the correlation between noise level and cost, setting a threshold at 40 dB to determine when noise impact becomes significant in path planning. The researchers used a hybrid algorithm, combining an estimation of distribution algorithm (EDA) with the CostA\* algorithm (EDA-CostA\*), to optimize UAV flight paths. This approach provides both global and local heuristic information for efficient path searching in cost-based environments.

The study in [25] offers a comprehensive approach to multi-objective path planning for UAVs, with an emphasis on noise reduction. The authors tackle the optimization problem of determining UAV paths in urban settings while considering multiple objectives. A height-dependent noise cost function is introduced to minimize noise

pollution for city dwellers. The study formulates a two-dimensional noise grid map, with each cell containing a noise value that represents the pollution cost for a drone passing through that cell. The path planning utilizes a 3D Non-Uniform Rational B-Splines (NURBS) based path representation and includes safety constraints to prevent UAVs from colliding with static obstacles and to maintain a minimal flight height. The study also evaluates various optimization algorithms and proposes a hybrid approach for solving the multi-objective path planning problem, taking into account both noise reduction and energy consumption.

Authors in [11] offer a comprehensive analysis of the noise impact of drones and urban air mobility vehicles on the environment. The authors used virtual flights and a Gaussian beam tracing solver, EnvARC, to model noise sources with arbitrary directivity patterns and consider various acoustic physics of propagation, reflection, absorption, and refraction. The study examined the effects of flight paths and boundary impedance on the environmental impact of quadcopter tonal noise on a simplified building model. It also assessed the noise impact in an urban setting with different flight paths and noise source models. The study underscored the importance of source type and flight path in influencing practical environmental noise distributions, offering valuable insights for path planning and noise management in drone operations.

### 1.1.2 UAV Fuselage and Rotor Blade Design

An effective approach to reducing drone noise pollution involves minimizing the noise at its source. Blade and vehicle fuselage optimization can be achieved through continuous experimental [26] and numerical investigations [27]. Additionally, practical UAS operational approaches from a managerial and logistical perspective are necessary to reduce the overall noise impact on the environment. This includes using different payloads for tasks with varying demands to minimize noise exposure [28], [29]. This technique involves modifying the UAV structure or propeller blades to reduce noise. Furthermore, modifications to propeller designs, inspired by low-noise fans, have been recommended to reduce the acoustic impact of routine UAS operations over populous areas [30].

There are two main strategies: the first one is modification of the whole UAV. Techniques like sound-proofing with rubber, foam pads, and aluminium foil have shown noise reduction, with foam pads reducing noise by about 15 dB [31]. Another design utilized the Coanda effect to reduce the number of propellers needed.

The second one is blade design. Optimizing propeller blades for noise reduction often involves balancing power constraints, thrust production, and noise. Studies have used genetic algorithms and biologically inspired designs like owl feather shapes to achieve noise reduction [32], [33]. Increasing blade numbers can reduce motor speed requirements, thus lowering noise. Other techniques include using porous materials [33] and loop-type blades [34] to further reduce noise. While these techniques can achieve noise reductions of 2-5 dB, they require precise design and testing to maximize effectiveness.

### 1.1.3 Active Noise Cancellation

Another noise mitigation method involves a digital signal processing technique known as active noise cancelling, applicable to cancelling drone noise. This technique requires capturing the noise to be cancelled, implying that the drone needs to carry on-board microphones to capture the noise created by its motors. Then, a phased version of the noise (the waveform of the noise multiplied by a factor of  $-1$ ) is reproduced near the location where noise mitigation is desired. The additive nature of the acoustic interaction of two signals results in destructive interference, cancelling the noise. This requires proper synchronization of the captured signal and the reproduced signal, as well as very low latency reproduction. Active Noise Control (ANC) systems, such as the FxLMS algorithm, have been effective in reducing acoustic noise, with online modelling of secondary paths enhancing adaptability to time-varying conditions [35].

Key methods for this technique include the use of on-board microphones, which capture noise in different directions and employ spherical sector harmonics to model and cancel out the noise [36]. Another technique is pitch shifting, where closed-loop controllers are used to monitor and reduce noise energy by up to 44% [37]. Additionally, a noise-free recording method combines on-board and secondary microphones to capture and cancel UAV noise, achieving near-complete cancellation of UAV noise in recorded audio [38]. These techniques are still emerging and may require additional audio hardware, which could be impractical in some scenarios.

## 1.2 Path Planning Algorithms

Path planning is a fundamental research problem in robotics, particularly from the perspective of control engineering. Its primary goal is to guide a drone from a

starting point to a specific objective, encompassing tasks from simple trajectory planning to selecting a suitable sequence of actions.

There are two main types of path planning: global and local. Global path planning computes the optimal path based on comprehensive environmental information known in advance. This type is more expensive to implement but provides a thorough plan. In contrast, local path planning generates paths in real-time using sensor data acquired during the drone's movement, making it more complex in design but adaptable to dynamic environments. Selecting the appropriate algorithm for a specific application can be challenging due to the multitude of available options. For UAVs, path planning is framed as an optimization problem, seeking the best possible solution among many. However, no single algorithm universally guarantees an optimal path for UAVs. Instead, various methods and algorithms are employed to analyse UAV behaviour and identify optimal solutions. The path planning algorithms are classified in five major categories described in this section.

### 1.2.1 Sampling Based Algorithms

Sampling-based algorithms in path planning do not require an explicit environmental representation, which offers a significant advantage due to the high computational cost associated with presenting complex environments. These algorithms generate a graph or roadmap to find a feasible route between a set of starting and ending points. A prominent example in UAS path planning is the Rapidly-exploring Random Tree (RRT) algorithm. Various sampling-based techniques are employed in UAV path planning, including RRT, RRT-star (RRT\*), and Probabilistic Roadmaps (PRM). Each of these methods offers unique advantages, with RRT and its variants particularly noted for their effectiveness in dynamic and unknown environments. These techniques collectively enhance the ability of UAVs to navigate complex spaces efficiently and safely. These algorithms are generally preferred for their efficiency and effectiveness in higher-dimensional spaces.

#### *Rapidly-exploring Random Tree*

The RRT algorithm is a widely used sampling-based technique in path planning for UAS. This algorithm is particularly effective in handling environments with obstacles and differential constraints. RRT is widely used in real-time applications where the environment is partially known or dynamic. It is particularly effective in high-dimensional spaces and can handle complex obstacle configurations.



The RRT algorithm's fundamental approach involves starting from a source point, randomly selecting positions, and expanding the tree by connecting feasible paths through free space. The process is iterated until a path to the goal is found or predefined iterations are completed. RRT's ability to rapidly explore new state space portions and its consistency in maintaining connectivity, even with minimal edges, make it a robust path planning tool. However, its standard implementation may converge to non-optimal values, necessitating improvements to increase convergence rates and efficiency. Recent advancements focus on heuristic functions to limit search areas and combining multiple trees from start and goal points, enhancing RRT's applicability and performance.

In UAS applications, the RRT planner has been effectively used for navigation and exploration in unknown and cluttered environments [39]. Variants such as the Predictive Rapid-exploring Random Tree have been implemented for collision avoidance in dynamic environments, showcasing the adaptability of RRT in different scenarios [40]. In search and rescue missions, RRT has demonstrated its ability to avoid static obstacles and no-fly zones, ensuring a collision-free path [41]. The basic RRT algorithm computes a single step of fixed distance, while its extension, RRT-Connect, offers a greedier approach for collision avoidance [42]. Zhang et al. [43] combined RRT-Connect with an artificial potential field to enhance UAV path planning, demonstrating its effectiveness in complex environments. Wen et al. [44] further explored RRT for UAVs, implementing it to predict dynamic threats and using the RRT\* algorithm for optimizing paths. Lin et al. [45] discussed variations of RRT for trajectory planning, intermediate pathfinding, and optimal solutions in obstacle-rich environments. Additionally, Yang et al. [46] introduced an environmental potential field-based RRT (EPF-RRT) to improve UAV path planning, highlighting the algorithm's adaptability and efficiency. The RRT algorithm has also been beneficial in other applications. Authors have noted its advantages in creating maps for unmanned ground vehicles (UGVs), with UAVs providing better results due to superior field view. For instance, Zu et al. [47] demonstrated that RRT could quickly find new paths when UAVs detect sudden obstacles, ensuring a safe and efficient route.

#### *Probabilistic Roadmap (PRM)*

The PRM algorithm [48], is a prominent sampling-based technique used for path planning in UAS. The PRM algorithm is tailored for multi-query applications and operates with probabilistic completeness, ensuring that the probability of failure diminishes exponentially with the number of samples used. The PRM algorithm operates through two main phases: construction and query. In the construction

phase, the connectivity of the free configuration space (c-space) is established by defining network curves in a three-dimensional environment. After constructing the roadmap, the query phase addresses the pathfinding between initial and final configuration points by concatenating these curves [49]. PRM is suitable for static environments where a map is available beforehand. It is used for offline planning, allowing the roadmap to be reused for multiple queries, thus saving computational resources. PRM is particularly useful for environments with complex free spaces where deterministic methods may fail or be computationally expensive.

This method has proven effective in generating collision-free paths for UAS in urban environments [50] and has been integrated with other path planning algorithms such as A\* [51] and D\* [52]. Path smoothing is an additional step often performed using algorithms like RRT, A\*, and RRT\* to refine the final path. For instance, Jang et al. [53] applied these algorithms to solve the traveling salesman problem (TSP) for UAVs, optimizing flight time and ensuring coverage of all remote sensing areas. PRM has been enhanced through various approaches, such as the kinodynamic PRM with a non-linear predictive control model proposed by Mansard et al. [54] which optimizes UAV path trajectories by initializing and monitoring state trajectories. Similarly, [55] utilized PRM for tracking and searching paths in urban environments, demonstrating its effectiveness in collision avoidance and communication gap detection.

### 1.2.2 Node-Based Optimal Algorithms

Node-based optimal algorithms utilize a decomposed node-grid of the environment, which includes detailed information about free and occupied spaces. These algorithms iteratively check if the current node, starting from the initial point, is the goal node. If it is not, the current node is marked as visited, and a neighbouring node is selected as the next current node. This process continues until the goal node is reached or all nodes have been evaluated. A prime example of such an algorithm is Dijkstra's algorithm. Research by Xinting Hu et al. [56] has highlighted the limitations of these algorithms, particularly their dependence on grid resolution, which exponentially increases the number of grid points and, consequently, the running time as the state space dimensionality grows

#### *Dijkstra's Algorithm*

Dijkstra's Algorithm, [57], [58] one of the earliest and foundational methods for solving the shortest path problem in graph theory, explores all possible paths

between a starting position and an endpoint, ordering them by cost. Initially designed to find paths between two specific points, a common variant sets a single node as the "source node" and finds the shortest path from this source to all other nodes, generating a shortest-path tree. The algorithm marks nodes as "visited" and updates the shortest known distances from the source, ensuring that the path connecting the source to all other nodes is the shortest possible.

While efficient for static, non-weighted graphs, Dijkstra's Algorithm has significant drawbacks in dynamic environments or when searching for the shortest path to a single specific target. It often explores large map portions aimlessly, increasing computational cost and time, making it less suitable for applications requiring quick reactivity, such as collision avoidance in UAVs. The algorithm's slow response time and high computational demand reduce autonomy and performance in dynamic environments. The algorithm operates on the principle of relaxation, gradually refining approximations of the shortest distance until reaching the optimal solution.

Dijkstra's algorithm has been effectively implemented in various applications, including the motion planning of fixed-wing UAS based on terrain data represented by digital elevation models [59]. Additionally, it has been used to solve path planning problems by considering the minimal gross propulsion energy required for a given route [60]. Advances in path-planning algorithms continue to build on the foundational principles established by Dijkstra, seeking to optimize performance and applicability in increasingly complex scenarios.

#### *A\* Algorithm*

The A\* algorithm [61] enhances Dijkstra's algorithm by incorporating a heuristic component, prioritizing nodes based on their estimated distance to the goal in addition to the path length. This combination of simplicity, effectiveness, and accuracy makes A\* a standard in robotics for solving pathfinding and graph traversal problems. A\* maintains a sorted queue of potential paths, prioritizing those minimizing total path cost. If an obstacle blocks the current path, the algorithm recalculates to find a new path that minimizes cost. The key formula for A\* is  $f(x) = g(x) + h(x)$  where  $f(x)$  is the total estimated cost,  $g(x)$  is the cost from the start node to the current node, and  $h(x)$  is the heuristic estimate of the distance from the current node to the goal, typically calculated as the straight-line distance to the goal.

Despite its strengths, A\*'s performance is influenced by map resolution: higher resolutions yield better paths but longer computation times, while lower resolutions are faster but risk less optimal paths and blocked narrow passages. A\* is designed for single-query applications, requiring continuous re-execution for updated paths,

which is computationally expensive in dynamic environments. Improved versions of A\* such as D\*, Field D\*, Fringe Saving A\* (FSA\*), and Generalized Adaptive A\* (GAA\*) have been developed to enhance efficiency and adaptability.

A\*'s effective integration of graph search logic and heuristics has been widely adopted in various applications [62], [63]. Chen et al. [64] discussed its advantages and drawbacks compared to genetic and ant colony optimization (ACO) algorithms, highlighting A\*'s effectiveness in fuel reduction and safe path planning. He et al. [65] extended A\* for energy-efficient, risk-aware path planning in dynamic environments. Benders et al. [66] applied A\* to account for wind and flight performance variations, demonstrating its robustness in challenging conditions. It has been used to optimize objectives such as minimizing travel distance or maximizing flight duration [67], [68]. In urban air transport management, A\* has been implemented in flight simulators for shortest path planning, considering altitude constraints and obstacle avoidance with buildings [69]. A study proposed a hybrid-cost objective function for urban UAS path planning, including a rudimentary consideration of noise effects [68].

### 1.2.3 Mathematical Model-Based Algorithms

Path planning for UAVs is crucial for efficient navigation in complex environments, requiring advanced algorithms and mathematical models. A prevalent approach involves formulating the problem as equations and inequalities that define environmental and system constraints. Mixed-Integer Linear Programming (MILP) methods, which combine discrete and linear variables, effectively model various path planning scenarios [70]. MILP has been used for UAS delivery service scheduling [71], optimizing path planning in challenging environments like the Arctic [72] and real-time trajectory planning in dynamic settings [73].

Researchers have successfully applied integer linear programming to solve problems like the TSP with multiple charging robots, minimizing path costs while accommodating fixed and reactive horizons [74]. MILP has also enhanced UAV trajectory planning, emphasizing scalability and consistency [75]. The use of MILP in mathematical modelling has significantly advanced UAV path planning. Studies have optimized task allocation for UAV fleets with timing constraints [76], formulated UAV mission scheduling for real-time applications in hostile environments [77], and incorporated terrain and communication constraints into MILP formulations [78]. However, the complexity of MILP solutions depends on solver efficiency, as the solution space expands exponentially with the number of binary variables [79], [80]. Researchers have explored heuristic approaches to

mitigate computational complexity [81] and investigated strategies to reduce the number of binary variables for faster solutions [82]. Additionally, efforts have been made to enhance MILP formulations for dynamic environments, integrating dynamic constraint builders for multi-UAV task allocation in hostile settings [83].

The paper [84], explores heuristic optimization techniques for logistics routing involving trucks and UAVs. It addresses the complex scheduling of delivery tasks, proposing solutions like local search algorithms and hybrid genetic algorithms. These methods optimize delivery efficiency by reducing total delivery time and improving coordination between trucks and drones. The research highlights the potential for UAVs to enhance logistical processes. The study in [85], introduces two greedy auction-based algorithms for assigning tasks to a heterogeneous fleet of UAVs. These algorithms prioritize safety and energy efficiency, addressing constraints like battery life and varying UAV capabilities. Moreover, the paper [86] proposes a hybrid auction-based architecture for task allocation among UAVs. This architecture aims to minimize energy consumption, ensure persistent service through strategic battery recharging, and evaluate safe aerial routes over populated areas.

#### 1.2.4 Bio-Inspired Algorithms

Evolutionary techniques such as memetic algorithms and Particle Swarm Optimization (PSO), inspired by bird flocking and fish schooling, have been extensively employed for UAV path planning [87]. These techniques involve selecting solutions from a pool of possible paths, considering constraints like environmental factors, robot capacities, and goal specifications. The fitness of solutions guides the selection of parent paths, which are then used to generate new paths through iterative processes, ultimately selecting the best-performing path as the optimal solution [88]. PSO is particularly effective in scenarios where UAVs must collectively explore the environment, adjusting their paths based on both individual experiences and social interactions with other UAVs. PSO's efficiency in finding optimal solutions with relatively low computational cost makes it suitable for real-time applications. The algorithm has been demonstrated in guiding UAV swarms during reconnaissance missions and optimizing path planning for data gathering from Wireless Sensor Networks [89], [90]. Improved variants of PSO enhance the rapidity and optimality of UAV formation path planning [91]. PSO algorithms have been prominent in trajectory and path planning for UAVs, aiming to minimize travel time while avoiding collisions [92], [93]. Incorporating bio-inspired algorithms into UAV path planning has yielded significant advancements in robotics and computer science. Ant Colony Optimization (ACO) algorithms, inspired by the foraging

behaviour of ants, have been instrumental in finding optimal routes for UAVs in complex environments [94], [95], [96].

Genetic Algorithms (GAs) are computational methods inspired by natural selection where potential solutions (represented as chromosomes) evolve over generations. This evolution occurs within a population of solutions, evaluated based on a fitness function that defines their quality relative to the optimization goal. Successful solutions are selected as parents to produce offspring through crossover and mutation operations, thereby exploring new potential solutions [97]. Despite their strengths, GAs are susceptible to premature convergence, where they settle on suboptimal solutions prematurely [98]. Researchers have addressed this issue through innovations like immune system-based GAs and vibrational algorithms enhanced with Voronoi Diagrams, aiming to improve convergence rates and solution quality [99], [100].

The application of GAs in UAV path planning has yielded promising results across various scenarios, from single to multiple UAS path planning. GAs have demonstrated efficacy in optimizing trajectories while considering dynamic environmental factors and energy constraints [101], [102], [103]. Incorporating prior knowledge into the GA process has further enhanced efficiency and reduced computational costs, enabling the derivation of optimal paths more effectively [104]. Novel variants like the multi-frequency vibrational genetic algorithm have emerged to streamline computational processes, reducing overall computational time for UAS path planning [105]. The versatility of GAs extends beyond path planning, finding applications in supervised learning and diverse optimization tasks [106], [107].

### 1.2.5 Machine Learning Based Algorithms

Machine learning (ML), a branch of artificial intelligence (AI), enables computers to respond intelligently without explicit programming. ML algorithms play distinct roles in UAV path planning. Unlike traditional planners, these systems heavily rely on image analysis from sensors, eliminating the need for map or graph construction. They use ML algorithms to learn control policies based on vision system outputs. Ground-truth labelling aids in obstacle recognition, allowing the drone to navigate towards paths with fewer obstacles or reduced crash risk.

RL and policy search algorithms are particularly beneficial as they can be trained within simulators using synthetic scenes. This enables drones to learn control policies and obstacle avoidance strategies in virtual environments, which can then be applied in real-world scenarios. Training algorithms on synthetic scenes offer significant



benefits, including obstacle pattern recognition and optimal trajectory generation. This approach improves the efficiency and safety of UAV operations, as drones can refine their navigation skills without the constraints and risks of real-world environments. In conclusion, ML-based approaches signify a paradigm shift in UAV path planning, utilizing advanced algorithms and computing power to enhance autonomy and efficiency.

### *Reinforcement Learning*

RL, characterized by its iterative learning process and continuous adaptation to the environment, is a key approach for addressing UAVs path planning challenges [108]. Various RL techniques, such as G-learning, Q-learning, deep reinforcement learning, and deep Q-networks, are extensively utilized to optimize UAV trajectories [109]. For example, G-learning computes a cost matrix based on geometric distances for UAV path planning, facilitating collision avoidance and reducing computational time [110].

Q-learning, a popular RL technique, aims to derive a strategy that maximizes returns by learning Q-values, mapping states to actions based on immediate and long-term returns [111]. It is used to solve the autonomous navigation problem of UAVs [112], [113]. However, its application in environments modelled as a grid world with limited UAV action space may reduce efficiency in real-world environments where UAVs operate according to a continuous action space [114]. Recent studies explore the application of deep reinforcement learning (DRL) methods for UAV navigation in urban settings. These methods address challenges posed by dynamic environments and rapidly moving obstacles. For instance, a distributed DRL framework decomposes tasks into subtasks to efficiently handle interactive data and discover optimal paths through obstructed terrain.

Deep RL has been deployed to autonomously control inverted helicopter [115], adaptively choose altitude for UAVs [116], and even resolve conflicts. Narrowing down the scope to trajectory optimization, one can refer to the work by Goddard et al. [117] in which Deep RL coupled with motion primitives was applied to descent trajectory of a six-degree-of-freedom F-16 model [118]. UAVs in obstacle-rich urban landscapes, systematically addressing flocking challenges through sequential subtask division. Additionally, multi-agent reinforcement learning is leveraged to model pursuit-evasion scenarios with unauthorized UAVs in urban airspace, effectively intercepting unauthorized UAVs.

## 1.3 Noise Propagation

This section delves into the intricate realm of acoustic noise evaluation. Two prominent methods that have significantly contributed to the field are explored: Ray Tracing (RT) and Gaussian Beam Tracing (GBT). These methods, each with their unique approach and algorithmic complexity, offer valuable insights into the propagation and behaviour of sound in various environments. This comparative analysis aims to provide a comprehensive understanding of these techniques, paving the way for further discussion on their applications and effectiveness in the realm of acoustic noise evaluation.

### 1.3.1 Ray Tracing

RT, initially developed for computer graphics, is now widely used in acoustics due to its efficiency in industrial applications. This method simulates sound wave trajectories by considering reflections and refractions, providing a detailed depiction of sound behaviour in complex environments. It accounts for boundary geometry, material, and medium characteristics [119], allowing for comprehensive sound distribution analysis, which is particularly useful for creating noise maps and designing spaces like concert halls and urban areas. The method incorporates obstacle modelling, facilitating the identification and mitigation of regions with high noise concentration and offering realistic simulations of sound propagation. However, the method may suffer from unrealistic caustics and shadow regions as stated by [120].

The computation process involves a dual-step procedure: the geometric step determines propagation paths using ray-tracing [121] or the image-source method [122], while the acoustic step computes sound attenuations based on factors like atmospheric absorption and the Doppler effect, adhering to standards such as ISO9613-2 [123], NMPB2008 [124], Harmonoise [125], and Nord2000 [126]. Ray tracing methodologies, like Krokstad's approach, generate rays from a source uniformly, considering directivity patterns and energy calculation at observation points by summing energies from all reaching rays. If a noise source, conceptualized as noise spheres, has specific physical attributes such as directivity patterns, the ray distribution can be weighted according to this directivity as shown in [127], [128].

Despite its strengths, acoustic ray tracing faces challenges such as systematic errors, sampling sensitivity, and the complexity of finding all possible sound paths. Methods to address these include visibility and obstruction tests or back tracing [129]. Additionally, it struggles with accuracy and efficiency, particularly in outdoor



environments, leading to the adoption of 2.5D ray tracing, which sacrifices realism. Various acceleration structures, such as the Uniform Grid, Bounding Volume Hierarchy (BVH), and Kd-tree, have been introduced to alleviate the intricacies of ray-scene intersections. Song et al. [130] state that to accurately model acoustic wave propagation below 5 Hz, the high-frequency approximation becomes invalid, requiring the inclusion of the influence of gravity and damping effects in the realistic atmosphere which is often disregarded. Schulthess [131] mentions that challenges also arise in modelling signal attenuation from geometric spreading loss and atmospheric attenuation loss.

### 1.3.2 Gaussian Beam Tracing

GBT extends RT by resolving the wave equation asymptotically near rays. In GBT, source energy is discretized into finite beams, and their evolution is governed by a Dynamic Ray Tracing (DRT) system. Acoustic energy at an observer is the sum of contributions from nearby Gaussian beams, eliminating nonphysical shadow regions and singularities of conventional RT. There are two main branches of GBT: one resembling ray tracing and the other related to the image-source method. GBT traces volumetric objects to determine specular reflection paths, enhancing accuracy in determining propagation distances by expanding rays into volumetric objects detectable by point-like receivers. The technique divides the space around the source into beams covering equal solid angles.

GBT variants are defined by geometric accuracies. Triangular beams provide exact coverage, conical beams are conservative but may detect multiple paths, and aggressive algorithms avoid multiple detections but might miss paths [132]. Early beam tracers overestimated detected energy due to multiple surface hits but Adaptive Beam Tracing and Adaptive Pyramid Tracing address this by splitting beams upon reflections while retaining initial beam tracing exactness. Adaptive pyramid tracing merges nearby narrow beams to reduce computation time and memory, albeit at the cost of exactness. Dalenback's approach uniquely considers both specular and diffuse reflections using successive cone tracing passes for diffuse energy propagation.

The second branch of GBT optimizes the image-source method by minimizing the growth in the number of image sources (ISs). It achieves optimization by culling ISs that cannot contribute to valid reflection paths. Unlike traditional GBT, where the beam count increases with each reflection, this approach forms beams by the reflecting geometry itself. First-order ISs generate beams from surface edges, creating

a minimal-sized beam tree and enhancing efficiency. Each beam is reflected only against surfaces within it, saving computation time and memory by neglecting surfaces outside the beam, resulting in a narrower IS tree and computational savings, especially at higher reflection orders [132].

GBT lacks diffraction modelling and assumes an omnidirectional point source, neglecting complex source directivity crucial for aircraft community noise predictions. Bian et al. [11] proposed a GBT-based model incorporating complex source directivity, but it struggles with projecting noise levels onto the ground in the presence of wind flow. Porter [133] noted that higher-order schemes do not significantly improve GBT results due to inaccuracies in environmental details and weather profile approximations. To enhance computational efficiency, adaptive step size strategies avoid computing redundant ray points [128]. When sources naturally produce a beam solution, a straightforward approximation uses a single beam. For sources lacking beamlike characteristics, the challenge is approximating the source as a superposition of beams. Porter [134] introduced a method aligning the high-frequency asymptotic field of a Gaussian beam representation with the exact solution in a homogeneous medium.

## 2. Methodology

### 2.1 Study Area

Turin, the capital city of the Piedmont region in northern Italy, is an ideal study area for evaluating UAV noise impact and path planning due to several compelling reasons. The city's dense urban core, surrounded by various residential, commercial, and industrial zones, presents a realistic scenario for testing UAV operations in different contexts. Additionally, Turin's diverse population density, with both densely populated neighbourhoods and sparsely inhabited areas, allows for a comprehensive analysis of UAV noise impact across different demographic settings. The architectural landscape of Turin, characterized by a mix of historical buildings, modern skyscrapers, and extensive residential complexes, provides a varied environment for testing the noise model, as noise propagation and reflection characteristics differ with building types and materials. Furthermore, Turin's involvement in smart city initiatives and sustainable urban development projects, coupled with its regulatory support for innovative transportation solutions, makes it a forward-thinking city likely to adopt advanced UAV technologies. These unique characteristics of Turin, combined with its rich data resources and proactive regulatory climate, make it a highly suitable and relevant study area for developing and validating a novel UAV path planning approach aimed at minimizing noise impact. The insights gained from this research in Turin can be generalized to other urban areas with similar characteristics, contributing to broader advancements in UAV noise mitigation and urban air mobility.

### 2.2 Obstacles Map

To construct the 3D occupancy grid for the study area, building location data was sourced from OpenStreetMap (OSM). The area of interest was defined by the longitude range from 7.6039 E to 7.71 E and the latitude range from 45.0386 N to

45.1094 N, encompassing the metropolitan area of Turin. Using the Overpass API, building data within these coordinates was downloaded. Once the data was acquired, the geographic coordinates (longitude and latitude) of the buildings were converted into Cartesian coordinates. This step was essential to preserve the relative positions of the buildings accurately on the map, ensuring that the spatial relationships among buildings were maintained. Given that OSM does not provide building height data, a random height between 12 and 18 m was assigned to each building, reflecting typical urban building heights in the region. The map was then trimmed to include only a specific portion with the remaining section discarded. This selection ensured that the resulting maps would be uniform in size, specifically  $200 \times 200 \times 20$  cells, which is crucial for consistency during the training phase.

The entire study area was then represented by a three-dimensional matrix, denoted as  $R$ . Each grid point  $R(i,j,k)$  in this matrix represents an occupancy value, indicating whether the corresponding geographical volume is occupied (value 1) or unoccupied (value 0). Each building's longitude and latitude were converted to  $(x,y)$  coordinates which were then mapped to the corresponding  $(i,j)$  indices in the matrix  $R$ . The height of each building was used to determine the  $k$  index, representing the vertical dimension. This ensured that the 3D representation accurately reflected the spatial structure of the urban environment. The resulting map is shown in figure 2.1 and the satellite photo of the study area is depicted in figure 2.2.

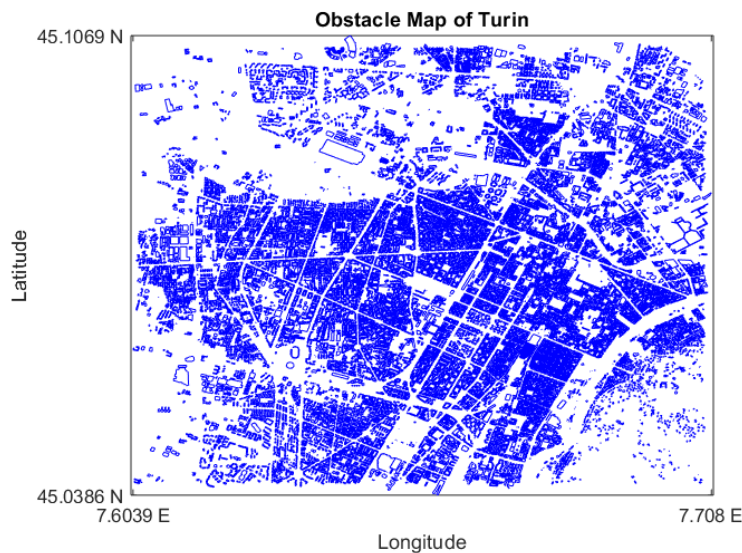


Figure 2.1 Obstacle map of the study area (Turin)

To facilitate efficient path planning and simulation, the large occupancy grid was subdivided into smaller chunks of  $200 \times 200 \times 20$  cells, resulting in 456 smaller maps. The uniform size of these chunks ensured consistency in the path planning process. This subdivision enabled the creation of training dataset for RL model which is necessary for training phase.



Figure 2.2 Satellite photo of the study area (Turin)

## 2.3 Population Density Map

To accurately represent population distribution within the study area, population density data for Italy was sourced from Meta. This data was filtered to include only the regions within the longitude range from 7.6039 E to 7.71 E and the latitude range from 45.0386 N to 45.1094 N, ensuring alignment with the area covered by the obstacles map. The dataset provides the number of individuals in each cell, where each square cell measures 1 arc second, corresponding to approximately 30.87 m at the equator. As latitude increases, the cell width decreases, necessitating adjustments in cell size representation.

The dataset includes the longitude and latitude coordinates of each cell's centre along with the corresponding population count. To process this data, the vertices of each cell were calculated from the centre coordinates based on their latitude, determining the exact boundaries of each cell. The population count assigned to each cell was

uniformly distributed across all points within the cell. These geographic coordinates were then converted to Cartesian coordinates to match the coordinate system used for the obstacles map. A 2D grid, mirroring the obstacles map, was constructed to represent the study area. Population density values were assigned to each point within this grid based on the processed population data. The resulting population density matrix was then normalized to ensure data consistency across the grid. The map is illustrated in figure 2.3. The normalized population density matrix was subdivided into smaller, more manageable maps, each measuring 200 x 200 cells. This subdivision ensured uniformity in map sizes and accurate overlap with the obstacles data which is necessary for training the model.

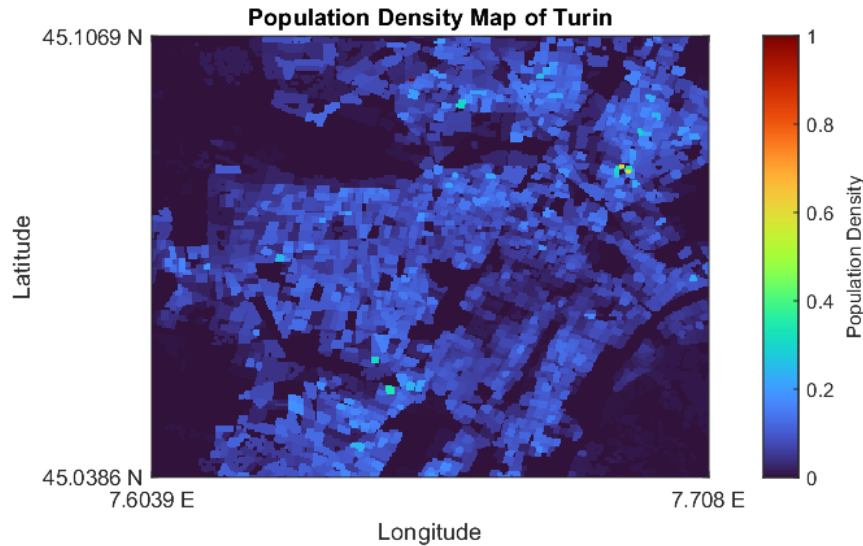


Figure 2.3 Population density map of the study area (Turin)

## 2.4 Noise Source Modelling

Numerous models of sound propagation exist, many of which incorporate complexities such as wind and other weather-related phenomena [135] as well as reflection or absorption from buildings, canopies, and geographic features [136]. For this study, a point-source spherical propagation model for the sound generated by the UAV was utilized. This model simplifies the calculations for computational efficiency within a simulation-based optimization framework while remaining reasonably representative of established findings [137]. Although multi-rotor UAVs produce highly directional noise in their immediate vicinity, it's assumed that the

noise propagates isotropically at the distances considered in this study. Consequently, as the UAVs traverse their routes, the noise pattern generated on the ground does not distort when turning.

Typically, a constant noise source is assumed in such models; however, the velocity of the drone is correlated with its noise level, and since the RL agent can adapt its velocity, it is beneficial to incorporate this variable into the noise model. To achieve this, data from the study in [138] was utilized that recorded noise levels at a fixed distance of 1 m from the 'DJI Inspire 2' quadrotor at speeds of 5, 10, and 20 km/h. Measurements were taken across various frequencies, ranging from 20 Hz to 20 kHz, using third-octave bands. For each velocity, the mean SPL across all frequencies was computed to obtain the overall SPL.

Once these three data points were established, an exponential curve was fit to the data, resulting in the SPL-velocity relationship as depicted in figure 2.4. The resulting equation is as follows:

$$L_0 = 60.24 \times \exp(0.003379 \times v) \quad (2.1)$$

- Where  $L_0$  is the source SPL in (dB) and  $v$  is the velocity in (km/h).

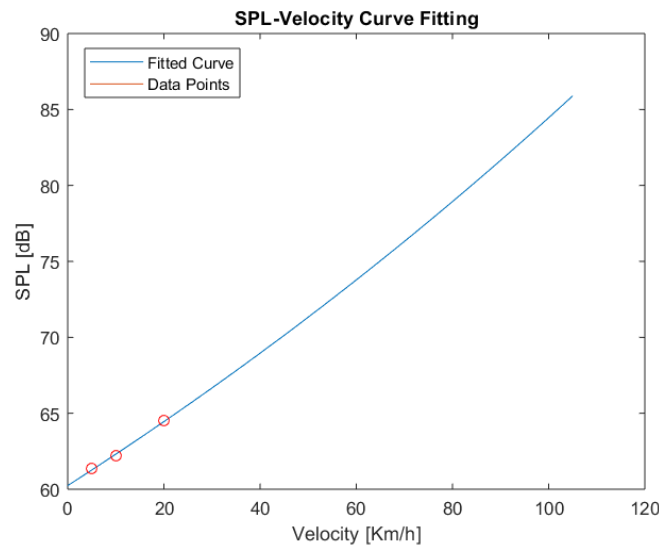


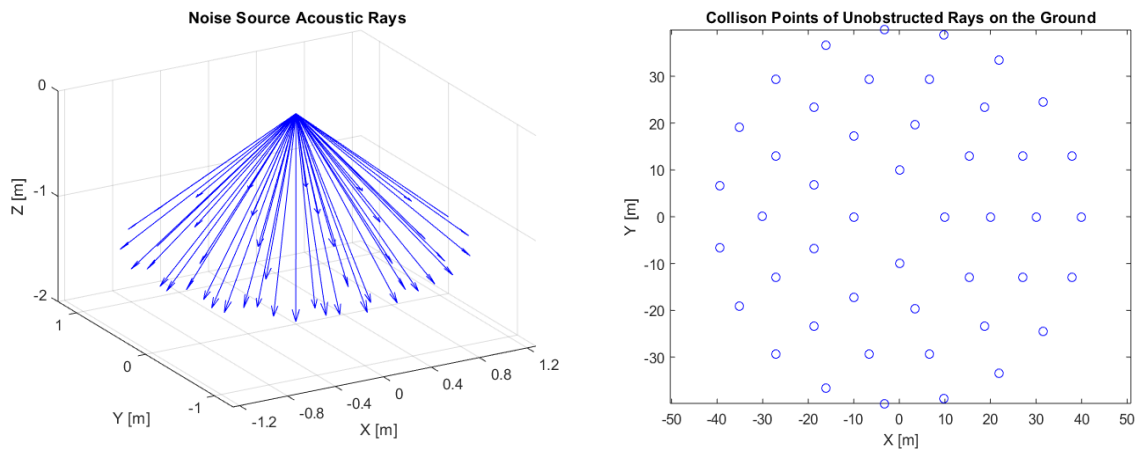
Figure 2.4 Exponential fit of velocity-SPL for the noise source



During flight, at any given instance, the emitted noise magnitude is determined based on the current velocity, allowing for a more accurate and dynamic representation of the UAV noise impact on the environment.

## 2.5 Noise Propagation Analysis

In the context of UAV path planning aimed at minimizing noise impact, accurately modelling noise propagation is critical. This process involves generating direction vectors from a noise source, which are utilized in ray tracing simulations. The following method was employed to generate these rays using. First, the altitude of the noise source was defined at 40 m. A radius of 10 m was set for generating points within concentric circles around the source at ground level. Four concentric circles were set around the source, each with a radius incrementally larger by 10 m, and each populated with an increasing number of points: 4, 9, 14 and 19 points respectively, distributed equally on the circle resulting in a total of 46 rays. Subsequently, direction vectors from the source point in the sky to each ground point were computed and normalized to unit vectors. This approach ensured a consistent direction magnitude for the vectors. By employing this method, the noise source is effectively modelled. The generated vectors were visualized in a 3D space to verify accuracy as illustrated in figure 2.5. The generated rays provide essential input for effective ray tracing, thereby facilitating precise analysis of noise propagation and aiding in the optimization of UAV flight paths to mitigate noise impact.



(a) Noise source rays visualized in 3D (b) Collision points of the unobstructed rays on the ground

Figure 2.5 The noise source acoustic rays model



## 2.6 Ray Tracing

Accurately modelling the propagation of UAV noise is essential for minimizing its impact on the environment. Acoustic ray tracing is a computational technique used to model the propagation of waves through a medium, interaction with obstacles, and attenuation over distance. This section details the ray tracing algorithm employed to evaluate the SPL at ground level due to emitted UAV noise. This method involves tracing the path of sound rays emitted from the noise source and calculating their interactions with the environment, considering reflections and atmospheric absorption. The following assumptions are made in the ray tracing model:

- **Medium Homogeneity:** The air through which the sound propagates is assumed to be homogeneous, with consistent properties such as temperature, atmospheric pressure and humidity.
- **Linear Propagation:** Sound waves are assumed to travel in straight lines unless they encounter obstacles or interfaces that cause reflection or absorption.
- **Specular Reflection:** Reflections off surfaces are assumed to follow the law of specular reflection, where the angle of incidence equals the angle of reflection. Diffuse reflections are neglected for simplicity.
- **Negligible Diffraction:** Diffraction effects, where sound waves bend around obstacles, are neglected for simplicity. Other more complex factors such as doppler effect are also not considered.

The ray tracing algorithm used to evaluate the SPL at ground level involves several key steps:

### i. Initialization:

The algorithm initializes several arrays to store data for each ray, including travelled distance, coordinates along the ray path, attenuation due to geometric divergence and atmospheric absorption and SPL values along the ray path. The noise source's current position and initial direction vectors for the rays are defined. The direction vectors are generated according to the methodology detailed in the preceding section.

The source noise magnitude emitted by the UAV is calculated based on its velocity. At the first iteration of a new episode, the minimum noise corresponding to the

UAV's minimum movement ( $[0,0]$ ) is considered. The velocity,  $v$ , is computed as follows:

$$v = \frac{||dx, dy||}{T} \quad (2.2)$$

- Where  $||dx, dy||$  represent the Euclidean norm of the movement vector and  $T$  is the sample time.

The division is done to obtain speed in (m/s), which is then converted to (km/h). The noise source level  $L_o$  is then determined using the exponential model in equation 2.1. This model reflects the increase in SPL with UAV velocity, derived from empirical noise-velocity data.

## ii. Ray Propagation:

For each ray, the initial position is set at the UAV's current coordinates  $(x, y, z)$ . The direction of each ray is initialized using pre-calculated direction vectors. The algorithm propagates the ray through the medium by iteratively updating the ray's position based on the direction and step length:

$$\begin{aligned} x_{t+1} &= x_t + x_{dir} \times \alpha \\ y_{t+1} &= y_t + y_{dir} \times \alpha \\ z_{t+1} &= z_t + z_{dir} \times \alpha \end{aligned} \quad (2.3)$$

- Where  $x_{dir}$ ,  $y_{dir}$ ,  $z_{dir}$  are the directional vectors for  $x$ ,  $y$  and  $z$  directions respectively and  $\alpha$  is the step length.

The updated coordinates are constrained within the map boundaries to ensure valid indices. The step length was selected to be 0.3 units. This value was determined through iterative ray tracing simulations and visualization of sound propagation to ensure model fidelity. A larger step length resulted in rays failing to detect obstacles, while a smaller step length significantly increased computational load. This step length struck a balance between accuracy and computational time.

### iii. Interaction Handling:

It's necessary to check for interactions with environmental elements (e.g., ground, buildings). If the ray reaches the map boundary (excluding the ground), ray tracing for that ray is terminated as it is leaving the area of interest and will no longer have any effect on it. If the ray hits an obstacle, the reflection is handled using the law of reflection. This ensures that the ray reflects accurately based on the incident angle as per the following equation:

$$R = I - 2 \cdot (I \cdot N) \cdot N \quad (2.4)$$

- Where  $R$  is the reflected vector,  $I$  is the incident vector and  $N$  is the normal vector of the hit surface.

### iv. Attenuation Calculation:

The SPL loss calculation as described here aligns with the methods prescribed by the ISO 9613-2 standard, which provide guidelines for predicting the attenuation of sound during propagation outdoors. It considers various factors such as geometric divergence, atmospheric absorption, ground effects and barriers. The SPL loss is calculated for each segment of the ray path due to atmospheric absorption and the geometric divergence.

The attenuation from geometric divergence,  $A_{div}$ , accounts for the reduced sound level as the sound propagates over a distance. For a point sound source in a free field, the attenuation in decibels (dB) is given by equation 2.5. The attenuation from atmospheric absorption  $A_{atm}$  during propagation over a distance  $d$  is given in (dB) by the equation 2.6:

$$A_{div} = 20 \log \left( \frac{d}{d_0} \right) + 11 \quad (2.5)$$

$$A_{atm} = \frac{\alpha d}{1000} \quad (2.6)$$

- where  $d$  is the distance travelled from source to receiver in (m),  $d_0$  is the reference distance (= 1 m); and  $\alpha$  is the coefficient of atmospheric attenuation in (dB/km) at the nominal median frequency of each third-octave band.

Given a temperature of 10 degrees Celsius, relative humidity (RH) of 70%, and assuming a nominal median frequency of 200 Hz (considering drones typically operate between 100-300 Hz with an average of 200 Hz), the atmospheric absorption coefficient ( $\alpha$ ) is set to be 0.76 dB/km as per ISO 9613-2.

In addition to these factors, the attenuation due to reflections off barriers such as walls is considered. The walls in the study are assumed to be solid with no windows, and as per ISO 9613, they are characterized by a coefficient of 1. This indicates that these walls act as perfect reflectors, meaning they do not allow sound to transmit through them. Consequently, there is no additional attenuation from these walls beyond the reflections they cause. Therefore, the sound propagation is influenced primarily by the direct and reflected paths around these solid walls.

#### v. SPL Calculation:

The SPL for each ray is recorded when the ray reaches within a height 2 m above the ground. This is done based on the equation:

$$L_i = L_0 - A_{div} - A_{atm} \quad (2.7)$$

- Where  $L_i$  is the SPL for each ray in (dB).

Once the SPL for each ray is known, the contributions from all N rays are summed to determine the overall SPL using the following equation. This is done considering only the non-zero value rays:

$$L_{tot} = 10 \log\left(\sum_i^N 10^{\left(\frac{L_i}{10}\right)}\right) \quad (2.8)$$

- Where  $L_{tot}$  is the noise impact at the ground level in (dB).

This ray tracing algorithm provides a comprehensive method for evaluating the SPL at ground level due to UAV noise. By accounting for reflections, atmospheric absorption, and geometric divergence, the algorithm accurately models noise propagation. This detailed analysis is crucial for optimizing UAV flight paths to minimize noise impact on the environment, ensuring compliance with noise regulations and enhancing community acceptance of UAV operations. The pseudocode for the method is detailed in Algorithm 1.

**Algorithm 1** Ray tracing pseudocode

---

```

1:   Input: UAV current coordinates  $(x, y, z)$ ; noise rays direction vectors  $x_{dir}, y_{dir}, z_{dir}$ ;
      action taken  $(dx, dy)$ 
2:   Initialize arrays to store distance travelled and attenuations
3:   If first iteration then
4:      $v = 0$ 
5:   else
6:      $v = \frac{||dx, dy||}{T}$ 
7:   end if
8:   Evaluate the source SPL:  $L_0 = 60.24 \times \exp(0.003379 \times v)$ 
9:   for  $ray = 1, N$  do
10:    while  $d < 200$  do
11:       $x_{t+1} = x_t + x_{dir} \times \alpha$ 
12:       $y_{t+1} = y_t + y_{dir} \times \alpha$ 
13:       $z_{t+1} = z_t + z_{dir} \times \alpha$ 
14:      If ray leaves map boundary, then
15:        Break
16:      end if
17:      If ray collides with obstacles, then
18:         $x_{ref} = x_{dir} - 2 \cdot (x_{dir} \cdot N) \cdot N$ 
19:         $y_{ref} = y_{dir} - 2 \cdot (y_{dir} \cdot N) \cdot N$ 
20:         $z_{ref} = z_{dir} - 2 \cdot (z_{dir} \cdot N) \cdot N$ 
21:        Update the direction vectors:  $x_{dir} \leftarrow x_{ref}, y_{dir} \leftarrow y_{ref}, z_{dir} \leftarrow z_{ref}$ 
22:      end if
23:      Evaluate distance travelled:  $distance = ||[x_t, y_t, z_t] - [x_{t+1}, y_{t+1}, z_{t+1}]||$ 
24:      Update the distance travelled:  $d \leftarrow d + distance$ 
25:      If  $z_{t+1} \leq 2$ , then
26:        Evaluate geometric divergence attenuation:  $A_{div} = 20 \log\left(\frac{d}{d_0}\right) + 11$ 
27:        Evaluate atmospheric absorption attenuation:  $A_{atm} = \frac{\alpha d}{1000}$ 
28:        Evaluate the SPL:  $L_i = L_0 - A_{div} - A_{atm}$ 
29:      end if
30:    end while
31:  end for
32:  Evaluate the overall SPL:  $L_{tot} = 10 \log(\sum_i^N 10^{\frac{L_i}{10}})$ 

```

---

## 2.7 Reinforcement Learning

This section provides an in-depth exploration of RL, focusing on fundamental principles, agent-environment interactions, and categorization of RL algorithms. Section 2.8, explores the Deep Deterministic Policy Gradient (DDPG) algorithm, detailing its implementation within the RL framework. It examines how DDPG optimizes policy and value functions, alongside noise models, to enable effective decision-making in environments with continuous action spaces.

RL aims to train an agent to accomplish tasks within uncertain environments. At each discrete time step, the agent receives observations and rewards from the environment and sends actions back. The reward provides immediate feedback on the success of the agent's previous action relative to the task goal. Unlike optimal control theory, RL can be model-free, in which the agent may not possess any prior knowledge about neither its mission nor the dynamics of the environment. This nature is beneficial particularly for problems whose dynamics cannot be easily described by physics-based differential equations. For RL problems with large admissible space, deep neural networks are employed to approximate values, forming a subsidiary paradigm called Deep RL.

A RL agent consists of two main components: a policy and a learning algorithm. The policy maps current environment observations to a probability distribution over possible action. This mapping is implemented by a function approximator with tuneable parameters and a specific model, such as a deep neural network. The learning algorithm updates the policy parameters continuously based on the agent's actions, observations, and received rewards. The objective is to find an optimal policy that maximizes the expected cumulative long-term reward.

Depending on the agent type, the learning algorithm may use one or more parameterized function approximators for policy learning. These approximators can function as:

- **Actors:** For a given observation, actors select the action that maximizes the policy value. i.e. the actor tunes its parameters to assign higher probabilities to actions yielding greater values.
- **Critics:** For a given observation and action, critics return an estimate of the policy value, representing the discounted expected cumulative long-term reward. i.e. the critic tunes its parameters to ensure predicted rewards match observed ones.

Different types of agents can be classified as:

- **Value-Based Agents:** These agents use only critics to select actions, relying on an indirect policy representation. They typically use approximators to represent value functions (value as a function of observation) or Q-value functions (value as a function of observation and action). Value-based agents are generally more efficient with discrete action spaces but may become computationally expensive for continuous action spaces.
- **Policy-Based Agents:** These agents use only actors to select actions, relying on a direct policy representation. The policy can be deterministic or stochastic. Policy-based agents are generally simpler and can handle continuous action spaces but may be sensitive to noisy measurements and local minima.
- **Actor-Critic Agents:** These agents use both an actor and a critic. During training, the actor learns the optimal action using feedback from the critic, who learns the value function from the rewards. Actor-critic agents can effectively manage both discrete and continuous action spaces.

These agents are either called On-Policy meaning they evaluate or improve the policy currently used for decision-making, or Off-Policy meaning they evaluate or improve a policy different from the one currently used, or from the one used to generate data.

When selecting an agent, it is advisable to start with simpler and faster-to-train algorithms that are compatible with the given action and observation spaces. More complex algorithms should be considered if simpler ones do not meet performance requirements. The observation and action spaces for each agent can be discrete, continuous, or mixed. For environments with small discrete action spaces, tabular approximators like Q-learning and SARSA are viable options, with Q-learning often offering faster training and SARSA providing robustness. However, as the state and action spaces grow, tabular methods become impractical, necessitating the use of network approximators. Agents supporting neural approximators for continuous action spaces include DDPG, TD3, SAC, and PPO. Among these, DDPG is the easiest to tune and a good starting point, while TD3 and SAC offer improved performance and robustness for computationally intensive environments. PPO excels in parallelization and training speed for computationally cheap environments.



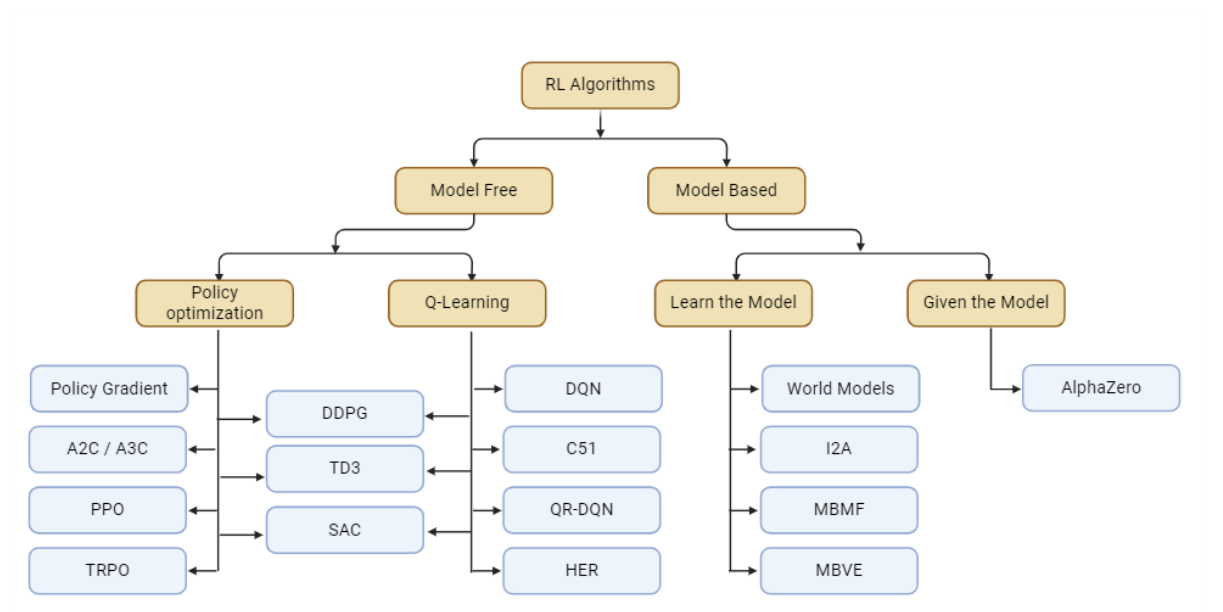


Figure 2.6 A non-exhaustive taxonomy of algorithms in modern RL.

Retrieved from <<https://spinningup.openai.com>>

A critical distinction in RL algorithms is whether the agent utilizes or learns a model of the environment. A classification of RL algorithms based on the models is illustrated in figure 2.6.

### 2.7.1 Approaches in Model-free RL

Model-free RL can be categorized into two main approaches: Policy Optimization and Q-Learning.

#### *Policy Optimization*

Policy optimization methods explicitly represent a policy  $\pi_{\theta}(a|s)$  and optimize its parameters  $\theta$  directly via gradient ascent on the performance objective  $J(\pi_{\theta})$  or indirectly by maximizing local approximations of  $J(\pi_{\theta})$ . Policy optimization is typically performed on-policy, meaning that updates to the policy parameters are made using data collected while following the current version of the policy. This ensures that the data used for learning is always relevant to the current policy. These methods often involve learning an approximator  $V_{\phi}(s)$  for the on-policy value function, which estimates the expected return of following the current policy from

state  $s$ . The value function helps in improving the policy updates by providing a baseline to reduce variance.

Examples of policy optimization methods include: A2C/A3C (Advantage Actor-Critic) where they perform gradient ascent to directly maximize performance, or Proximal Policy Optimization (PPO) where it indirectly maximizes performance by optimizing a surrogate objective function that conservatively estimates the change in  $J(\pi_\theta)$ .

### *Q-Learning*

Q-learning methods learn an approximator  $Q_\theta(s, a)$  for the optimal action-value function  $Q^*(s, a)$ . The optimization is conducted off-policy, meaning that it uses data collected at any stage of training, regardless of the current policy. This allows greater flexibility in data usage, as experiences collected from different policies can be reused. The objective function in Q-learning is typically based on the Bellman equation, which provides a recursive relationship for the action-value function. The goal is to minimize the difference (error) between the current Q-value predictions and the target values derived from the Bellman equation. The policy is derived from the Q-function, where actions are chosen to maximize  $Q_\theta(s, a)$ . Essentially, the agent selects the action with the highest predicted value in each state.

Example of Q-learning methods include Deep Q-Network (DQN), A foundational method that significantly advanced deep reinforcement learning. DQN combines Q-learning with deep neural networks to handle high-dimensional state spaces, making it effective for complex tasks like playing video games.

## 2.7.2 Interpolating Between Policy Optimization and Q-learning

The main strength of policy optimization methods is their principled approach, directly optimizing the desired objective, which generally results in stable and reliable performance. In contrast, Q-learning methods indirectly optimize performance by training  $Q_\theta(s, a)$  to satisfy a self-consistency equation. This indirect approach can lead to multiple failure modes, making Q-learning less stable. However, when effective, Q-learning is more sample-efficient due to its ability to reuse data more effectively than policy optimization techniques.

Serendipitously, policy optimization and Q-learning are not mutually exclusive and can sometimes be equivalent. There are algorithms that lie between these two extremes, balancing their respective strengths and weaknesses. By integrating

elements from both approaches, these algorithms achieve a flexible trade-off, leveraging the advantages of both policy optimization and Q-learning. Some examples include:

- **Deep Deterministic Policy Gradient (DDPG):** This algorithm concurrently learns a deterministic policy and a Q-function, using each to improve the other.
- **Soft Actor-Critic (SAC):** This method uses stochastic policies, entropy regularization, and other techniques to stabilize learning and outperform DDPG on standard benchmarks.

### 2.7.3 Markov Decision Process

RL is a method for addressing sequential decision-making problems under uncertainty, where an agent learns by interacting with its environment and receiving feedback. This process is typically modelled as a Markov Decision Process (MDP) as



Figure 2.7 The interaction between agent and environment

depicted in figure 2.7, which is a control sequence for decision making in circumstances where outcomes are partially controllable yet uncertain at some degrees.

It consists of a state space  $S$ , an action space  $A$ , an initial state distribution with density  $p_1(s_1)$ , and a stationary transition dynamics distribution with conditional density  $p(s_{t+1}|s_t, a_t)$  that satisfies the Markov property:

$$p(s_{t+1} | s_1, a_1, \dots, s_t, a_t) = p(s_{t+1} | s_t, a_t) \quad (2.9)$$

for any trajectory  $s_1, a_1, s_2, a_2, \dots, s_T, a_T$  in state-action space. This property implies that the future state  $s_{t+1}$  depends only on the current state  $s_t$  and action  $a_t$ , not on the sequence of events that preceded it. The agent receives rewards for performing actions based on the reward function  $R(s_t, a_t)$  which quantifies the success or failure of actions. The reward  $R$  can be formulated in various ways, either as a function of the action  $R(a)$  or as a function of action-state pairs  $R(a, s)$ .

The primary objective of the agent is to maximize the expected sum of discounted rewards, which incentivizes the agent to select specific actions. The cumulative reward is calculated by aggregating all rewards obtained during the execution of an episode. An episode, or trajectory, consists of a finite sequence of actions and concludes when the agent reaches a terminal state, such as a collision in a simulated navigation environment. The return from a state is defined as the sum of discounted future rewards:

$$G_t = E \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right] \quad (2.10)$$

- where  $\gamma$  is the discount factor ( $\gamma \in [0,1]$ ).

This factor emphasizes current rewards and reduces the impact of future rewards, facilitating convergence of the expected summation of discounted rewards. Therefore, a larger discount rate encourages more exploration by looking further into the future, and vice versa. The value of  $k$  increases by one at each time step to emphasize the current reward and to reduce the impact of the future rewards, hence the term discounted reward.

### 2.7.4 Policy and Value Functions

Considering its current state  $s$ , the agent generates an action  $a$  via its policy deterministically,  $a = \pi(s)$ , or stochastically,  $a \sim \pi(a|s)$ . There exists at least one optimal deterministic policy  $\pi^*(a|s)$  for any MDP.

The agent gradually updates its policy  $\pi$  towards  $\pi^*$  via action-value methods or policy gradient methods, which are the two principal approach categories. In the first group, the agent learns values of feasible actions and/or states, based on which action decisions are elected. The latter classification involves methods which parameterize the agent policy and improve its performance via gradient ascent. Contrary to its value-based counterparts, policy gradient is proved to be more stable yet sample-inefficient. Fortunately, this detriment of policy-based approaches can be improved by combining with value-based methods.

The state-value function  $V_\pi(s)$  describes the expected return from state  $s$  following policy  $\pi$ :

$$V_\pi(s) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right] \quad (2.11)$$

The action-value function  $Q_\pi(s, a)$  describes the expected return after taking an action  $a_t$  in state  $s_t$  and thereafter following policy  $\pi$ :

$$Q_\pi(s, a) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \quad (2.12)$$

The agent selects actions that maximize the Q-value. However, the optimal Q-value,  $Q^*(s, a)$  which maximizes the expected summation of discounted rewards under the optimal policy  $\pi$ , satisfies the Bellman optimality equation which is equal to the expected reward  $R_{t+1}$ , plus the maximum expected discounted return that can be achieved for any possible next state-action pairs  $(s', a')$ .

$$Q^*(s, a) = \max_{\pi} Q(s, a) \quad (2.13)$$

$$Q^*(s, a) = E[R_{t+1} + \gamma \max_{a'} Q^*(s', a')]$$

This optimal Q-value  $Q^*(s, a)$  is utilized for training neural networks. The predicted Q-value  $Q(s, a)$  is compared to the optimal Q-value estimated by Bellman's equation, and the discrepancy is backpropagated through the network. The loss function is defined as:

$$Loss = E [R_{t+1} + \gamma \max_{a'} Q^*(s', a')] - E \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right] \quad (2.14)$$

### 2.7.5 Experience Replay

Experience Replay is a technique wherein experiences  $e$  ( $e$  can be described as the knowledge produced from the agent performing an action  $a$  in a state  $s$  causing a new state  $s'$  and a generated reward  $r$ ) can be defined as tuples  $(s, a, r, s')$  stored in a replay memory  $R$  and reused to train the agent. This approach enhances convergence by allowing the agent to learn from significant past experiences. However, sampling experiences uniformly from  $R$  may not reflect their significance. Prioritized Experience Replay was later introduced, which prioritizes experiences based on Temporal Difference (TD) error, replaying experiences with lower TD-error more frequently.

## 2.8 Deep Deterministic Policy Gradient

The DDPG algorithm [139], is a model-free, off-policy RL method. It is an actor-critic RL agent designed to concurrently learn a Q-function and a policy, leveraging off-policy data and the Bellman equation to optimize both. In RL, policy gradient methods are used to find the optimal policy by optimizing a parametric policy  $\pi_{\theta}$ . Traditional policy gradient methods use stochastic policies, where the action is sampled from a probability distribution. Deterministic Policy Gradient (DPG)

methods, however, use a deterministic policy  $\pi_{\theta}(s)$  that directly maps states to actions. DDPG agents are suitable for environments with continuous or discrete observation spaces and continuous action spaces.

DDPG agents utilize two main components; an actor and a critic as depicted in figure 2.8:

- **Actor:** The actor represents a deterministic policy  $\pi(s)$ . A deterministic actor takes an environment observation as input and returns as output an action that is a parametrized deterministic function of the observation, thereby implementing a parametrized deterministic policy.
- **Critic:** The critic evaluates the action-value function  $Q(S,A)$  which is a mapping from an environment observation-action pair to the value of a policy.

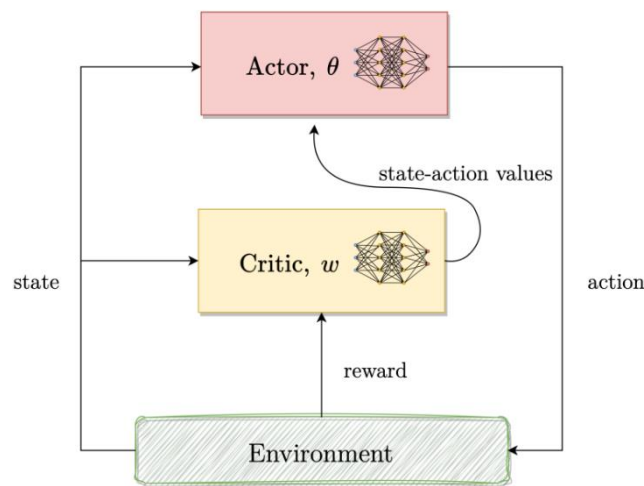


Figure 2.8 Generic architecture for actor-critic agent (Nguyen, 2023)

During training, DDPG agents update both the actor and critic properties at each time step. They store past experiences in a circular experience buffer and update the actor and critic using mini-batches of experiences randomly sampled from this buffer. To ensure stable behaviour, the replay buffer must be sufficiently large to encompass a diverse range of experiences. However, it is not always beneficial to retain all past experiences. Using only the most recent data can lead to overfitting, while using an excessive amount of data can slow down the learning process. Therefore, careful tuning is required to strike the right balance.

DDPG trains a deterministic policy using an off-policy approach. Since the policy is deterministic, on-policy exploration at the start may not cover a sufficiently diverse range of actions to obtain useful learning signals. To enhance exploration, noise is added to actions during training. While the original DDPG paper recommended using time-correlated Ornstein-Uhlenbeck (OU) noise, more recent findings indicate that uncorrelated, mean-zero Gaussian noise is equally effective and simpler to implement. The noise scale can be reduced over the course of training using a decaying factor to improve the quality of the training data. During testing, noise is not added to the actions to accurately evaluate the policy's performance.

The noise value  $v(t)$  at each time step  $t$  is updated according to:

$$v(t+1) = v(t) + Mean_{ac} \cdot (Mean - v(t)) \cdot T + std(t) \cdot \mathcal{N}(0,1) \cdot \sqrt{T} \quad (2.15)$$

- Where  $Mean_{ac}$  is the mean attraction constant which specifies how quickly the noise model output is attracted to the mean,  $Mean$  is the mean of the noise value,  $std$  is the standard deviation,  $T$  is the sampling time and  $\mathcal{N}(0,1)$  represents a normal random variable.

The standard deviation decays at each time step as:

$$std(t+1) = std(t) \cdot (1 - DecayRate) \quad (2.16)$$

It is crucial to set the noise standard deviation appropriately to promote exploration, typically setting  $std \times \sqrt{T}$  between 1% and 10% of the action range. If the agent converges on local optima too quickly, exploration can be enhanced by increasing the standard deviation or reducing the standard deviation decay rate.

DDPG agents maintain four function approximators to estimate the policy and value functions:

- **Actor  $\pi(S; \theta)$**  takes observation  $S$  and returns the corresponding action that maximizes the long-term reward.
- **Target Actor  $\pi_t(S; \theta_t)$**  Periodically updated using the latest actor parameters  $\theta$  to improve optimization stability.



- **Critic**  $Q(\mathcal{S}, \mathcal{A}; \phi)$  Takes observation  $\mathcal{S}$  and action  $\mathcal{A}$  as inputs and returns the expected long-term reward.
- **Target Critic**  $Q_t(\mathcal{S}, \mathcal{A}; \phi_t)$  Periodically updated using the latest critic parameters  $\phi$  to maintain stability.

Both the target and main networks (actors and critics) share the same structure and parameterization. Q-learning algorithms utilize target networks to stabilize training. The expression

$$r + \gamma(1 - d)\max_{a'} Q_t(s', a') \quad (2.17)$$

is referred to as the target, as minimizing the Mean Squared Bellman Error (MSBE) loss aims to align the Q-function with this target.  $d$  represents a binary indicator variable that signifies whether the next state  $s'$  is terminal or not. Specifically,  $d = 1$  if  $s'$  is a terminal state and  $d = 0$  otherwise. However, the target relies on the same parameters  $\phi$  that are being optimized, leading to instability in MSBE minimization. To address this, a secondary network called the target network, with parameters  $\phi_t$  is introduced. This target network lags behind the main network, providing more stable target values. In DQN algorithms, the target network's parameters are periodically copied from the main network. In contrast, DDPG algorithms update the target network more smoothly using Polyak averaging:

$$\phi_t = \tau\phi + (1 - \tau)\phi_t \quad (2.18)$$

where  $\tau$  is a hyperparameter between 0 and 1, referred to as smoothing factor. This continuous update method helps maintain stability during training.

The DDPG algorithm is detailed in Algorithm 2 and works as follows: Initially, both actor and critic networks, along with their corresponding target networks, are initialized with random parameters. As the agent interacts with the environment, it selects actions by adding stochastic noise to the policy output to encourage exploration. Each action results in a reward and a new state, and these experiences

are stored in a replay buffer. During training, mini-batches of experiences are sampled from this buffer to update the critic by minimizing the mean squared error between the predicted Q-values and target Q-values. The actor is updated using the policy gradient, which involves calculating the gradient of the Q-value with respect to the action and adjusting the policy parameters to increase the expected return. To maintain stability, the target networks are updated more slowly using a smoothing factor. This careful coordination between exploration, experience replay, and gradual policy improvement allows DDPG to effectively learn optimal policies in high-dimensional, continuous action spaces.

---

**Algorithm 2** DDPG algorithm
 

---

- 1: Randomly initialize critic network  $Q(s, a; \phi)$  and actor network  $\pi(s; \theta)$  with weights  $\phi$  and  $\theta$  respectively
  - 2: Initialize critic and actor target networks  $Q_t(s, a; \phi_t)$  and  $\pi_t(s; \theta_t)$  with weights  $\phi$  and  $\theta$  respectively
  - 3: Initialize replay buffer R
  - 4: **for**  $episode = 1, K$  **do**
  - 5:     Initialize a random noise  $N$  for exploration
  - 6:     Receive initial observation state  $s_1$
  - 7:     **for**  $t = 1, T$  **do**
  - 8:         Select action  $a_t = \pi(s_t; \theta) + N_t$  based on the current policy and exploration noise
  - 9:         Execute action  $a_t$  and observe reward  $r_t$  and new state  $s_{t+1}$
  - 10:        Store transition  $(s_t, a_t, r_t, s_{t+1})$  in R
  - 11:        Sample a mini batch of  $M$  transitions  $(s_i, a_i, r_i, s_{i+1})$  from R
  - 12:        Set  $y_i = r_i + \gamma Q_t(s_{i+1}, \pi_t(s_{i+1}; \theta_t); \phi_t)$
  - 13:        Update critic by minimizing the loss:  $L = \frac{1}{M} \sum_{i=1}^M (y_i - Q(s_i, a_i; \phi))^2$
  - 14:        Update the actor policy using the sampled policy gradient:
 
$$\nabla_{\theta} J \approx \frac{1}{M} \sum_{i=1}^M \nabla_a Q(s, a; \phi)|_{s=s_i, a=\pi(s_i)} \nabla_{\theta} \pi(s; \theta)|_{s_i}$$
  - 15:        Update the target networks:
 
$$\begin{aligned} \phi_t &\leftarrow \tau \phi + (1 - \tau) \phi_t \\ \theta_t &\leftarrow \tau \theta + (1 - \tau) \theta_t \end{aligned}$$
  - 16:     **end for**
  - 17: **end for**
-

### 3. Implementation and Results

In this section, the algorithmic implementation is detailed, with a particular focus on the configuration and utilization of the RL neural network. The RL neural network is a fundamental component of this approach, serving as the primary mechanism for decision-making based on the state of the environment. The architecture of the RL neural network is composed of an input layer, one or more hidden layers, and an output layer. Each of these layers is populated with a collection of interconnected nodes, commonly referred to as neurons. The configuration of these layers and neurons is crucial to the effective operation of the network.

The input layer's neuron count is determined by the dimensionality of the state representation. This state representation encapsulates the current condition of the environment, providing a comprehensive snapshot that the RL uses to make informed decisions. The output layer, on the other hand, corresponds to the action space. The number of neurons in this layer should align with the range of possible actions that the agent can take in the environment. Essentially, the neural network ingests the state representation through the input layer and, after processing through the hidden layers, produces the optimal action via the output layer. The reward function, another pivotal component of the RL paradigm, quantifies the benefit or penalty associated with the agent's actions, guiding the RL towards strategies that maximize cumulative rewards.

The process of training, tuning, and testing the RL is also elaborated in this section. Training involves adjusting the weights of the neural network based on the feedback from the reward function, while tuning refers to the fine-tuning of hyperparameters to optimize the learning process. Testing, finally, evaluates the performance of the trained and tuned RL agent in unseen scenarios, providing a measure of its generalization capability.

### 3.1 State Representation

The construction and significance of the state representation, which is a critical component of the RL approach is described here. The state representation is a 24-dimensional vector, with the features corresponding to the input layer of the neural network.

The vector encapsulates a comprehensive set of features that characterize the current state of the drone and its environment:

1. **Drone's Position:** The cartesian coordinates of the drone in XY plane.
2. **Target's Position:** The cartesian coordinates of the target in XY plane.
3. **Distance to Target:** The Euclidean distance from the drone to the target.
4. **Target Orientation:** The orientation of the target with respect to the drone, represented as an angle in the XY plane.
5. **Local Population Density:** The population density at the current drone location.
6. **Mean Population Density:** The mean population density across two sub grids of sizes 15x15 and 51x51, with the drone at the centre of each.
7. **Orientation of Maximum Population Density:** The orientation of the point of maximum population density in the sub grid with respect to the drone.
8. **Noise Level:** The overall SPL value on the ground.
9. **Obstacle Distances:** The distances from the drone to the three closest and three furthest obstacles.
10. **Obstacle Orientations:** The orientations of these six obstacles with respect to the drone.

To ensure stability and prevent divergence during the training process, each of these values is normalized to fall within the range  $[0, 1]$ . When training a RL agent using techniques like policy gradients or Q-learning, the scale of the input features can affect the scale of the gradients. If the input features have vastly different scales, it can lead to imbalanced gradients and slower convergence. Normalizing the input features ensures that the gradients are scaled appropriately, allowing for more stable and efficient learning. Normalizing input features helps in improving the generalization capability of the RL agent. By scaling the features to a similar range, the agent can learn a more generalized policy that performs well across different states or observations. Without normalization, the agent might focus more on certain

features with larger magnitudes, leading to suboptimal performance in states with different feature distributions. In summary, this state representation provides a detailed snapshot of the drone's current situation, enabling the RL neural network to make informed decisions based on the current state of the environment.

## 3.2 Action Space

The agent operates within a fixed height airspace and possesses the capability to navigate in any direction across the continuous space of the XY plane. Such an environment presents unique challenges in terms of decision-making and path planning. To effectively train the agent in this dynamic setting, a critical consideration is the selection of an appropriate sampling time,  $T$ , which dictates the frequency at which the agent takes actions. The sampling time chosen for the training phase is 0.3 units. This selection is deliberate, as a smaller  $T$  implies that the agent must make decisions more frequently. This characteristic proves advantageous in environments characterized by high dynamism or instances where the drone's situational awareness is of paramount importance. However, it's essential to acknowledge that reducing  $T$  comes with a computational cost, as the agent must process information and make decisions more rapidly. Thus, there exists a trade-off between the frequency of actions and computational efficiency.

The chosen value of  $T$  strikes a balance between responsiveness and computational feasibility, thereby providing a representative simulation of real-world scenarios. To further enhance the realism of the simulation, it's imperative to appropriately define the range of movement available to the agent at each time step. This range directly influences the agent's velocity and, by extension, its navigational capabilities. To align the simulation with real-world parameters, the range of movement is calibrated to match the velocity of commercial drones, which typically have a maximum speed of 100 km/h. By setting the action range  $[dx, dy]$  for both the  $x$  and  $y$  directions to  $[3, 3]$ , corresponding to a movement of 6 meters per time step, the simulated drone achieves a maximum velocity of approximately 101 km/h. This adjustment ensures that the agent's movement closely mirrors the operational constraints and capabilities of real-world drones, thereby enhancing the fidelity and applicability of the simulation outcomes.

### 3.3 Reward Function

The reward function plays a pivotal role in achieving the desired behavior from the model. It's crucial to set the components of the reward function to guide it towards achieving optimal path and policy. At any timestep, if the episode has terminated meaning the agent has reached the target, a substantial reward is given otherwise the reward is calculated by weighted sum of the following terms:

- **Idle penalty:**

This penalty is introduced to incentivize efficient navigation of drones within a given space. At every timestep, the current position of the drone is evaluated in comparison to its position in the previous step. This assessment determines whether the drone has remained stationary or has actively navigated through the environment. Specifically, if the coordinates of the drone at the current timestep match those of the previous step, a penalty of -1 is applied, indicating a state of idleness. Conversely, if the drone's coordinates have changed between timesteps, no penalty is applied.

$$P_{idle}(t) = \begin{cases} -1, & \text{if } p_t = p_{t-1} \\ 0, & \text{otherwise} \end{cases} \quad (3.1)$$

- Where  $p_t$  and  $p_{t-1}$  are the position of the drone at timesteps  $t$  and  $t - 1$ .

The rationale behind this penalization scheme is to motivate the drone to explore and traverse the space efficiently, avoiding prolonged periods of inactivity. By assigning a penalty for idle behavior, the algorithm encourages the drone to continuously explore its surroundings, thus maximizing the utilization of available time and resources for completing its tasks. This approach is fundamental in scenarios where time efficiency is crucial, such as surveillance, search and rescue missions, and delivery operations.

- **Distance penalty:**

This penalty is introduced to incentivize the agent to efficiently approach the target. At each step of the simulation, the Euclidean distance between the drone's current position and the target location is computed, both at the current step and the

previous step. This evaluation enables us to quantify the extent of distance reduction achieved by the drone in its trajectory towards the target. The implemented penalty mechanism is proportionate to the magnitude of this distance reduction.

$$P_{distance}(t) = - \frac{\left\| p_t - p_{target} \right\| - \left\| p_{t-1} - p_{target} \right\|}{\|p_{max}\|} \quad (3.2)$$

- Where  $p_{target}$  is the target location and  $p_{max}$  is the maximum movement range of the drone.

By applying this penalty setup, we aim to guide the drone towards the target while mitigating deviations that lead to increased distances from the target. Overall, this penalty framework serves as a critical component in optimizing the drone's navigation strategy, ensuring efficient and effective movement towards the designated target within the operational environment.

#### ▪ Population density penalty:

This penalty was implemented to guide the agent through environments with varying population densities. This is achieved by evaluating the disparity in population density between the current timestep and the preceding one. If the computed density change is positive, indicating an increase in population density at the current timestep compared to the previous one, a substantial penalty of -1 is imposed. Conversely, if the density change is negative or remains unchanged, suggesting a decrease or stability in population density, a smaller penalty of -0.1 is applied.

$$P_{density}(t) = \begin{cases} -1, & \text{if } Pd_t > Pd_{t-1} \\ -0.1, & \text{otherwise} \end{cases} \quad (3.3)$$

- Where  $Pd_t$  and  $Pd_{t-1}$  are the population density at timesteps  $t$  and  $t - 1$ .

This penalty scheme is designed to firstly, dissuade the agent from traversing through densely populated areas, which may cause annoyance to the public or pose safety concerns; and secondly, to encourage the agent to actively seek routes that lead away from high-density regions, thereby promoting efficient navigation

through the environment. By incorporating this penalty framework, our approach aims to enhance the adaptability and navigation capabilities of agents operating in diverse urban or populated settings, facilitating safer and more effective traversal through such environments.

- **Noise penalty:**

For each velocity, and consequently for each noise source SPL, the maximum possible noise level on the ground is evaluated. This maximum noise level, referred to as the threshold, is determined under the assumption that there are no obstructions along the acoustic rays' paths. At each timestep, the actual noise level generated by the agent is compared against this threshold. If the noise level exceeds the threshold, a penalty of -1 is applied, indicating an undesirable high noise impact. If the noise level is below the threshold, the penalty is calculated using the exponential function. This formula yields a penalty that decreases exponentially from -1 to 0 as the noise level decreases from the threshold to zero.

$$P_{noise}(t) = \begin{cases} -1, & \text{if } L_t > N_{thr} \\ -\exp(-3 \times \frac{N_{thr} - L_t}{N_{thr}}), & \text{otherwise} \end{cases} \quad (3.4)$$

- Where  $N_{thr}$  is the maximum possible noise and  $L_t$  is the actual noise at time  $t$ .

The purpose of this penalty structure is to incentivize the agent to choose paths where the noise levels are naturally reduced due to obstructions or other factors. By doing so, the agent is encouraged to navigate routes that mitigate ground-level noise pollution, thus promoting quieter and more environmentally friendly operations.

- **Cumulative noise penalty:**

From the beginning of each episode, the noise level at each timestep is recorded. The cumulative noise, representing the total noise produced up to the current timestep, is calculated iteratively. At each timestep, the agent is penalized based on the normalized cumulative noise, which is scaled between 0 and 1. Initially, at the first timestep, the penalty is 0. As the episode progresses, if the agent continuously generates high noise levels, the penalty increases and can theoretically reach a maximum of -1 by the last step, provided the maximum possible noise is emitted at each step.



$$P_{cum\_noise}(t) = -\frac{N_t}{t \cdot N_{max}} \quad (3.5)$$

- Where  $N_t$  is the cumulative noise up to timestep  $t$  and  $N_{max}$  is the maximum possible noise at each timestep.

This penalty mechanism serves dual purposes: firstly, it incentivizes the agent to choose paths that produce lower noise levels, thus reducing the overall acoustic impact on the environment. Secondly, it encourages the agent to reach its target as quickly as possible, since prolonged operations would result in higher cumulative noise and hence a greater penalty. By optimizing for both minimal noise production and efficient task completion, this approach aims to enhance the agent's performance in noise-sensitive environments, promoting quieter and more sustainable operations.

▪ **Smoothness penalty:**

This penalty is introduced to ensure smooth and stable navigation by the agent by evaluating the orientation of its movement vector at each timestep. For every action taken by the agent, the angle of the movement vector in a fixed frame of reference is computed. At each timestep, the difference in this angle between the current and previous steps is determined. This angular difference is then normalized to a range of  $[0, -1]$ , where a penalty of 0 corresponds to the movement vectors being aligned in consecutive steps, indicating consistent and smooth movement. Conversely, a penalty of -1 is applied when the movement vectors are in opposite directions, representing movement in complete opposite direction.

$$P_{smooth}(t) = -\frac{|\theta_t - \theta_{t-1}|}{\pi} \quad (3.6)$$

- Where  $\theta_t$  and  $\theta_{t-1}$  are the orientation angle of the movement vector at timesteps  $t$  and  $t - 1$ .

The primary goal of this penalty structure is to discourage abrupt changes in direction like sharp turns, erratic behavior and promote a smooth trajectory. By penalizing significant deviations in the movement vector's orientation, the agent is

encouraged to maintain steady and predictable motion. This approach is crucial for applications requiring precise and stable navigation, such as aerial surveillance, autonomous driving, and robotic pathfinding, where smooth trajectories enhance operational efficiency and safety.

Finally, the mathematical representation of the reward is given by:

$$R_t = \lambda_1 \cdot P_{distance}(t) + \lambda_2 \cdot P_{density}(t) + \lambda_3 \cdot P_{noise}(t) + \lambda_4 \cdot P_{cum\_noise}(t) + \lambda_5 \cdot P_{smooth}(t) + \lambda_6 \cdot P_{idle}(t) \quad (3.7)$$

- Where  $R$  is the reward and  $\lambda$  is the scaling factor.

## 3.4 Training and Hyperparameter Tuning

The training and simulations for evaluating the agent were conducted using MATLAB, an industry-standard tool for numerical computation and simulation. MATLAB's Deep Learning and Reinforcement Learning Toolboxes facilitated the development, training, and testing of the DDPG agent in a structured environment.

### 3.4.1 Training Setup

To initiate the training process, it is essential to set up the environment and configure both the actor and critic neural networks. The environment encompasses the algorithms previously described and any auxiliary functions necessary for the seamless execution of its operations. The actor and critic networks are each initialized with three hidden layers, each containing 256 nodes. This symmetry ensures consistent parameterization across both networks.

The overall dataset, consisting of 456 obstacle and population density maps, is divided into a training set of 410 maps and a testing set of 46 maps. During the training phase, obstacle and density maps and noise ray direction vectors are imported. At the beginning of each episode, a random obstacle map and its corresponding density map are selected, and the agent's starting position and target coordinates are specified randomly. Training proceeds by executing actions in the environment, observing the resultant state and reward, and continuing until the episode's termination condition is met—either the agent reaches within 2 m of the

target or the maximum of 280 steps is reached. A new episode begins with another random map and initial coordinates.

Throughout training, learning curves such as cumulative reward per episode, moving average of cumulative reward, and initial Q-value of each episode are monitored. These metrics, along with the agent's success rate, guide the continuation of training until convergence to the optimal policy is achieved.

### 3.4.2 Hyperparameter Configuration

The next step is to set the hyperparameters. Some of the main hyperparameters meticulously tuned to optimize the training process are reported in table 1 and the rationale behind them is discussed here.

For optimization, the Adam algorithm (Adaptive Moment Estimation) is employed, leveraging adaptive learning rates along with first and second moment estimates to enhance convergence speed and manage sparse gradients effectively. Both networks incorporate a gradient threshold of 1 for gradient clipping, thereby mitigating the exploding gradient problem and ensuring stable updates. Gradient explosion, characterized by the training loss becoming Inf due to exponential increases in gradient magnitude, can lead to training instability and divergence. Gradient clipping stabilizes the training process, particularly at higher learning rates and in the presence of outliers, without typically affecting the accuracy of the learned task.

The learning rate, determines the step size for each iteration of gradient descent towards minimizing the loss function. The tuned value favors training stability, where higher learning rates might initially accelerate convergence but could result in oscillations or divergence later, and lower learning rates may converge more slowly but with greater robustness.

The DDPG agent is then configured, with the first and last layers matching the observation and action space. A smoothing target update method is chosen, and the network is updated at each step. The update rate of the target networks determines how swiftly they track the main networks' parameters. A higher update rate could lead to faster convergence but might introduce instability or oscillations. Conversely, a lower update rate necessitates more iterations for convergence but offers more stability. The update rate influences the balance between exploration and exploitation during training, with faster updates promoting exploitation and slower updates encouraging exploration by providing a more stable value function estimate.

The discount factor prioritizes long-term rewards over immediate ones, encouraging the agent to develop more far-sighted policies. A higher discount factor tends to favor long-term rewards, while a lower factor focuses more on short-term gains. The replay buffer size affects the diversity of experiences the agent encounters during training. A larger buffer stores more experiences, potentially leading to better policies through diverse training data. However, excessively large buffers may slow down training and increase memory requirements.

The batch size determines the number of experiences sampled from the replay buffer for each update step. Larger batch sizes provide more accurate gradient estimates, leading to smoother updates and potentially faster convergence, though they also demand more computational resources. Smaller batch sizes, while leading to noisier gradient estimates and potentially slower convergence, can escape local minima more effectively and are more memory-efficient.

Hyperparameter		Value
<b>DDPG Algorithm</b>	Mini batch size	128
	Discount factor $\gamma$	0.99
	Smoothing factor $\tau$	$10^{-3}$
<b>Actor / Critic</b>	Learning rate $\alpha$	$10^{-5}$
	Number of hidden layers	3
	Number of nodes in each hidden layer	256

**Table 1** Key tuned hyperparameters associated with the algorithm

### 3.5 Simulation and Results

In order to validate the accuracy of the agent's learned policy, the trained agent is tested on two maps from the testing dataset. As a means to compare its performance, this trajectory is compared with the optimal path found from A\* and the shortest path which is a direct path between start and target locations.

The A\* path is designed to effectively guide the agent along an optimal path that minimizes both travel distance and population density exposure. The heuristic function in the A\* algorithm is adapted to account for both distance and population density, leveraging a pre-constructed population density map as described in chapter 2. Each grid cell in this map represents a normalized population density value, providing a detailed layout of population distribution. The cost function in the A\*

algorithm is calculated as the weighted average of the normalized population density (60%) and the Euclidean distance (40%), ensuring that both factors are consistently scaled between 0 and 1. This dual consideration allows the A\* algorithm to identify a path that balances minimal travel distance with lower population density areas.

In order to be able to make a comparison between the methods, a performance metric needs to be defined. Along the trajectory, the path length, minimum SPL, maximum SPL and SPL per unit length are recorded. The reason for the last term is because each trajectory has a different length so just comparing the cumulative noise along the path is not a fair comparison since longer paths will have higher total noise.

Initially the RL agent is tested and the noise value on the ground along the path is recorded. The optimal path based on the A\* and direct path are also found. Another challenge arises here. In order to compare the noise values, the noise data along the paths for the other two methods is needed. But since the noise source emitted SPL is dependent on velocity, a velocity needs to be defined to be able to measure the noise value along these paths. To do so, the average velocity of the RL agent is computed, which is then employed as a constant velocity throughout the entire path for the alternative methods. Using this velocity, the noise along the paths is assessed and now that all data are available, the performances can be compared and the results are visualized.

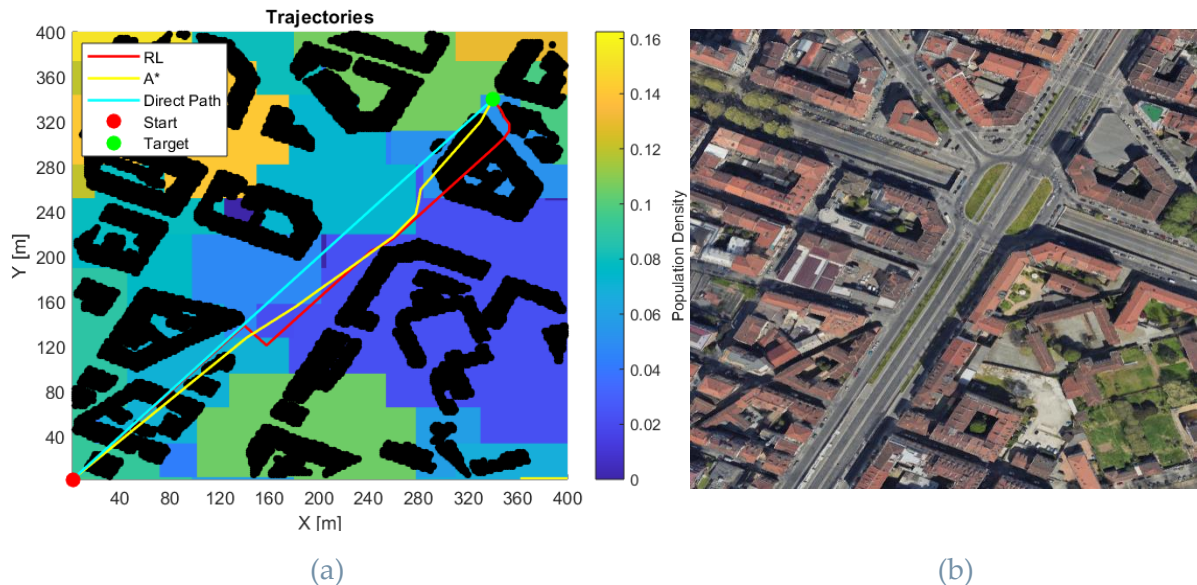


Figure 3.1 (a) Flight paths for test 1, (b) satellite photo of the test site

Two tests are performed on two separate maps and the performance metrics for the trajectories are recorded. The performance metrics reveal a nuanced picture of the RL agent's efficacy in noise management compared to the Direct and A\* path. The trajectories for the first test are depicted in figure 3.1 and the obtained results are tabulated in table 2.

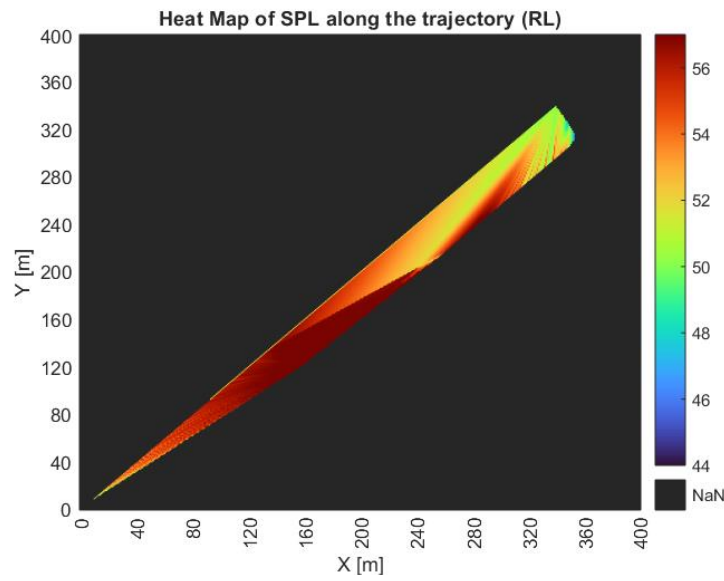
	Path Length [m]	Average Velocity [km/h]	Flight Time [s]	Minimum SPL [dB]	Maximum SPL [dB]	Total SPL [dB]	SPL per unit length [dB]
<b>Direct</b>	478.01	85.92	20.03	47.89	52.88	26829.48	56.13
<b>A*</b>	485.29	85.92	20.33	48.04	56.94	27186.92	56.02
<b>RL</b>	518.2	85.92	21.71	45.79	57.83	28542.56	55.08
<b>RL vs. Direct</b>	8.41%	0%	8.41%	- 4.39%	9.35%	6.39%	- 1.87%
<b>RL vs. A*</b>	6.78%	0%	6.78%	- 4.68%	1.56%	4.99%	- 1.68%

**Table 2** The performance metrics for test 1

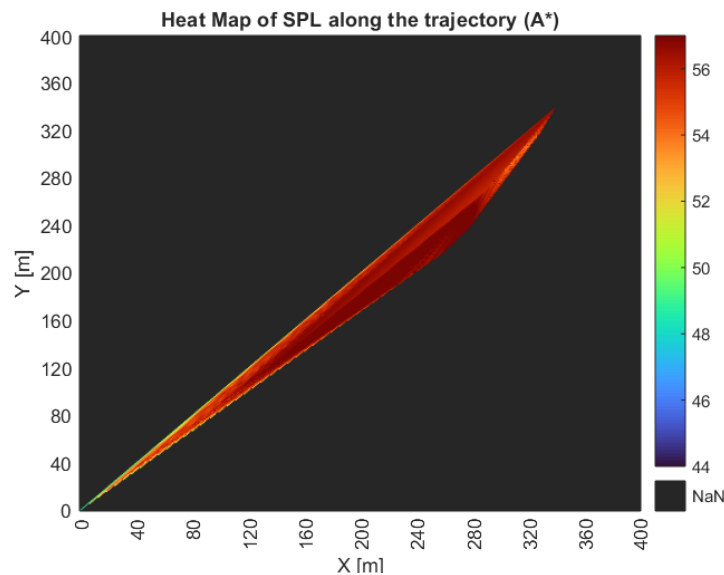
The RL agent follows the longest trajectory at 518.2 m, which is approximately 8.4% longer than the direct path and 6.8% longer than the A\* path. Consequently, the it also has the longest flight time at 21.71 seconds. While the RL agent's trajectory is the longest and results in the highest cumulative noise, it also achieves the lowest noise per unit length. The longer path suggests that the RL agent prioritizes other factors, such as noise reduction, over the shortest distance, potentially trading off travel efficiency for noise management. The direct path, while shortest in distance and flight time, does not optimize noise as effectively as the RL agent.

The RL agent achieved the lowest minimum noise level at 45.79 dB, compared to 47.89 dB and 48.04 dB for the Direct Path and A\* path. This significant reduction (~2.25 dB) indicates its capability to navigate the environment effectively and significantly reduce noise in certain segments of its path. This selective noise reduction is indicative of the agent's potential for fine-tuned control in sensitive environments. This is promising for scenarios where specific areas require stringent noise control, demonstrating the RL agent's potential for targeted noise mitigation.

The RL agent's maximum noise level is slightly higher at 57.83 dB, compared to 56.94 dB for the A\* path. The higher maximum noise encountered by the agent indicates its adaptive velocity control. By adjusting its speed based on the environmental noise sensitivity, the agent can speed up in less sensitive areas to save time and slowdown in noise-sensitive areas to minimize impact. This dynamic adaptation showcases the RL model's sophisticated approach to balancing time and noise constraints. The velocity distribution is depicted in figure 3.5.

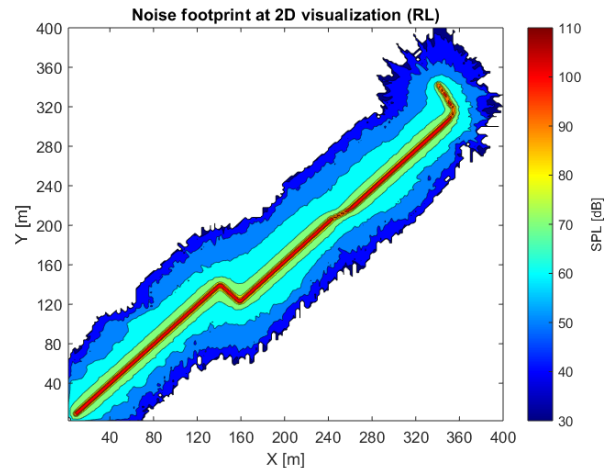


(a) RL

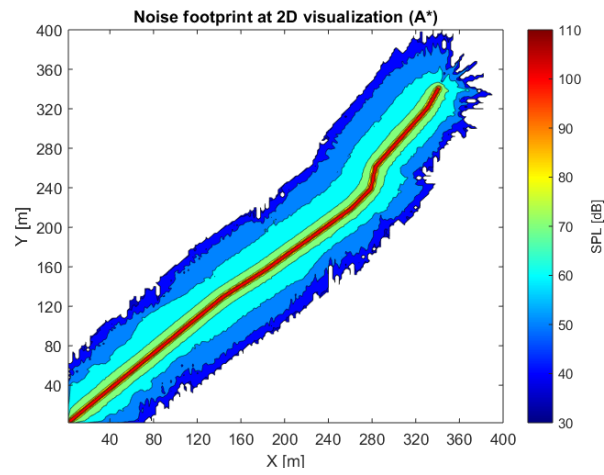


(b) A\*

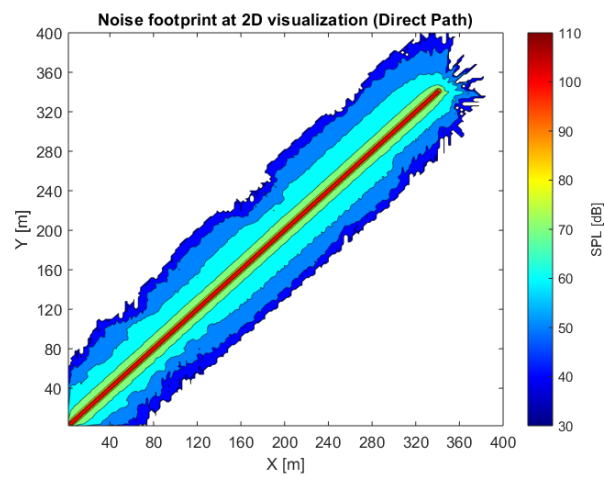
Figure 3.2 Heatmap of the SPL along the (a)RL and (b)A\* flight paths (test 1)



(a) RL



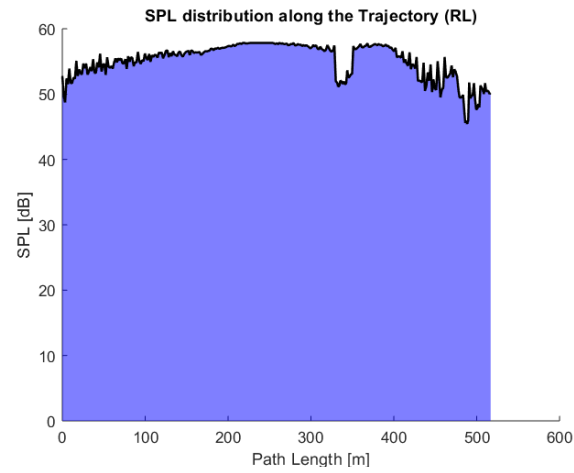
(b) A\*



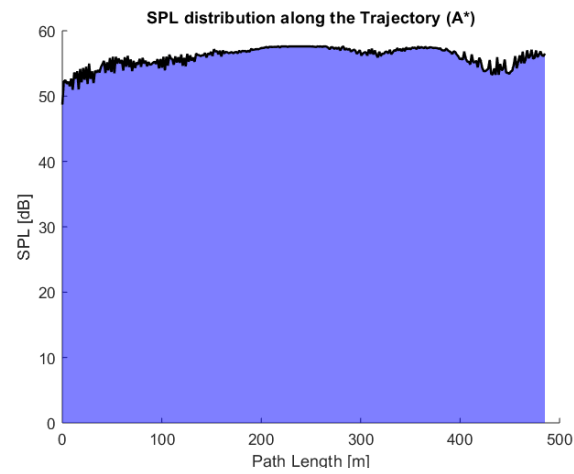
(c) Direct Path

Figure 3.3 Noise impact of the (a)RL, (b)A\* and (c)Direct flight paths in the environment at 2D visualization (test 1)

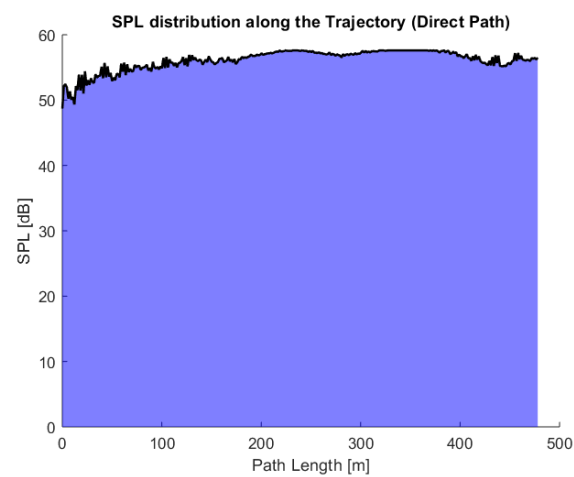




(a) RL



(b) A\*



(c) Direct Path

Figure 3.4 SPL distribution along the (a)RL, (b)A\* and (c)Direct flight paths (test 1)

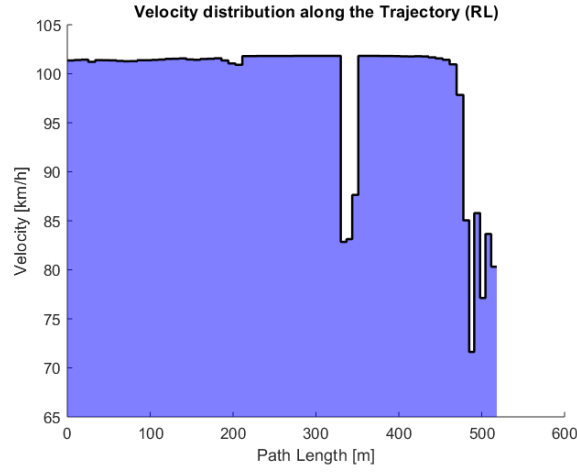


Figure 3.5 Velocity distribution along the RL flight path (test 1)

The RL agent's cumulative noise is the highest at 28542 dB, reflecting the longer path. This is 6.4% higher than the direct path and 5% higher than the A\* path. Despite higher cumulative noise, the agent excels in noise management with the lowest noise per unit length at 55.08 dB/m. This indicates an efficient distribution of noise along its trajectory, making it less impactful per unit distance traveled. The heatmap of the noise along the trajectory, the total noise impact of the path and the distribution of the SPL along the trajectory are depicted in figures 3.2, 3.3 and 3.4 respectively.

Similar pattern can be seen for the second test as well. The trajectories for the second test are depicted in figure 3.6 and the obtained results are tabulated in table 3.

	Path Length [m]	Average Velocity [km/h]	Flight Time [s]	Minimum SPL [dB]	Maximum SPL [dB]	Total SPL [dB]	SPL per unit length [dB]
<b>Direct</b>	332.97	84.04	14.27	47.89	52.87	16782.8	50.41
<b>A*</b>	335.99	84.04	14.4	47.47	52.8	16974.2	50.52
<b>RL</b>	356.2	84.04	15.25	44.76	56.13	17652.2	49.55
<b>RL vs. Direct</b>	6.98%	0%	6.98%	- 6.54%	6.17%	5.18%	- 1.71%
<b>RL vs. A*</b>	6.02%	0%	6.02%	- 5.71%	6.31%	5.11%	- 1.92%

Table 3 The performance metrics for test 2

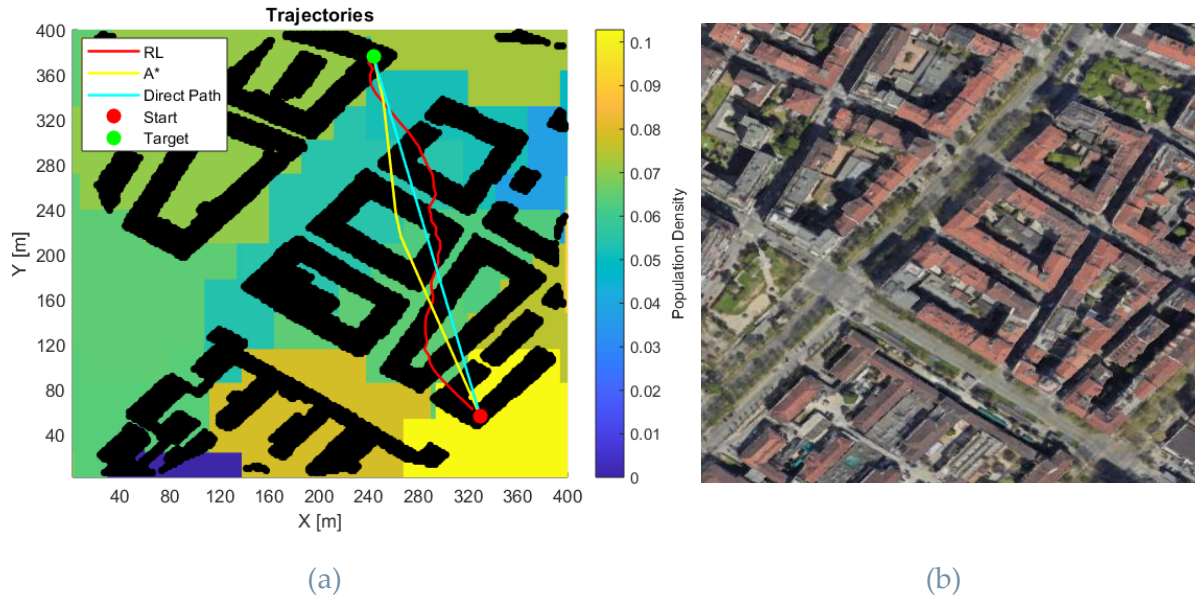
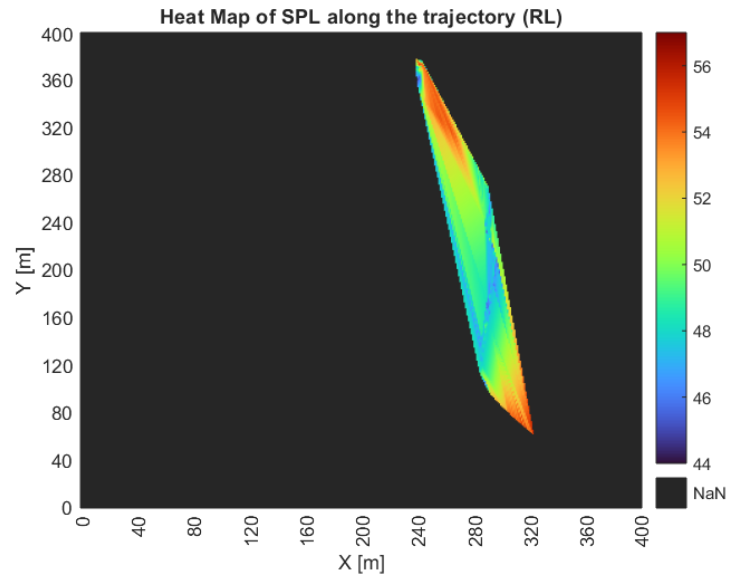


Figure 3.6 (a) Flight paths for test 2, (b) satellite photo of the test site

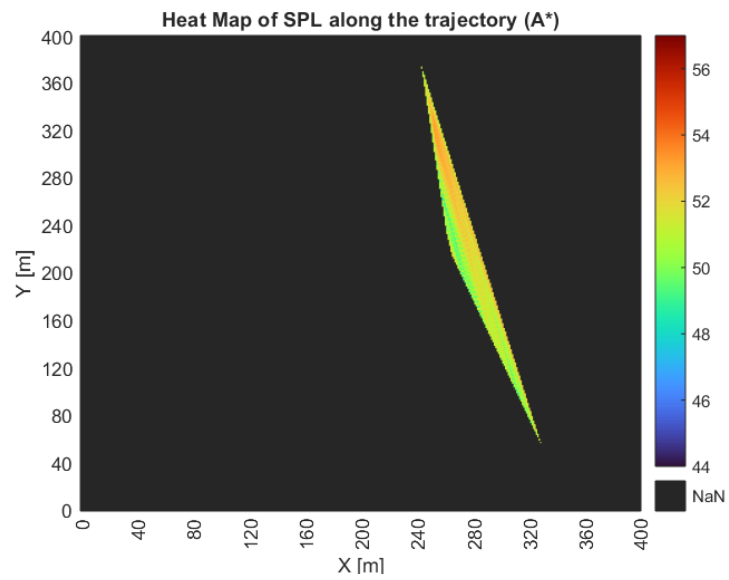
The heatmap of the noise along the trajectory, the total noise impact of the path, the distribution of the SPL along the trajectory and the velocity distribution along the flight path are depicted in figures 3.7, 3.8, 3.9 and 3.10 respectively.

The comparative analysis reveals that the RL model, despite generating a longer path and higher overall noise levels, effectively minimizes SPL per unit length and achieves lower minimum SPL values. The increased path length and flight time associated with the RL model suggest a deliberate trade-off, where the model prioritizes noise reduction over directness of the path. The higher maximum SPL observed in the RL model's path indicates occasional peaks in noise, which may be a result of the model's dynamic adjustments to minimize overall noise exposure. The reductions in SPL per unit length achieved by the RL model highlight its potential for applications where minimizing noise pollution is critical. These findings suggest that while the RL model may not always produce the shortest or fastest path, it is more effective in reducing noise impact on the ground, which is a key objective for noise-sensitive environments.

In conclusion, the RL-based navigation model demonstrates a promising approach for noise minimization in drone operations. While it currently trades off path length and cumulative noise for better noise distribution, ongoing refinement can reduce maximum SPL peaks and improve overall noise performance while maintaining efficient path lengths.

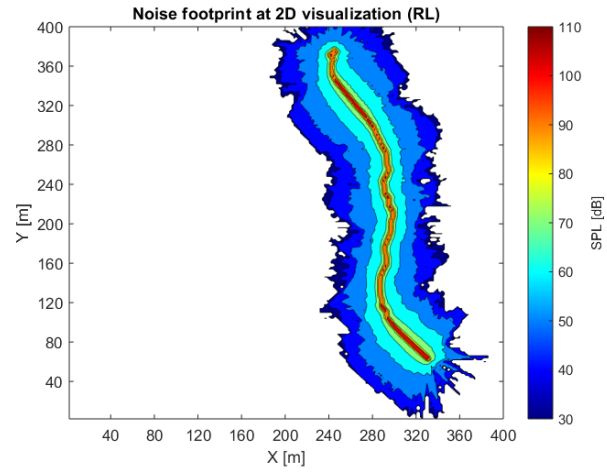


(a) RL

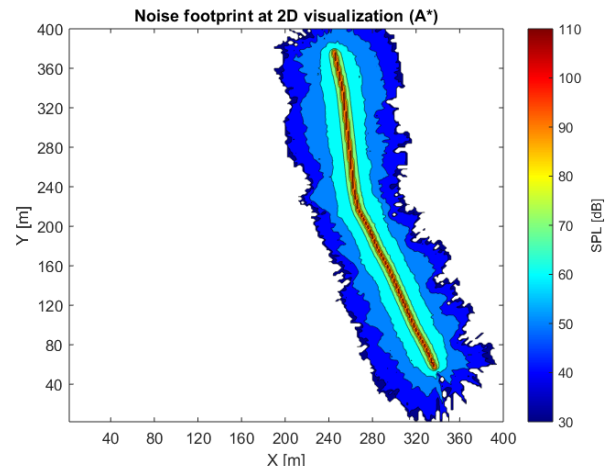


(b) A\*

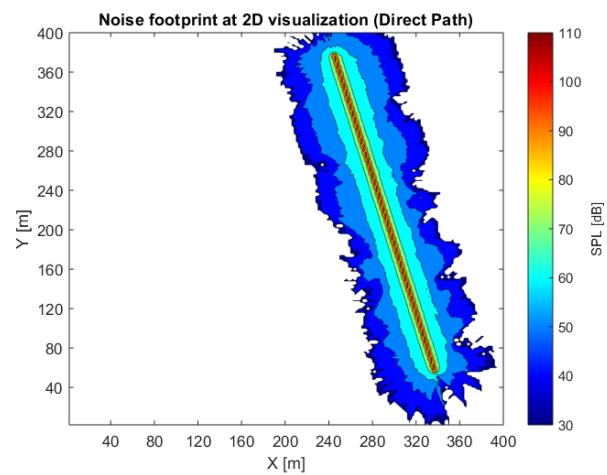
Figure 3.7 Heatmap of the SPL along the (a)RL and (b)A\* trajectories (test 2)



(a) RL

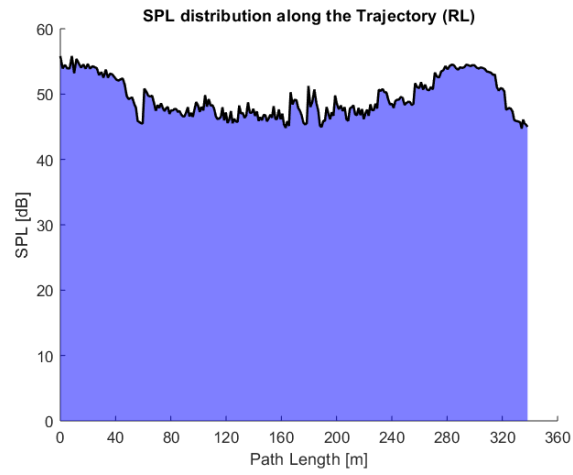


(b) A\*

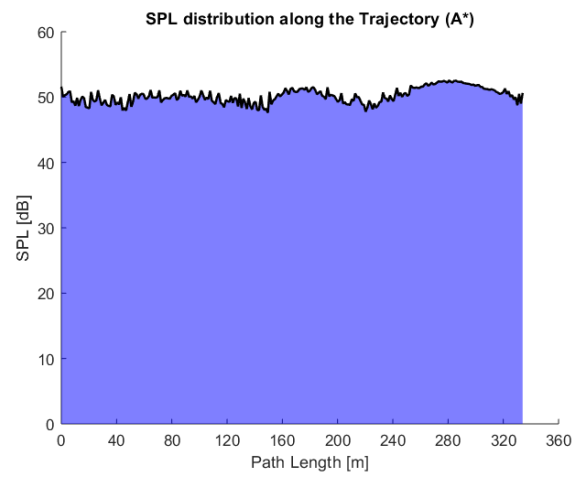


(c) Direct Path

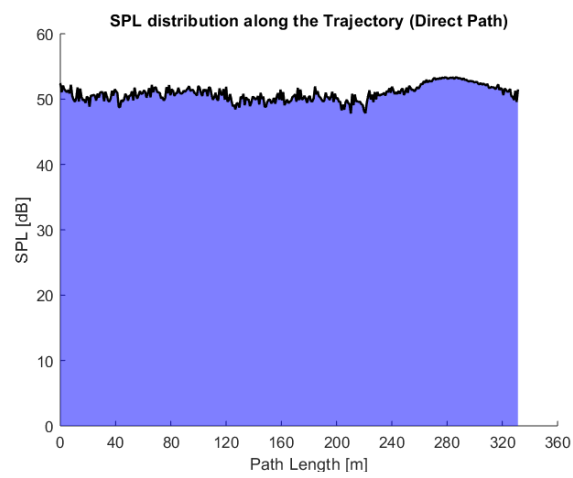
Figure 3.8 Noise impact of the (a)RL, (b)A\* and (c)Direct flight paths in the environment at 2D visualization (test 2)



(a) RL



(b) A\*



(c) Direct Path

Figure 3.9 SPL distribution along the (a)RL, (b)A\* and (c)Direct flight paths (test 2)

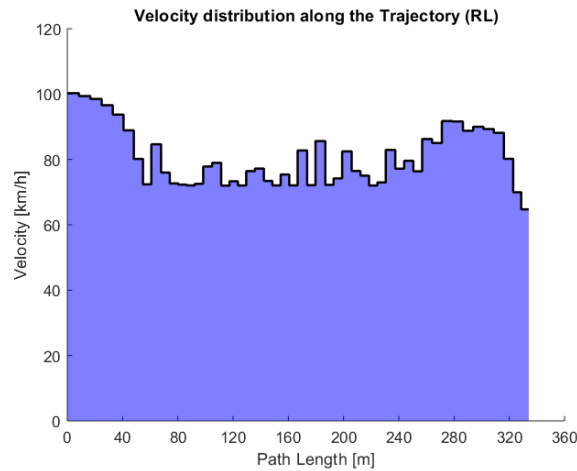


Figure 3.10 Velocity distribution along the RL flight path (test 2)

### 3.6 Model Validation

To rigorously evaluate the generalization capability and accuracy of the RL model, extensive testing was conducted on ten different maps, each featuring random start and target locations. For each map, two separate tests were performed, and the key performance metrics were recorded for the three methods. The statistical analysis, presented as the median and standard deviation (std) for each performance metric, provides a comprehensive view of the model's efficacy across varied scenarios. The statistical results are reported in the table 4 and visualized in figure 3.11.

The RL agent exhibited a median path length of 333.18 m with a standard deviation of 138.8 m. This performance closely mirrors the A\* path. In contrast, the Direct path demonstrated a shorter median path length of 295.44 m. The relatively longer path lengths of the RL and A\* trajectories indicate a prioritization of navigational strategies beyond mere distance minimization, aligning with the goal of noise reduction. The higher variability in the RL path length (std = 138.8 m) suggests an adaptive strategy, capable of tailoring paths to specific environmental conditions and noise constraints.

The RL agent achieved the lowest median minimum SPL at 45.57 dB, significantly lower than the Direct path and the A\* path. The standard deviation for the RL path's minimum SPL was 3.88 dB, slightly higher than the Direct (3.82 dB) and A\* (3.65 dB) paths. This notable reduction in minimum SPL highlights the RL agent's effectiveness in identifying possibilities to minimize the noise impact in sensitive regions.

	Path Length [m]		Minimum SPL [dB]		Maximum SPL [dB]		Total SPL [dB]		SPL per unit length [dB]	
	<i>Median</i>	<i>Std</i>	<i>Median</i>	<i>Std</i>	<i>Median</i>	<i>Std</i>	<i>Median</i>	<i>Std</i>	<i>Median</i>	<i>Std</i>
<b>Direct</b>	295.44	122.1	47.89	3.82	53.19	2.58	14925.4	6446	52.9	2.91
<b>A*</b>	336	128.28	47.47	3.65	54.19	2.7	16588.8	6794.9	52.78	2.73
<b>RL</b>	333.18	138.8	45.57	3.88	56.61	1.91	17625.4	7363.5	51.77	2.97
<b>RL vs. Direct</b>	12.78%	13.68%	- 4.8%	1.68%	6.44%	- 26%	18.09%	14.2%	- 2.1%	2.13%
<b>RL vs. A*</b>	- 0.8%	8.2%	- 4%	6.24%	4.49%	- 29%	6.25%	8.37%	- 1.9%	8.84%

**Table 4** Medians and standard deviations of the performance metrics

The RL path's median maximum SPL was 56.61 dB with a standard deviation of 1.91 dB. While this value is slightly higher than the Direct and A\* paths, the low standard deviation for the RL path suggests a consistent upper bound on noise levels. This consistency is crucial for applications requiring predictable noise management. The slightly higher maximum SPL encountered by the RL path can be attributed to its dynamic velocity adjustments, which allow for brief increases in noise to optimize overall trajectory efficiency.

The total SPL for the RL path had a median value of 17625.4 dB with a standard deviation of 7363.5 dB which is higher than both the A\* and the Direct paths. The higher total SPL is a consequence of the longer path lengths and the emphasis on reducing noise per unit length rather than total noise. The larger variability in the RL path's total SPL reflects its adaptive nature, adjusting paths to balance noise distribution effectively. The RL path excelled in minimizing noise impact per unit distance, achieving a median SPL per unit length of 51.77 dB/m with a standard deviation of 2.97 dB. This performance surpasses both the Direct and A\* paths. The lower SPL per unit length indicates a more efficient distribution of noise along the trajectory, reducing the noise impact in noise-sensitive areas despite longer overall paths.



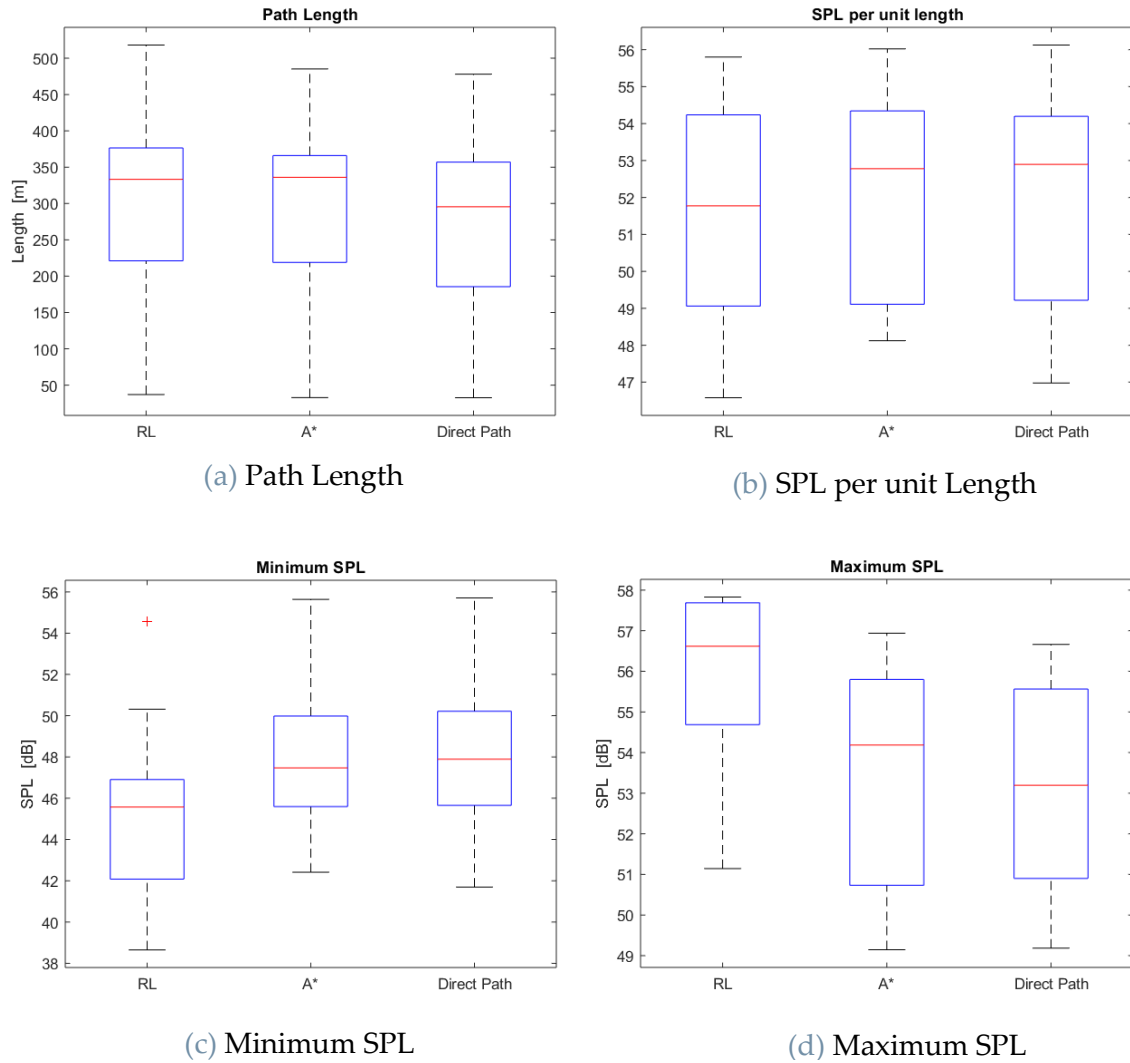


Figure 3.11 Box plot for (a)Path length, (b)SPL per unit length, (c)Minimum SPL and (d)Maximum SPL for the flight paths

The validation results confirm that the RL model demonstrates strong generalization capability and accuracy across diverse map layouts. The RL agent consistently achieves lower minimum SPL and SPL per unit length, highlighting its proficiency in managing noise effectively. While the RL paths are generally longer and exhibit higher total SPL, the significant reduction in noise per unit length underscores the agent's strategic trade-off between path length and noise distribution. The RL model's adaptability is evident from the higher variability in path lengths and total SPL, indicating its capacity to tailor trajectories to specific environmental conditions.

This flexibility is advantageous for dynamic operational environments, where noise-sensitive navigation is critical.

### 3.7 Application

The use of UAVs for delivery purposes is rapidly gaining traction in commercial sectors. However, current commercial drones typically lack advanced automatic control systems and are not optimized to achieve specific operational objectives, such as minimizing noise pollution or enhancing delivery efficiency. Incorporating the RL model into drones can significantly enhance their capabilities by providing both active control and minimizing the noise impact during flights. The model's flexibility is crucial since not all delivery packages have the same priority or urgency, necessitating adaptable control strategies. Machine learning models, particularly RL, demonstrate exceptional adaptability to varying circumstances, making them suitable for this application.

To cater to different delivery scenarios, two versions of the RL model were developed. By tailoring the reward functions of the models, effectively solutions that can be deployed in various real-world scenarios are created, each requiring different emphasis on speed and noise considerations. This differentiation allows each model to specialize in distinct delivery priorities:

1. **High-Priority Delivery:** In scenarios where timely delivery is critical, such as delivering medical supplies to a hospital, the model emphasizes on reducing delivery time over noise minimization and optimizes the flight path primarily for speed and efficiency, accepting higher noise levels as a trade-off.
2. **Low-Priority Delivery:** For deliveries that do not require urgency, such as regular postal services, the model focuses on minimizing noise impact as much as possible while still ensuring reasonable delivery times. This mode focuses on optimizing flight paths to reduce noise pollution, even if it means longer delivery times and is beneficial in residential or sensitive areas.

The tuned values for the scaling factors of the reward function for both the models are tabulated in table 6. The high priority model is the same model which was employed in the simulation and validation sections.

To validate the effectiveness of the two specialized models, a map was selected, and both versions of the RL model were tested under identical conditions, and the performance metrics were recorded for comparison. The results from these tests are

documented in the following section, providing insights into the trade-offs and performance characteristics of each model.

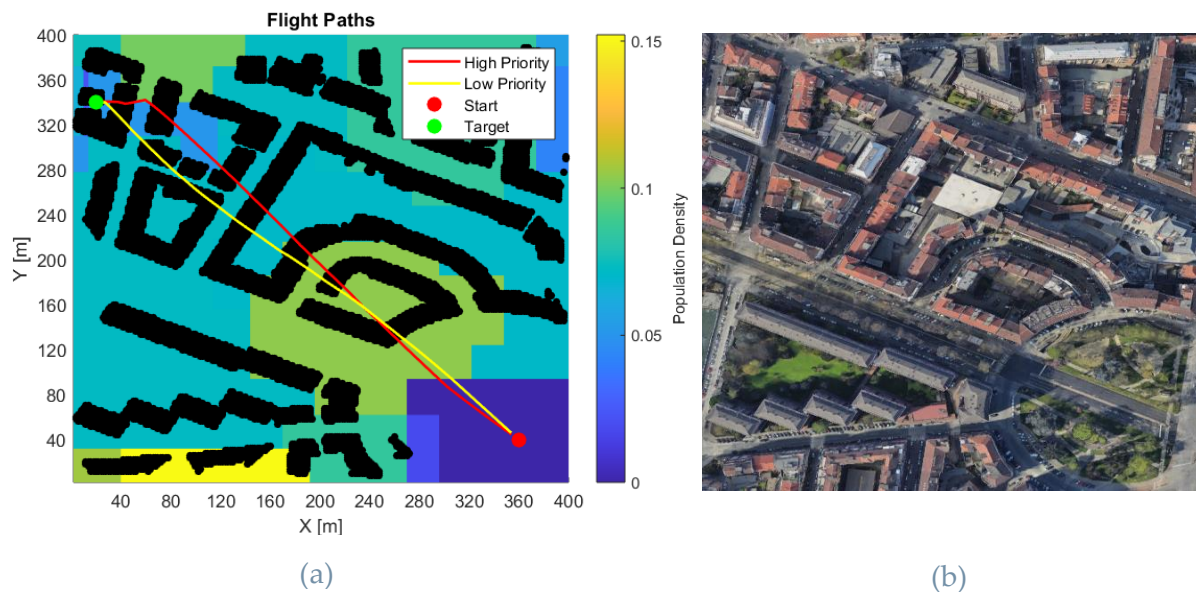
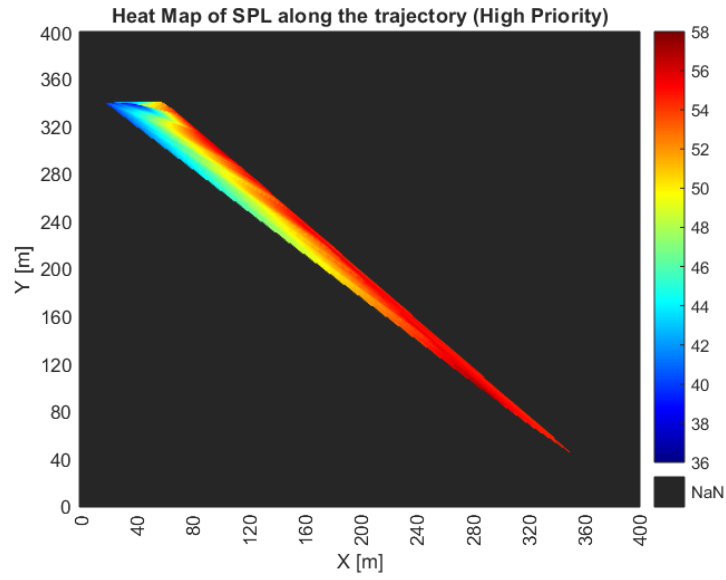
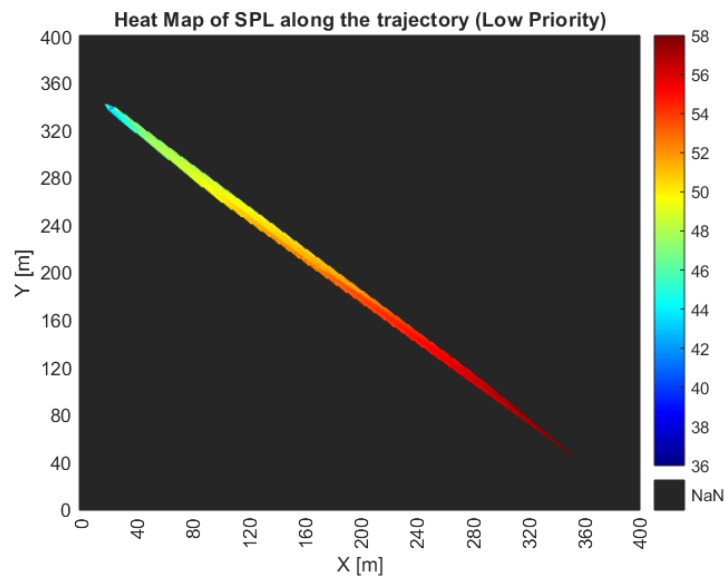


Figure 3.12 (a) Flight paths for High and Low priority models,  
(b) satellite photo of the test site

The High-Priority model prioritizes speed and efficiency. Even though this comes at the cost of increased noise levels, in emergency scenarios, this trade-off is acceptable. The data shows an average velocity of 88.17 km/h and a flight time of 18.66 seconds, which underscores the model's capability to meet urgent delivery demands. The Low-Priority model excels in minimizing noise pollution, making it suitable for routine deliveries in noise-sensitive areas. The model demonstrates a significant reduction in SPL, with a total SPL of 24114.3 dB and SPL per unit length of 52.32 dB, ensuring minimal disturbance. The Low-Priority model's focus on noise reduction makes it compliant with urban noise regulations, fostering a harmonious coexistence with the urban population. This is particularly valuable for maintaining community relations and adhering to legal requirements.

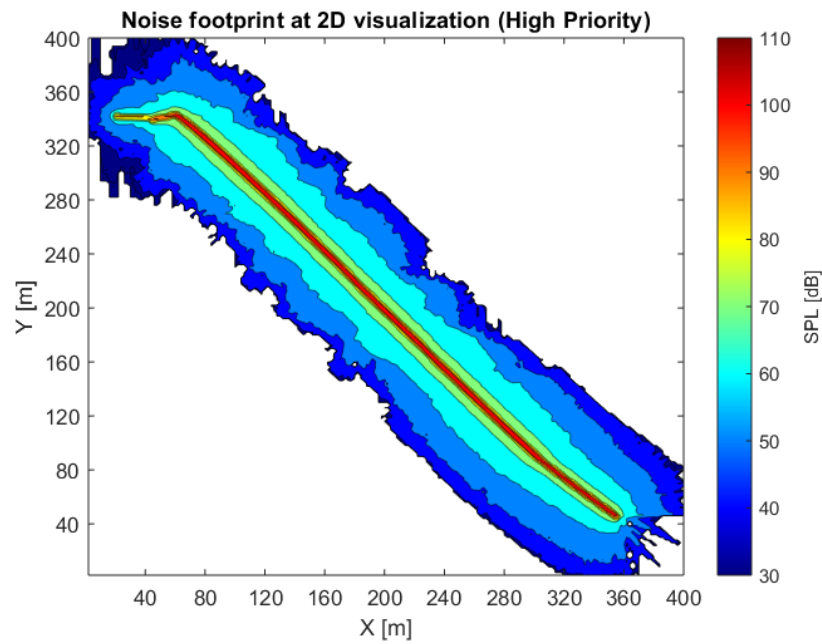


(a) High Priority

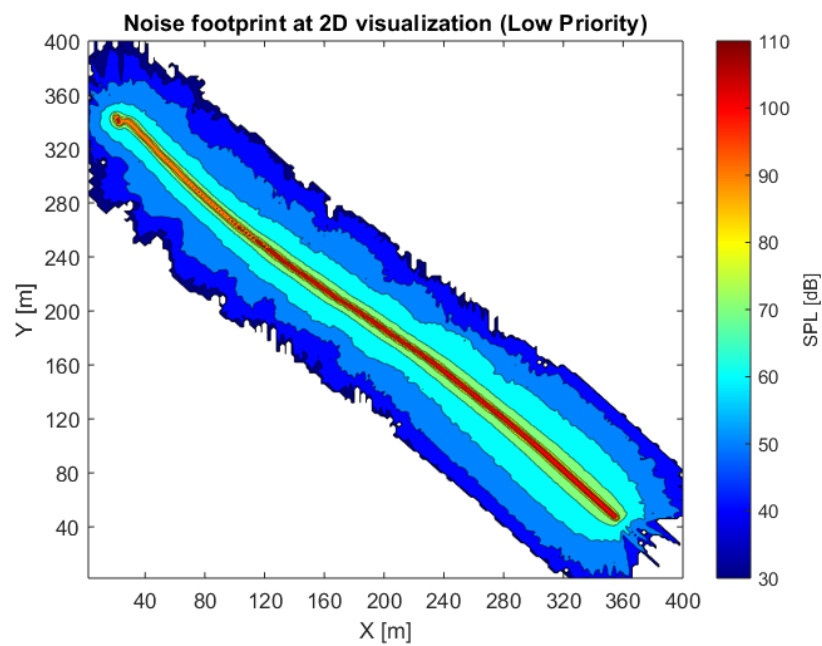


(b) Low Priority

Figure 3.13 Heatmap of the SPL along the (a)High priority and (b)Low priority flight paths

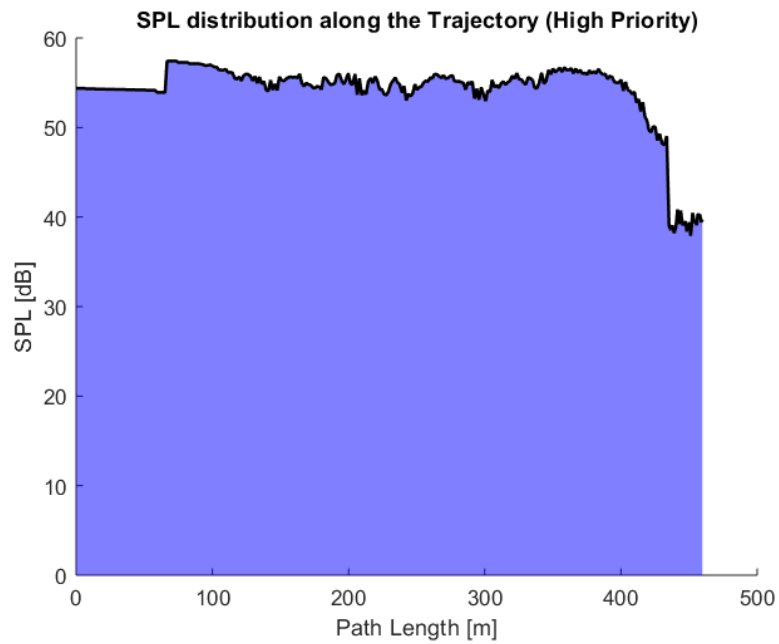


(a) High Priority

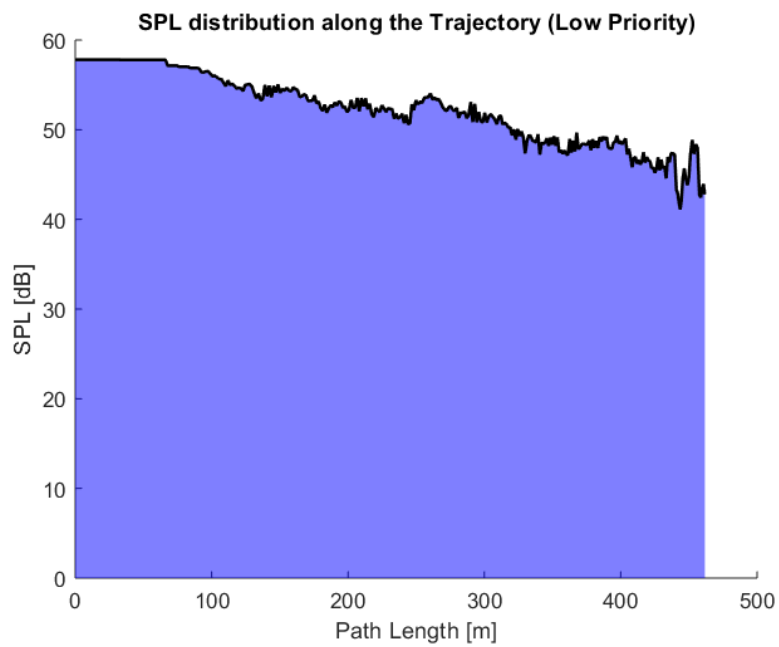


(b) Low Priority

Figure 3.14 Noise impact of the (a)High priority and (b)Low priority flight paths in the environment at 2D visualization

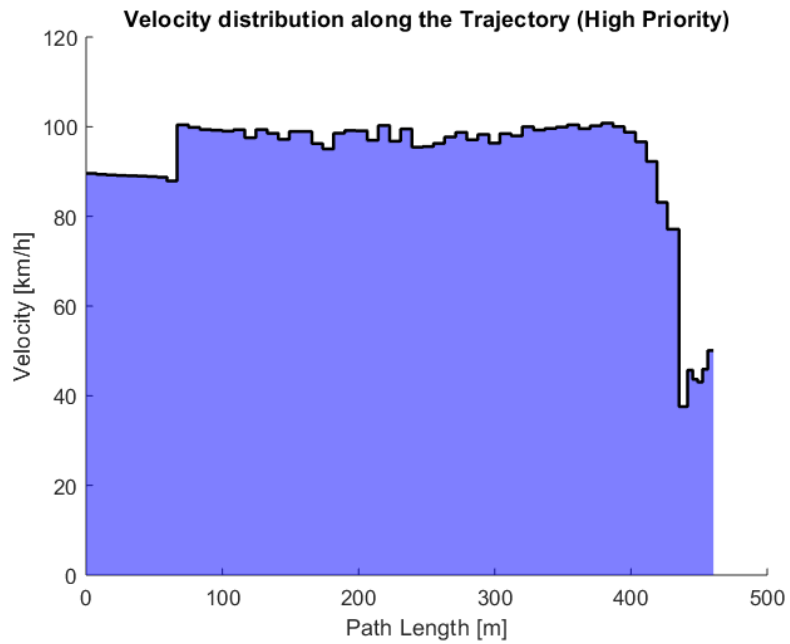


(a) High Priority

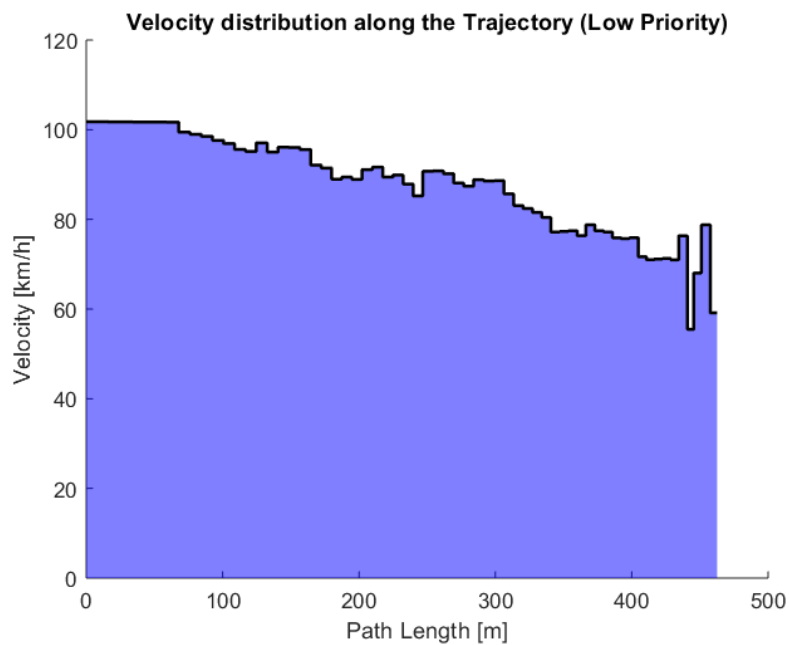


(b) Low Priority

Figure 3.15 SPL distribution along the (a)High priority and (b)Low priority flight paths



(a) High Priority



(b) Low Priority

Figure 3.16 Velocity distribution along the (a)High priority and (b)Low priority flight paths

	Path Length [m]	Average Velocity [km/h]	Flight Time [s]	Minimum SPL [dB]	Maximum SPL [dB]	Total SPL [dB]	SPL per unit length [dB]
<b>High Priority</b>	456.94	88.17	18.66	39.32	57.43	24597.1	53.83
<b>Low Priority</b>	460.9	86.74	19.13	41.59	57.81	24114.3	52.32
<b>Low vs. High</b>	0.87%	- 1.62%	2.52%	5.77%	0.66%	- 1.96%	- 2.81%

**Table 5** The performance metrics for High and Low priority models

	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_5$	$\lambda_6$
<b>High Priority</b>	0.02	0.07	0.43	0.01	0.47	1
<b>Low Priority</b>	0.02	0.06	0.48	0	0.44	1

**Table 6** The scaling factors of the reward function for High and Low priority models

The high priority model, with its higher average velocity, is more effective in reducing flight time. However, this comes at the cost of a marginally higher total SPL and SPL per unit length. Conversely, the low priority model, with its longer path and flight time, achieves lower total SPL and SPL per unit length, indicating a more efficient noise management relative to the distance traveled.

These results highlight the adaptability of the RL models in handling different delivery scenarios. Integrating RL models into UAV delivery systems offers a robust solution for achieving flexible and optimized drone operations, tailored to the specific needs of different delivery scenarios. By adapting the models for different delivery priorities, we can ensure that drones can operate efficiently in diverse environments, meeting specific operational goals while adhering to noise regulations and reducing environmental impact.



## 4. Conclusion and Future Development

### 4.1 Conclusion

The results of this study underscore the significant potential of RL for noise-aware navigation of UAVs. Through comprehensive testing on both specific and generalized scenarios, the RL agent demonstrated a sophisticated balance between path length and noise management, outperforming traditional methods such as direct paths and A\* algorithms in key aspects of noise control. The RL agent's ability to follow a longer trajectory, while reducing noise per unit length and achieving lower minimum noise levels, highlights its effectiveness in prioritizing noise-sensitive navigation. Despite the longer paths resulting in higher cumulative noise, the RL agent's capacity to minimize noise impact per distance traveled showcases a nuanced approach to noise distribution. This capability is particularly crucial in environments where minimizing noise pollution is essential, such as urban areas or noise-sensitive zones.

The comparative analysis revealed that the RL agent's trajectory, although the longest, effectively managed noise through dynamic adaptation of velocity based on environmental noise sensitivity. This adaptive control allowed the RL agent to reduce noise in critical areas while balancing overall travel efficiency. The lower minimum noise levels and efficient noise per unit length metrics indicate the RL agent's proficiency in navigating through noise-sensitive environments with minimal disruption.

Furthermore, the validation on multiple maps with varying start and target locations affirmed the RL agent's generalization capability. The consistent performance in minimizing noise and managing maximum noise levels across diverse scenarios underscores the robustness and reliability of the RL model. The adaptability in path length and cumulative noise, driven by context-aware adjustments, demonstrates the RL agent's potential to operate effectively in dynamic and varied environments. The study also highlighted the inherent trade-off between path length and noise management, a critical consideration for optimizing UAV navigation. While the RL agent currently prioritizes noise reduction over the shortest distance, ongoing

refinements could enhance its overall performance, balancing both path efficiency and noise management.

In conclusion, this research validates the efficacy of RL-based strategies for UAV navigation in noise-sensitive applications. The RL agent's advanced noise management capabilities, combined with its adaptability and generalization across different scenarios, make it a promising approach for achieving sophisticated, context-aware navigation solutions.

## 4.2 Scope for Future Work

Future advancements in RL for drone navigation could focus on improving agent performance and efficiency through several avenues. While DDPG has been effective, exploring newer algorithms such as Proximal Policy Optimization (PPO) or Trust Region Policy Optimization (TRPO) could potentially enhance training stability and convergence speed. These algorithms offer advantages like improved sample efficiency and robustness to hyperparameters. Integrating RNNs can capture temporal dependencies in drone navigation tasks, allowing the agent to better understand and react to dynamic environmental changes over time. This approach could lead to more adaptive and responsive navigation strategies, particularly in scenarios with varying noise levels or complex terrain.

Improving the accuracy and realism of noise modeling is crucial for better understanding and mitigating the environmental impact of drone operations. Instead of relying solely on simplified SPL calculations, incorporating actual noise signals emitted by drones could provide a more realistic assessment of noise impact. This involves capturing the nuances of noise generation at different velocities and maneuvers. Utilizing high-resolution maps that include vegetation density, building layouts, and terrain variations can significantly refine noise predictions. Machine learning models, integrated with lidar data and convolutional neural networks (CNNs), could automatically update and analyze these maps to dynamically adjust noise predictions based on real-time conditions.

Enhancing the granularity and informativeness of state representation and action space can lead to more effective decision-making and navigation strategies. Continuously updating local population density maps using real-time data from sensors or crowd-sourced information can improve navigation decisions, particularly in urban areas with fluctuating population densities. Expanding the action space to include multi-objective optimization, such as minimizing noise while optimizing

energy efficiency or navigating around dynamic obstacles, can broaden the applicability of drone navigation systems in diverse operational scenarios.

The future of drone navigation and noise mitigation lies in integrating advanced RL techniques, realistic noise modeling, and adaptive environmental mapping. By addressing these challenges through interdisciplinary research and technological innovation, we can pave the way for safer, more efficient, and environmentally conscious drone operations in the years to come.



## Bibliography

- [1] A. Li, M. Hansen, and B. Zou, "Traffic management and resource allocation for UAV-based parcel delivery in low-altitude urban space," *Transp Res Part C Emerg Technol*, vol. 143, p. 103808, 2022.
- [2] R. Kellermann, T. Biehle, and L. Fischer, "Drones for parcel and passenger transportation: A literature review," *Transp Res Interdiscip Perspect*, vol. 4, p. 100088, 2020.
- [3] C. A. Wargo, G. C. Church, J. Glaneueski, and M. Strout, "Unmanned Aircraft Systems (UAS) research and future analysis," in *2014 IEEE Aerospace Conference*, 2014, pp. 1–16.
- [4] A. Straubinger, E. T. Verhoef, and H. L. F. de Groot, "Going electric: Environmental and welfare impacts of urban ground and air transport," *Transp Res D Transp Environ*, vol. 102, p. 103146, 2022.
- [5] H. Eißfeldt and M. Biella, "The public acceptance of drones – Challenges for advanced aerial mobility (AAM)," *Transportation Research Procedia*, vol. 66, pp. 80–88, 2022.
- [6] C. Ramos-Romero, N. Green, S. Roberts, C. Clark, and A. J. Torija, "Requirements for Drone Operations to Minimise Community Noise Impact," *Int J Environ Res Public Health*, vol. 19, no. 15, 2022.
- [7] S. Watkins *et al.*, "Ten questions concerning the use of drones in urban environments," *Build Environ*, vol. 167, p. 106458, 2020.
- [8] E. A. King, "Here, There, and Everywhere: How the SDGs Must Include Noise Pollution in Their Development Challenges," *Environment: Science and Policy for Sustainable Development*, vol. 64, no. 3, pp. 17–32, 2022.
- [9] Q. Tan *et al.*, "Virtual flight simulation of delivery drone noise in the urban residential community," *Transp Res D Transp Environ*, vol. 118, p. 103686, 2023.
- [10] C. I. Chessell, "Propagation of noise along a finite impedance boundary," *J Acoust Soc Am*, vol. 62, no. 4, pp. 825–834, Oct. 1977.
- [11] H. Bian, Q. Tan, S. Zhong, and X. Zhang, "Assessment of UAM and drone noise impact on the environment based on virtual flights," *Aerosp Sci Technol*, vol. 118, p. 106996, 2021.
- [12] R. Kapoor, N. Kloet, A. Gardi, A. Mohamed, and R. Sabatini, "Sound Propagation Modelling for Manned and Unmanned Aircraft Noise Assessment and Mitigation: A Review," *Atmosphere (Basel)*, vol. 12, p. 1424, Jun. 2021.
- [13] J. Kennedy, S. Garruccio, and K. Cussen, "Modelling and mitigation of drone noise," *Vibroengineering PROCEDIA*, vol. 37, Jun. 2021.
- [14] "inoise: Noise prediction for industry and wind turbines."
- [15] R. Adlakha, W. Liu, S. Chowdhury, M. Zheng, and M. Nouh, "Integration of acoustic compliance and noise mitigation in path planning for drones in human–robot collaborative environments," *Journal of Vibration and Control*, vol. 29, p. 107754632211240, Jun. 2022.
- [16] H. Šiljak, K. Einicke, J. Kennedy, and S. Byrne, "Noise mitigation of UAV operations through a Complex Networks approach," Jun. 2022.
- [17] Q. Tan *et al.*, "Enhancing sustainable urban air transportation: Low-noise UAS flight planning using noise assessment simulator," *Aerosp Sci Technol*, vol. 147, p. 109071, 2024.
- [18] C. Nguyen, J. Shihua, K. Hui, and R. Liem, *Noise- and Fuel-Minimal Departure Trajectory Optimization with Reinforcement Learning*. 2023.
- [19] Q. Tan *et al.*, "Low-Noise Flight Path Planning of Drones Based on a Virtual Flight Noise Simulator: A Vehicle Routing Problem," *IEEE Intelligent Transportation Systems Magazine*, vol. PP, pp. 2–17, Jun. 2024.
- [20] Q. Tan *et al.*, "Noise assessment and low-noise flight path planning platform for urban air mobility," *J Acoust Soc Am*, vol. 154, pp. A293–A293, Jun. 2023.
- [21] G. 'Scozzaro, D. 'Delahaye, and A. 'Ernesto Vela, "Noise Abatement Trajectories for a UAV Delivery Fleet," *9th SESAR Innovation Days*, Dec. 2019.

- [22] D. Scott, S. G. Manyam, D. W. Casbeer, M. Kumar, M. J. Rothenberger, and I. E. Weintraub, "Power Management for Noise Aware Path Planning of Hybrid UAVs," in *2022 American Control Conference (ACC)*, 2022, pp. 4280–4285.
- [23] M. B. 'Galles, "Reducing the Noise Impact of Unmanned Aerial Vehicles by Flight Control System Augmentation," Old Dominion University, 2019.
- [24] B. Pang, X. Hu, W. Dai, and K. H. Low, "UAV path optimization with an integrated cost assessment model considering third-party risks in metropolitan environments," *Reliab Eng Syst Saf*, vol. 222, p. 108399, 2022.
- [25] N. Hohmann, M. Bujny, J. Adamy, and M. Olhofer, "Multi-objective 3D Path Planning for UAVs in Large-Scale Urban Scenarios," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, 2022, pp. 1–8.
- [26] H. 'Wu *et al.*, "The multifunctional rotor aerodynamic and aeroacoustic test platform at HKUST," *NOISE-CON Paper*, pp. 4410–4417, 2021.
- [27] H. Jiang, X. Zhang, and X. Huang, "Reduced-basis boundary element method for efficient broadband acoustic simulation," *J Sound Vib*, vol. 456, pp. 374–385, 2019.
- [28] Q. Tan *et al.*, "Virtual flight simulation of delivery drone noise in the urban residential community," *Transp Res D Transp Environ*, vol. 118, p. 103686, 2023.
- [29] Q. 'Tan, H. 'Bian, H. 'Jiang, H. 'Lo, S. 'Zhong, and X. 'Zhang, "A virtual flight simulation platform for community drone noise assessment," *Quiet Drones 2022*, 2022.
- [30] N. Kloet, S. Watkins, X. Wang, S. Prudden, R. Clothier, and J. Palmer, "DRONE ON: A PRELIMINARY INVESTIGATION OF THE ACOUSTIC IMPACT OF UNMANNED AIRCRAFT SYSTEMS (UAS)," Jun. 2017.
- [31] A. Mohamud and A. Ashok, "Drone Noise Reduction through Audio Waveguiding," Jun. 2018, pp. 92–94.
- [32] R. Noda *et al.*, "Development of Bio-Inspired Low-Noise Propeller for a Drone," *Journal of Robotics and Mechatronics*, vol. 30, pp. 337–343, Jun. 2018.
- [33] P. Candeloro, T. Pagliaroli, D. Ragni, and S. Di Francesco, "Small-Scale Rotor Aeroacoustics for Drone Propulsion: A Review of Noise Sources and Control Strategies," Jun. 2019.
- [34] J. Sun, K. Yonezawa, E. Shima, and H. Liu, "Integrated Evaluation of the Aeroacoustics and Psychoacoustics of a Single Propeller," *Int J Environ Res Public Health*, vol. 20, p. 1955, Jun. 2023.
- [35] P. Gupta, "Different Techniques of Secondary Path Modeling for Active Noise Control System: A Review," *International Journal of Engineering Research & Technology*, vol. 5, pp. 611–616, Jun. 2016.
- [36] H. Bi, F. Ma, T. Abhayapala, and P. Samarasinghe, "Spherical Sector Harmonics Based Directional Drone Noise Reduction," Jun. 2022, pp. 1–5.
- [37] M. D. Narine, M. D. Narine, A. G. Bourgeois, and A. A. Y. Mussa, "Active Noise Cancellation of Drone Propeller Noise through Waveform Approximation and Pitch-Shifting," 2020. [Online]. Available: <https://api.semanticscholar.org/CorpusID:235670849>
- [38] H. Ahn, D.-T. Le, B. Dang, S. Kim, and H. Choo, "Hybrid Noise Reduction for Audio Captured by Drones," Jun. 2018, pp. 1–4.
- [39] S. K. G. Kwangjin Yang and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.
- [40] S. Bhandari, J. Farinella, and C. Lay, "UAV Collision Avoidance using a Predictive Rapidly-Exploring Random Tree (RRT)," Jun. 2016.
- [41] L. Gonzalez, "Adaptive dynamic path re-planning RRT algorithms with game theory for UAVs," Jun. 2013.
- [42] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning.," in *Proceedings - IEEE International Conference on Robotics and Automation*, Jun. 2000, pp. 995–1001.
- [43] D. Zhang, Y. Xu, and X. Yao, "An Improved Path Planning Algorithm for Unmanned Aerial Vehicle Based on RRT-Connect," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 4854–4858.
- [44] N. Wen, L. Zhao, X. Su, and P. Ma, "UAV online path planning algorithm in a low altitude dangerous environment," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 173–185, 2015.
- [45] Y. Lin and S. Saripalli, "Sampling-Based Path Planning for UAV Collision Avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, pp. 1–14, Jun. 2017.
- [46] H. Yang, Q. Jia, and W. Zhang, "An Environmental Potential Field Based RRT Algorithm for UAV Path Planning," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 9922–9927.

- [47] W. Zu, Guoliang. Fan, Y. Gao, Y. Ma, H. Zhang, and H. Zeng, "Multi-UAVs Cooperative Path Planning Method based on Improved RRT Algorithm," in *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2018, pp. 1563–1567.
- [48] L. E. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan, "Randomized Query Processing in Robot Path Planning," *J Comput Syst Sci*, vol. 57, no. 1, pp. 50–60, 1998.
- [49] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *International Journal of Robotic Research - IJRR*, vol. 30, pp. 846–894, Jun. 2011.
- [50] P. Pettersson and P. Doherty, "1 Probabilistic Roadmap Based Path Planning for an Autonomous Unmanned Helicopter," *Journal of Intelligent and Fuzzy Systems*, vol. 17, pp. 395–405, Jun. 2006.
- [51] Á. Madridano, A. Al-Kaff, D. Martín Gómez, and A. de la Escalera, "3D Trajectory Planning Method for UAVs Swarm in Building Emergencies," *Sensors*, vol. 20, p. 642, Jun. 2020.
- [52] S. Hrabar, "3D Path Planning and Stereo-based Obstacle Avoidance for Rotorcraft UAVs," Jun. 2008, pp. 807–814.
- [53] D.-S. Jang, H.-J. Chae, and H.-L. Choi, "Optimal control-based UAV path planning with dynamically-constrained TSP with neighborhoods," Jun. 2017, pp. 373–378.
- [54] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2986–2993.
- [55] W. Meng, Z. He, R. Su, and L. Xie, "Decentralized Multi-UAV Flight Autonomy for Moving Convoys Search and Track," *IEEE Transactions on Control Systems Technology*, vol. PP, Jun. 2016.
- [56] X. Hu, B. Pang, F. Dai, and K. H. Low, "Risk Assessment Model for UAV Cost-Effective Path Planning in Urban Environments," *IEEE Access*, vol. PP, p. 1, Jun. 2020.
- [57] T. Misa, "An Interview with Edsger W. Dijkstra," *Commun. ACM*, vol. 53, pp. 41–47, Jun. 2010.
- [58] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," 2022, pp. 287–290.
- [59] F. Medeiros and J. Silva, "A Dijkstra Algorithm for Fixed-Wing UAV Motion Planning Based on Terrain Elevation," Jun. 2010.
- [60] J. Economou, G. Kladis, A. Tsourdos, and B. White, "UAV optimum energy assignment using Dijkstra's Algorithm," Jun. 2007, pp. 287–292.
- [61] P. E. 'Hart, N. J. 'Nilsson, and B. 'Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.* 4 (2), pp. 100–107, 1968.
- [62] A. Stentz, "Optimal and Efficient Path Planning for Unknown and Dynamic Environments," *Proc IEEE Int Conf on Robotics and Automation*, vol. 10, Jun. 2003.
- [63] F. Islam, V. Narayanan, and M. Likhachev, "A\*-Connect: Bounded suboptimal bidirectional heuristic search," Jun. 2016, pp. 2752–2758.
- [64] T. Chen, G. Zhang, X. Hu, and J. Xiao, "Unmanned aerial vehicle route planning method based on a star algorithm," Jun. 2018, pp. 1510–1514.
- [65] Z. He and L. Zhao, "The Comparison of Four UAV Path Planning Algorithms Based on Geometry Search Algorithm," in *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2017, pp. 33–36.
- [66] S. Benders and S. Schopferer, "A Line-Graph Path Planner for Performance Constrained Fixed-Wing UAVs in Wind Fields," Jun. 2017.
- [67] N. Wen, X. Su, P. Ma, L. Zhao, and Y. Zhang, "Online UAV path planning in uncertain and hostile environments," *International Journal of Machine Learning and Cybernetics*, vol. 8, Jun. 2015.
- [68] B. Pang, X. Hu, W. Dai, and K. H. Low, "UAV path optimization with an integrated cost assessment model considering third-party risks in metropolitan environments," *Reliab Eng Syst Saf*, vol. 222, p. 108399, 2022.
- [69] C. 'Neto, G. 'Bertoli, and O. 'Saotome, "A-star path planning simulation for UAS Traffic Management (UTM) application," 2021.
- [70] L. Magatão, "Mixed integer linear programming and constraint logic programming: towards a unified modeling framework," 2005. [Online]. Available: <https://api.semanticscholar.org/CorpusID:7086435>
- [71] B. Song, K. Park, and J. Kim, "Persistent UAV delivery logistics: MILP formulation and efficient heuristic," *Comput Ind Eng*, vol. 120, Jun. 2018.
- [72] A. 'Stalmakou, "Uav/uas path planning for ice management information gathering," Institutt for teknisk kybernetikk, 2011.
- [73] W. A. Kamal, D.-W. Gu, and I. Postlethwaite, "Real-time trajectory planning for UAVs using MILP," Jun. 2006, pp. 3381–3386.
- [74] N. Mathew, S. Smith, and S. Waslander, "Multirobot Rendezvous Planning for Recharging in Persistent Tasks," *submitted to IEEE Transactions on Robotics*, vol. 31, Jun. 2013.

- [75] J. De Waen, H. T. Dinh, M. H. Cruz Torres, and T. Holvoet, "Scalable multirotor UAV trajectory planning using mixed integer linear programming," in *2017 European Conference on Mobile Robots (ECMR)*, 2017, pp. 1–6.
- [76] M. Alighanbari, Y. Kuwata, and J. How, "Coordination and Control of Multiple UAVs with Timing Constraints and Loitering," in *Proceedings of the American Control Conference*, Jun. 2003, pp. 5311–5316 vol.6.
- [77] Y. Kim, D.-W. Gu, and I. Postlethwaite, "Real-Time Optimal Mission Scheduling and Flight Path Selection," *Automatic Control, IEEE Transactions on*, vol. 52, pp. 1119–1123, Jun. 2007.
- [78] E. Grötli and T. Johansen, "Path planning for UAVs under communication constraints using SPLAT! and MILP," *J Intell Robot Syst*, vol. 65, pp. 265–282, Jun. 2012.
- [79] M. Radmanesh and M. Kumar, "Flight formation of UAVs in presence of moving obstacles using fast-dynamic mixed integer linear programming," *Aerosp Sci Technol*, vol. 50, Jun. 2015.
- [80] M. Radmanesh, M. Kumar, A. Nemat, and M. Sarim, "Dynamic optimal UAV trajectory planning in the National Airspace System via mixed integer linear programming," *Proc Inst Mech Eng G J Aerosp Eng*, vol. 230, Jun. 2015.
- [81] C. Branca and R. Fierro, "A hierarchical optimization algorithm for cooperative vehicle networks," Jun. 2006, p. 6 pp.
- [82] M. Earl and R. D'Andrea, "Iterative MILP Methods for Vehicle Control Problems," *Robotics, IEEE Transactions on*, vol. 21, pp. 1158–1167, Jun. 2006.
- [83] M. Darrah, W. Niland, and B. Stolarik, "Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission," Jun. 2005.
- [84] M. Rinaldi, S. Primatesta, M. Bugaj, J. Rostáš, and G. Guglieri, "Development of Heuristic Approaches for Last-Mile Delivery TSP with a Truck and Multiple Drones," *Drones*, vol. 7, no. 7, 2023.
- [85] M. Rinaldi, S. Primatesta, G. Guglieri, and A. Rizzo, "Auction-based Task Allocation for Safe and Energy Efficient UAS Parcel Transportation," *Transportation Research Procedia*, vol. 65, pp. 60–69, 2022.
- [86] M. Rinaldi, S. Primatesta, G. Guglieri, and A. Rizzo, "Multi-Auctioneer Market-based Task Scheduling for Persistent Drone Delivery," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2023, pp. 790–797.
- [87] I. Hasircioglu, H. R. Topcuoglu, and M. Ermis, "3-D path planning for the navigation of unmanned aerial vehicles by using evolutionary algorithms," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, in GECCO '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 1499–1506.
- [88] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, and C. Lucas, "Memetic Algorithm Based Path Planning for a Mobile Robot," in *International Journal of Information Technology*, Jun. 2004, pp. 56–59.
- [89] T. Ho, E. Grötli, S. P.B, T. Johansen, and J. Sousa, "Performance Evaluation of Cooperative Relay and Particle Swarm Optimization Path Planning for UAV and Wireless Sensor Network," *2013 IEEE Globecom Workshops, GC Wkshps 2013*, Jun. 2013.
- [90] Y. Wang, P. Bai, X. Liang, W. Wang, J. Zhang, and Q. Fu, "Reconnaissance Mission Conducted by UAV Swarms Based on Distributed PSO Path Planning Algorithms," *IEEE Access*, vol. PP, Jun. 2019.
- [91] S. Shao, Y. Peng, C. He, and Y. Du, "Efficient path planning for UAV formation via comprehensively improved particle swarm optimization," *ISA Trans*, vol. 97, pp. 415–430, 2020.
- [92] L. Zhang, J. Chen, and F. Deng, "Aircraft trajectory planning for improving vision-based target geolocation performance," in *2017 13th IEEE International Conference on Control & Automation (ICCA)*, 2017, pp. 379–384.
- [93] J. Chen, F. Ye, and Y. Li, "Travelling salesman problem for UAV path planning with two parallel optimization algorithms," in *2017 Progress in Electromagnetics Research Symposium - Fall (PIERS - FALL)*, 2017, pp. 832–837.
- [94] Ma, Guanjun, Duan, Haibin, and L. Senqi, "Improved Ant Colony Algorithm for Global Optimal Trajectory Planning of UAV under Complex Environment.," *International Journal of Computer Science & Applications*, vol. 4, Jun. 2007.
- [95] F. Ling, J. Chen, and C. Du, "Multi-obstacle Path Planning of UAV Based on Improved Ant Colony System Algorithm," Jun. 2020, pp. 1731–1735.
- [96] S. Konatowski, "Application of the ACO algorithm for UAV path planning," *PRZEGLĄD ELEKTROTECHNICZNY*, vol. 1, pp. 117–121, Jun. 2019.
- [97] C. M. Eaton, E. K. P. Chong, and A. A. Maciejewski, "Multiple-Scenario Unmanned Aerial System Control: A Systems Engineering Approach and Review of Existing Control Methods," *Aerospace*, vol. 3, no. 1, 2016.



- [98] O. Sahingoz, "Flyable path planning for a multi-UAV system with Genetic Algorithms and Bezier curves," in *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, Jun. 2013, pp. 41–48.
- [99] Z. Cheng, Y. Sun, and Y. Liu, "Path planning based on immune genetic algorithm for UAV," Jun. 2011.
- [100] V. Pehlivanoglu, "A new vibrational genetic algorithm enhanced with a Voronoi diagram for path planning of autonomous UAV," *Aerospace Science and Technology - AEROSP SCI TECHNOL*, vol. 16, Jun. 2011.
- [101] U. Cekmez, M. Ozsiginan, and O. Sahingoz, "Adapting the GA approach to solve Traveling Salesman Problems on CUDA architecture," Jun. 2013, pp. 423–428.
- [102] M. Gemeinder and M. Gerke, "GA-based path planning for mobile robot systems employing an active search algorithm," *Appl. Soft Comput.*, vol. 3, pp. 149–158, Jun. 2003.
- [103] S.-Y. Fu, L. Han, Y. Tian, and G.-S. Yang, "Path planning for unmanned aerial vehicle based on genetic algorithm," in *Proceedings of the 11th IEEE International Conference on Cognitive Informatics and Cognitive Computing, ICCI\*CC 2012*, Jun. 2012, pp. 140–144.
- [104] Y. Wang and W. Chen, "Path planning and obstacle avoidance of unmanned aerial vehicle based on improved genetic algorithms," Jun. 2014, pp. 8612–8616.
- [105] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," in *Proceedings - IEEE International Conference on Robotics and Automation*, Jun. 1991, pp. 1398–1404 vol.2.
- [106] I. Nikolos, E. Zografos, and A. Brintaki, "UAV Path Planning Using Evolutionary Algorithms," in *Studies in Computational Intelligence*, vol. 70, 2007, pp. 77–111.
- [107] E. Zarbaf, M. Norouzi, R. Allemang, V. Hunt, and A. Helmicki, "Stay Cable Tension Estimation of Cable-Stayed Bridges Using Genetic Algorithm and Particle Swarm Optimization," *Journal of Bridge Engineering*, vol. 22, p. 5017008, Jun. 2017.
- [108] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.
- [109] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [110] S. Luan, Y. Yang, H. Wang, B. Zhang, B. Yu, and C. He, "3D G-learning in UAVs," in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2017, pp. 953–957.
- [111] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," in *Proceedings of 32nd IEEE Conference on Decision and Control*, 1993, pp. 395–400 vol.1.
- [112] C. Yan and X. Xiaojia, "A Path Planning Algorithm for UAV Based on Improved Q-Learning," Jun. 2018, pp. 1–5.
- [113] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "Q-learning based Routing Scheduling for a Multi-Task Autonomous Agent," Jun. 2019, pp. 634–637.
- [114] V. Sichkar, "Reinforcement Learning Algorithms in Global Path Planning for Mobile Robot," Jun. 2019, pp. 1–5.
- [115] P. Abbeel, A. Coates, M. Quigley, and A. Ng, "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," in *Advances in Neural Information Processing Systems*, Jun. 2006, pp. 1–8.
- [116] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement Learning for UAV Attitude Control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, Jun. 2018.
- [117] Z. Goddard *et al.*, "Utilizing Reinforcement Learning to Continuously Improve a Primitive-Based Motion Planner," Jun. 2021.
- [118] R. Chai, A. Savvaris, A. Tsourdos, and S. Chai, "Multi-objective trajectory optimization of Space Maneuver Vehicle using adaptive differential evolution and modified game theory," *Acta Astronaut.*, vol. 136, Jun. 2017.
- [119] L. N. Yang and B. M. Shield, "DEVELOPMENT OF A RAY TRACING COMPUTER MODEL FOR THE PREDICTION OF THE SOUND FIELD IN LONG ENCLOSURES," 2000. [Online]. Available: <http://www.idealibrary.comon>
- [120] E.M. Salomons, *Computational Atmospheric Acoustics*, no. 1. Sci Publ House, 2012.
- [121] A. Krokstad, S. Str, and S. Sbrsdal, "CALCULATING THE ACOUSTICAL ROOM RESPONSE BY THE USE OF A RAY TRACING TECHNIQUE," 1968.
- [122] J. B. Allen and D. A. Berkley, "Image method for efficiently simulating small-room acoustics," *Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.
- [123] "Acoustics - Attenuation of sound during propagation outdoors, ISO 9613-2," 1996.

- [124] Francis Besnard *et al.*, *Road noise prediction 2 - Noise propagation computation method including meteorological effects (NMPB 2008)*. SETRA, 2009.
- [125] D. Vanmaercke and J. Defrance, "Development of an Analytical Model for Outdoor Sound Propagation Within the Harmonoise Project," 2007.
- [126] H. G. Jonasson and S. Storeheier, "Nord 2000. New Nordic Prediction Method for Road Traffic Noise," pp. 2001–2013, 2001.
- [127] F. Yunus, D. Casalino, F. Avallone, and D. Ragni, "Efficient prediction of urban air mobility noise in a vertiport environment," *Aerosp Sci Technol*, vol. 139, Aug. 2023.
- [128] Y. Fuerkaiti, D. Casalino, F. Avallone, and D. Ragni, "Aircraft community noise prediction in 3D environments using Gaussian beam tracing," in *28th AIAA/CEAS Aeroacoustics Conference, 2022*, American Institute of Aeronautics and Astronautics Inc, AIAA, 2022.
- [129] H. Lehnert, "Systematic Errors of the Ray-Tracing Algorithm," 1993.
- [130] Y. Song, Y. Zhang, C. Zhou, and Z. Zhao, "Acoustic ray tracing in the atmosphere: With gravitational effect and attenuation considered," *Annals of Geophysics*, vol. 57, no. 5, 2014.
- [131] G. B. Schulthess, and G. Schulthess, "Acoustic Waves in the Upper Atmosphere." [Online]. Available: <https://digitalcommons.usu.edu/gradreports>
- [132] L. Savioja and U. P. Svensson, "Overview of geometrical room acoustic modeling techniques," *J Acoust Soc Am*, vol. 138, no. 2, pp. 708–730, Aug. 2015.
- [133] M. B. Porter, "Beam tracing for two- and three-dimensional problems in ocean acoustics," *J Acoust Soc Am*, vol. 146, no. 3, pp. 2016–2029, Sep. 2019.
- [134] M. B. Porter, "Gaussian beam tracing for computing ocean acoustic fields," 1987.
- [135] L. Nijs and C. P. A. Wapenaar, "The influence of wind and temperature gradients on sound propagation, calculated with the two-way wave equation," *J Acoust Soc Am*, vol. 87, no. 5, pp. 1987–1998, May 1990.
- [136] R. H. Lyon, "Role of multiple reflections and reverberation in urban noise propagation," *J Acoust Soc Am*, vol. 55, no. 3, pp. 493–503, Mar. 1974.
- [137] V. Bulusu, V. Polishchuk, R. Sengupta, and L. Sedov, "Capacity Estimation for Low Altitude Airspace," Jun. 2017.
- [138] F. Škultéty, E. Bujna, M. Janovec, and B. Kandra, "Noise Impact Assessment of UAS Operation in Urbanised Areas: Field Measurements and a Simulation," *Drones*, vol. 7, no. 5, 2023.
- [139] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," *31st International Conference on Machine Learning, ICML 2014*, vol. 1, Jun. 2014.

## List of Figures

Figure 2.1 Obstacle map of the study area (Turin).....	22
Figure 2.2 Satellite photo of the study area (Turin).....	23
Figure 2.3 Population density map of the study area (Turin).....	24
Figure 2.4 Exponential fit of velocity-SPL for the noise source .....	25
Figure 2.5 The noise source acoustic rays model.....	26
Figure 2.6 A non-exhaustive taxonomy of algorithms in modern RL. Retrieved from < <a href="https://spinningup.openai.com">https://spinningup.openai.com</a> > .....	35
Figure 2.7 The interaction between agent and environment .....	37
Figure 2.8 Generic architecture for actor-critic agent (Nguyen, 2023).....	41
Figure 3.1 (a) Flight paths for test 1, (b) satellite photo of the test site .....	55
Figure 3.2 Heatmap of the SPL along the (a)RL and (b)A* flight paths (test 1) .....	57
Figure 3.3 Noise impact of the (a)RL, (b)A* and (c)Direct flight paths in the environment at 2D visualization (test 1) .....	58
Figure 3.4 SPL distribution along the (a)RL, (b)A* and (c)Direct flight paths (test 1)	59
Figure 3.5 Velocity distribution along the RL flight path (test 1).....	60
Figure 3.6 (a) Flight paths for test 2, (b) satellite photo of the test site .....	61
Figure 3.7 Heatmap of the SPL along the (a)RL and (b)A* trajectories (test 2) .....	62
Figure 3.8 Noise impact of the (a)RL, (b)A* and (c)Direct flight paths in the environment at 2D visualization (test 2) .....	63
Figure 3.9 SPL distribution along the (a)RL, (b)A* and (c)Direct flight paths (test 2)	64
Figure 3.10 Velocity distribution along the RL flight path (test 2).....	65

Figure 3.11 Box plot for (a)Path length, (b)SPL per unit length, (c)Minimum SPL and (d)Maximum SPL for the flight paths .....	67
Figure 3.12 (a) Flight paths for High and Low priority models, (b) satellite photo of the test site .....	69
Figure 3.13 Heatmap of the SPL along the (a)High priority and (b)Low priority flight paths .....	70
Figure 3.14 Noise impact of the (a)High priority and (b)Low priority flight paths in the environment at 2D visualization .....	71
Figure 3.15 SPL distribution along the (a)High priority and (b)Low priority flight paths .....	72
Figure 3.16 Velocity distribution along the (a)High priority and (b)Low priority flight paths .....	73

## List of Tables

Table 1 Key tuned hyperparameters associated with the algorithm .....	54
Table 2 The performance metrics for test 1.....	56
Table 3 The performance metrics for test 2.....	60
Table 4 Medians and standard deviations of the performance metrics .....	66
Table 5 The performance metrics for High and Low priority models.....	74
Table 6 The scaling factors of the reward function for High and Low priority models .....	74



