Received 3 October 2023, accepted 18 October 2023, date of publication 20 October 2023, date of current version 30 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3326435

RESEARCH ARTICLE

Dynamic Navigation in Unconstrained Environments Using Reinforcement Learning Algorithms

CHRISTOS CHRONIS, GEORGIOS ANAGNOSTOPOULOS, ELENA POLITI, GEORGE DIMITRAKOPOULOS, AND IRAKLIS VARLAMIS, (Member, IEEE)
Department of Informatics and Telematics, Harokopio University of Athens, 17779 Athens, Greece

Corresponding author: Christos Chronis (chronis@hua.gr)

This work was supported in part by the ECSEL Joint Undertaking (JU) under Grant 876019, and in part by the European Union's Horizon 2020 Research and Innovation Program and National Authorities.

ABSTRACT The potential for the use of drones in logistics and transportation is continuously growing, with multiple applications both in urban and rural environments. The safe navigation of drones in such environments is a major challenge that requires sophisticated algorithms and systems, that can quickly and efficiently estimate the situation, find the shortest path to the target, and detect and avoid obstacles. Traditional path planning algorithms are unable to handle the dynamic and uncertain nature of real environments, while traditional machine learning models are insufficient due to the constantly changing conditions that affect the drone location and the location of obstacles. Reinforcement learning (RL) algorithms have been widely used for autonomous navigation problems, however, computational complexity and energy demands of such methods can become a bottleneck to the execution of UAV flights. In this paper, we propose the use of a minimum set of sensors and reinforcement learning (RL) algorithms for the safe and efficient navigation of drones in urban and rural environments. Our approach considers the complex and dynamic nature of such environments by incorporating real-time data from low-cost onboard sensors. After performing a thorough review of the existing solutions in drone path planning and navigation in 3-D environments, we experimentally evaluate our proposed approach in a simulated environment, and in various scenarios. The test results demonstrate the effectiveness of the proposed RL-based approach in the navigation of drones in complex and unconstrained environments. The implemented approach can serve as a basis for the development of advanced and robust navigation systems for drones, which can improve safety and efficiency in transportation applications in the near future.

INDEX TERMS Actor-critic algorithm, drone navigation, dynamic path planning, proximal policy optimization, reinforcement learning.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have rapidly gained popularity in recent years due to their versatility and flexibility in various domains. With the growing demand for faster, cheaper, and more efficient delivery, drones have emerged as a potential solution [1]. In addition to logistics, drones have also proven to be useful in fields such as agriculture, surveillance, search and rescue, and more [2]. The ability of drones to access hard-to-reach areas and collect data from unique perspectives has made them an attractive tool for various applications. However, the challenge lies in navigating drones through different environments, particularly in urban and rural areas, which can be complex and dynamic. Particularly Beyond Visual Line Of Sight (BVLOS) operations have gained significant attention in recent years, for endowing flights with several degrees of intelligence, lower costs, and a reduced risk to human life [3]. The demand

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

for autonomous operations beyond the Visual Line of Sight (BVLOS) has further amplified the market share of UAVs, raising the need for faster and safer communication protocols, efficient scene perception models, obstacle avoidance and navigation algorithms, etc. The successful deployment and operation of BVLOS drones depend on their ability to adapt to the environmental conditions in which they operate. This is particularly important in urban and rural areas, where environmental conditions such as weather, traffic congestion, and terrain can vary greatly and impact the performance and safety of these systems [4]. In such environments, the ability of drones to adapt to changing conditions in real time is crucial for their effective and efficient operation. This requires intelligent algorithms that can learn from the environment and make optimal decisions based on the current conditions. Deep Reinforcement Learning algorithms have been proven very successful for learning complex behaviors from low-level observations, primarily with applications in games and recently with applications in real-world robot training [5]. Both model-based and model-free methods [6], [7] can be used to make an agent respond well in a specific task or gradually in a series of tasks of increasing complexity [8], [9] that enable BVLOS drones to adapt to their environment and navigate safely and efficiently in all areas.

Building a solution for reinforcement learning-based drone navigation requires a combination of several tools and technologies. One critical component is the use of sensors that provide situational awareness to the drone, allowing it to perceive and understand its environment [10]. Sensors such as GPS, LIDAR, proximity sensors, and cameras enable the drone to collect data on its surroundings, which can be used to make decisions on navigation and obstacle avoidance [11]. Additionally, fast and reliable communication is essential for remote control from the ground, enabling real-time monitoring and intervention in case of emergency. An efficient algorithm for autonomous navigation is also necessary to enable the drone to make decisions based on its perception of the environment and its current state. The algorithm should be able to learn from its experiences and adjust its behavior accordingly to ensure safe and efficient navigation. In this paper, we introduce a novel reinforcement learning algorithm for drone navigation in urban and rural environments. By leveraging sensors and reinforcement learning technologies, our proposed approach enables BVLOS drones to navigate autonomously in urban and rural environments while adapting to changing conditions, providing a more efficient and safer alternative for BVLOS drone operations.

In the field of drone navigation, different path-planning alternatives can be used depending on the specific requirements of the application [12]. For instance, a path can be planned based on the shortest distance between two points, or it can be optimized for energy consumption or time efficiency. However, regardless of the approach used, obstacles can occur in a drone's path, which may lead to collisions or other safety hazards. The detection and avoidance of obstacles are therefore critical to ensure the safe and efficient navigation of drones. Obstacles can take many forms, such as buildings, trees, power lines, or even other drones. Complex and dynamic environments, that combine static and moving obstacles can present significant challenges for space representation and path-planning algorithms [12]. The environment is constantly changing, with conditions and forces such as wind, turbulence, and weather affecting the drone's position and the behavior of obstacles. This composite nature of the environment makes traditional path-planning algorithms inadequate for drone navigation in urban areas.

The use of sensors can help detect and classify obstacles in the drone's environment, and predict the trajectory of moving objects. Additionally, efficient algorithms can be used to process the sensor data and plan a safe path around the obstacles [12], [13]. While traditional artificial intelligence (AI) algorithms have been used in 2-D or 3-D environments for obstacle avoidance and navigation, they can become ineffective in unconstrained, dynamic, and previously unknown environments. In this context, this work proposes an innovative approach, based on a reinforcement learning algorithm, that can adapt to dynamic conditions and learn to make optimal decisions based on the changing environment, enabling safe and efficient drone navigation in urban environments. It also uses input from a few low-cost sensors thus reducing the overall cost and complexity of the solution.

The proposed solution provides a holistic approach to UAV navigation in dynamic and unconstrained environments for BVLOS operations, where the position (i.e. latitude, longitude, and height), and speed of the vehicle are taken into consideration, along with the position of obstacles, which is not known in advance for each route.

The study uses a simulated world environment (using the AirSim framework) to validate the performance of the proposed solution with respect to energy efficiency and overall performance, and the results are promising. While further experiments in real-world environments are required, this work provides an essential foundation in the area of autonomous UAV navigation in dynamic and non-predefined environments.

The contributions of this work can be summarized in the following:

- it proposes a reinforcement learning-based approach for autonomous drone navigation, which can be adapted to both urban and rural environments, with static and dynamic obstacle detection, specifically focusing on beyond visual line of sight (BVLOS) and autonomous drone operations,
- it evaluates several reinforcement learning algorithms and compares them with algorithms from various families that have been used for the task in the past and reveals the superiority of the most recent RL approaches,
- it allows obstacle detection and avoidance using a limited number of sensors, using the minimum information

that is needed to properly navigate the drone. This differentiates the current approach from existing methods that employ LiDARs or cameras, which increase the drone equipment cost and weight, and inflate the computational requirements.

The proposed approach enables drones to adapt to changing environmental conditions and make optimal decisions for safe and efficient navigation with respect to energy efficiency. It innovatively manages a limited number of sensors and the information they provide in order to gain a perception of the surrounding environment. The experiments that we performed in the AirSim simulation environment, demonstrate that without relying on heavy and expensive sensors like LiDARs or computationally intensive sensors like cameras, especially considering the preprocessing and use of multilayer convolutional layers, we can achieve commendable results. Using input from a few simple sensors also prevents the model from overfitting to a particular environment, and enhances its generalization capabilities.

Section II that follows provides a comprehensive review of the existing literature on drone navigation and path planning techniques. Various types of algorithms from different categories are surveyed, including sampling-based, bio-inspired, mathematical-based, AI, and ML-based approaches. Section III introduces the proposed approach, which is based on the Proximal Policy Optimization (PPO) reinforcement learning algorithm, and presents the formulation of the drone navigation task. In Section IV, the experimental setup used for comparing the proposed approach against well-known methods from each category is explained in detail. Section V presents and discusses the results obtained from this work, including a thorough analysis of the performance of the proposed approach in comparison to other methods. Finally, in Section VI, the paper concludes with a summary of the findings and potential future research directions in the field of drone navigation and path planning algorithms.

II. RELATED WORK

Drone navigation is a challenging task that involves two primary sub-tasks: path planning and obstacle detection and avoidance. In unknown environments, the planning of the whole path is not always possible in advance and thus the task has to be broken down into sub-tasks that are optimally solved [14]. Simultaneous localization and mapping (SLAM) based solutions are ideal for unconstrained environments where changes in the positions of landmarks in the surroundings are common. These methods obtain input from sensors like IMU and LiDAR, however, they tend to increase the computational burden [15]. In the case of dynamic environments, the path plan must be continuously updated based on obstacles that are being detected during flight and appear in the drone's path. Both sub-tasks are critical for safe and efficient drone navigation in dynamic and complex environments. Over the years, various approaches have been proposed, ranging from traditional artificial intelligence algorithms to more recent bio-inspired methods and deep learning and reinforcement learning techniques. In this section, we review the related work in drone navigation with emphasis on path planning and obstacle detection and avoidance [16]. Having broken down our navigation problem into smaller path-planning tasks, a few noteworthy algorithms and models that have been used will be discussed in the following, organized by their respective fields. The techniques are divided into five major categories: Sampling-Based, Artificial Intelligence, Mathematical Models, Bio-inspired Models, and Machine Learning Models [17]. In recent years, there has been a surge of studies that employ hybrid methods, methods that combine approaches from the five major categories and achieve promising results. These methods focus not only on UAVs but also on other types of unmanned vehicles. In the following subsections we discuss the main approaches in each category.

A. SAMPLING-BASED TECHNIQUES

The sampling-based techniques are widely used in path planning. They require prior knowledge of the 3-D environment, which is either divided into cells (nodes) and annotated with the location of obstacles that constitute the workspace configuration or is represented as a graph with obstacle-free areas as nodes and obstacle-free lines connecting them as edges. In both cases, the objective is to connect the start to the end node (constructing roadmaps) using optimal path-planning algorithms.

A method for constructing roadmaps is the **Probabilistic Roadmaps**, which are useful for collision avoidance and for identifying communication gap failures in UAVs [18], [19]. **Rapid-exploring Random Trees** (RRT) is another popular sampling-based technique that is commonly used as a single motion planner. It is used to navigate in high-dimensional spaces by randomly building a space-filling tree. RRT-Connect, an extension of the basic RRT algorithm, is a more greedy variant that is useful for collision avoidance between UAVs [20], [21].

The **A-star** algorithm is a widely used technique in drone path planning. It is used to compute the optimal path based on the cost of traversing the edges of the graph, or on the singular cost of moving from a cell to a nearby free cell. This algorithm is commonly used on a map grid to find an optimal and shortest path [22]. **Voronoi diagrams** [23], **A-star** and **Dijkstra's algorithm** [24] are also frequently employed for finding the shortest path in such representations.

All the above-mentioned methods assume a static representation of the world and thus increase in complexity when they have to be applied in dynamic environments. The **Potential Field Method** (PFM) is a simple and lightweight method for dynamic path planning. It represents the environment as a particle moving under the control of potential fields around the c-space. The shortest path is consequently calculated based on the resultant fields from the initial point to the target point. However, conventional PFM suffers from local minima, which can cause the moving object to stuck before reaching the target. Overall, sampling-based techniques have shown promising results in path planning and obstacle avoidance for UAVs [25].

B. ARTIFICIAL INTELLIGENCE TECHNIQUES

Artificial intelligence techniques are frequently adapted to solve path-planning tasks. Exhaustive or **brute-force search algorithms** can be easily applied in UAV path planning tasks. Breadth-first and depth-first search space exploration approaches are exhaustive and always find a path if it is available, or the shortest of all available paths, but they can be quite slow. They can also be combined with backtracking to avoid dead-ends [26]. Greedy methods can also be employed to accelerate search but are always at the risk of stacking into local minima.

To tackle this problem, **local search algorithms**, such as hill climbing, simulated annealing, and local beam search have been employed to solve hard path-planning problems with complicated constraints, like the Vehicle Routing Problem (VRP) or the Travelling Salesman Problem (TSP) in spaces with many targets that must be visited and many obstacles. Since the application of brute-force search is infeasible when the number of targets exceeds 50, such methods start from one or more quick solutions and then apply local changes, and random restarts if needed, in order to find the optimal solution [27].

Instead of using and examining one solution at a time, population-based solutions have been proposed in the literature to speed up the process of optimal solution search. The parallel processing and comparison of generated paths enhances the algorithm's ability to explore the search space. **Genetic algorithms** have been based on operations like mutation, crossover, and selection in order to generate and keep the best route from a population of solutions, which either minimizes flight time from start to destination [28] or maximizes the terrain coverage in land monitoring scenarios [29], [30].

C. BIO-INSPIRED MODELS

Bio-inspired path planning algorithms for UAVs [31], drawing inspiration from biological evolution, offer potential solutions for addressing challenges in UAV path planning, despite nonlinear constraints, time restrictions, and high dimensionality. They are known for their robustness, adaptability, and ability to avoid local optima, making them popular for UAV path planning due to their self-organized patterns, ability to handle adverse conditions, and flexibility in adapting to changing environments. **Particle Swarm Optimization** (PSO) is the most popular bio-inspired method for path planning [32] with various implementations [33] that include, weighted PSO [34], distributed PSO [35], etc. **Ant Colony Optimization** (ACO) is another swarm intelligence model that has been employed for UAV path planning [36], [37], [38] and in some cases has been combined with PSO to find optimal path planning solutions [39]. **Grey wolf optimization** (GWO) is another bio-inspired model that has been used for UAV path planning [40], [41] and collision avoidance [42]. All the above-mentioned methods tend to explore the search space efficiently by exploiting local information, while also maintaining the ability to escape local optima and explore diverse regions of the search space. They use a population of agents that interact or evolve in parallel. This allows them to explore multiple solutions concurrently and increases their robustness against local optima. Their main disadvantage is that they have computational cost and memory requirements and due to the multitude of particles they also require efficient communication and coordination. They rely on heuristics or probabilistic rules, which do not always guarantee optimal solutions and may also lead to slow convergence.

D. MATHEMATICAL MODELS

All the approaches presented so far assume an apriori knowledge of the flying space and all the obstacles within it. In the case of completely unknown environments, it is important to break the path-planning problem into smaller segments, defined by sequences of waypoints, and safely fly the UAV from one waypoint to the other. In this direction, mathematical models that emerge for the Control Theory, such as the **Bezier curves** can be used to compose a navigation path out of smooth curves that connect the consecutive way-points [43], [44]. As the drone safely moves to the next free waypoint, information from the sensors reveals the drone's surroundings and helps find the next waypoints using sampling-based, AI, or bio-inspired algorithms. **Markov decision process** (MDP) is another mathematical model for decision-making in uncertain environments, which has been employed for real-time path planning in dynamic environments [45], [46]. Once again based on control theory, MDP needs a formulation of the action space, the allowed transitions, and the state-transition law as well as the objective function, in order to find the optimal action for each state. **Integer and Mixed Integer Linear Programming** have also been employed for solving UAV optimization problems, from energy consumption minimization to obstacle avoidance with minimum deviations [47], [48]. **Dynamic programming** approaches have also been extensively used in robotics for path planning [49] and optimization of robotic trajectories [50] but have also been used for UAV path planning under adverse weather conditions (e.g. wind [51], [52]) that affect the UAV position. **Statistical modeling** has also been employed in the past for path planning since it allows learning the motion or movement primitives from the distributions detected in the training data [53]. They offer a way to imitate the optimal path given by one or more humans, and in some cases can be used to learn how to personalize a generic path planning model [54].

Among the advantages of mathematical models is their ability to systematically search for the best solution based on defined objectives and constraints. They are very flexible and can be adapted to handle different scenarios, environments, and mission objectives. They can also provide interpretable solutions, which makes it easier to understand and analyze. However, they are based on certain assumptions and simplifications about the problem domain, which may not always accurately capture the real-world complexities, leading to sub-optimal or unrealistic solutions. The uncertainties that hold in real-world system dynamics, sensor measurements, or environmental conditions may also affect the performance of such models in real conditions.

E. MACHINE LEARNING MODELS

Machine learning is a part of AI that enables computers to respond without being explicitly programmed. The machine learning-based algorithms are divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. The recent progress in complex neural network architectures made them very popular in various learning setups and gave rise to a fourth deep learning category. A wide range of algorithms and models from all three categories have been used for UAV path planning [55].

Supervised learning [56], is used to build models where dependencies exist between the input elements and predicted target value. The nonlinear least-square method has been proposed in [57] for UAV dynamic path planning tasks in environments with obstacles, no-go areas, and other moving threats (e.g. attacking missiles, or other drones). The extra challenge of the task was that the targets were moving and the UAV had to follow them from a safe distance, whilst avoiding collision with the moving threats. In order to solve any issues of noise in the detection of drone, target, and obstacle positions, authors in [58] have employed Kalman filters. Similarly, authors in [59] used Kalman filters to solve various problems such as noise in the air, collision probability, cluster state prediction, and track planning of UAVs.

Artificial neural networks (ANNs) have also been used to learn and regulate the line-of-sight path control of UAVs so that UAVs can act precisely in adverse conditions [60]. They have also been used for finding optimal UAV paths and controlling the drone kinematics to avoid collisions [61]. More recently, attention-based models and Graph Neural Networks have been employed for multi-agent navigation in unconstrained environments [62]. Building on the complexity of **deep neural networks** and their ability to solve complex decision tasks, authors in [63] propose an end-to-end UAV path planning solution that takes as input multimodal information (i.e. visual depth and UAV state information) from the UAV's onboard sensors and learns to output collision-free trajectories. **Unsupervised learning** methods such as Quality Threshold Clustering (QT clustering), DBScan, k-means etc., have also been used for detecting obstacles and grouping them into no-fly zones [64], [65]. The input to such algorithms is either a depth image, a point cloud, or a set of edges which are consequently clustered to capture the potential obstacles and their shapes [66].

Deep Reinforcement learning is widely used to solve the path-planning techniques of UAVs. It is a continuous process and always learning from the environment in an iterative manner. In the literature, authors mainly used reinforcement learning, deep reinforcement learning, and deep Q-network for the path planning of UAVs [67], [68], [69]. Reinforcement learning has been used for high-accuracy tracking of hydraulic systems in recent years [70]. The G-learning method has been proposed in [71] to solve the problems of path planning in 3-D-based UAV path planning scenarios. This algorithm is used to compute the cost matrix based on a geometric distance for UAV path planning. Then, the updated information is broadcasted to the other UAVs for collision avoidance.

The Q-learning algorithm has been used in [72] for UAV path planning, allowing the drone to interact with the environment without any previous knowledge or training samples. The authors have used an adaptive and random exploration technique for UAV navigation and collision avoidance. Finally, the Advantage Actor-Critic (A2C) algorithm has been proposed in [73] for safely navigating a UAV through a plane full of obstacles using only four distance sensors to detect obstacles and minimum control (speed and z-rotation) to avoid them. In urban UAV path planning, [74] proposed a DDQN network, assisted by IoT devices, in order to optimize paths while considering flying time and sensor metrics. For obstacle avoidance, especially in 3D urban contexts, the Deep Deterministic Policy Gradient (DDPG) is used to differentiate between moving and static objects [75]. For Indoor navigation the [76] uses a Deep Q-network trained on various image datasets. Both [76] and [77] focused into enhancing indoor navigation, with integrating memory into UAV decision-making and combining a cGAN with a deep recurrent Q-network having temporal attention. Further, [78] employs a reinforcement learning model centered on the UAV's current state and learning paradigm. For pathfinding challenges, such as obstacle avoidance and efficient trajectory determination the [77] utilize the Q-learning algorithm. The [79] utilize interference-aware strategies for path planning harness a deep reinforcement learning method with ESN cells to ensure UAV paths experience minimal disruptions. In scenarios with potential hostilities, Dueling Double Deep Q-Networks (D3QN) are preferred for real-time path planning, especially when considering threats like enemy radars [80], [81].

DRL and Imitation Learning, often referred to as Demonstration Cloning, also have been used for UAV navigation. In this approach, a human expert demonstrates the task. Utilizing the collected experience, the DRL algorithm then aims to imitate the human expert and even surpass their performance. He et al. [82] used a DRL approach for UAV navigation in uncharted terrains, combining imitation learning with the TD3 algorithm. Their technique leverages expert demonstrations to improve both policy and Q-value networks. Additionally, it uses the TD-error with a gradually reducing imitation loss, showcasing its efficacy in complex

3D UAV navigation scenarios. Fu et al. [83] addressed the exploration limitation problems in DDPG (Deep Deterministic Policy Gradient) when applied to UAV pursuit-evasion strategies. By integrating imitation learning, they created a guidance control law geared towards generating effective learning samples. The outcome was the introduction of the IL-DDPG strategy, which amplifies exploration efficiency while concurrently reducing unproductive explorations. Alagha et al. [84] unveiled two multi-agent DRL models, specifically designed for complex environments. One of these models utilizes the Expert Demonstrations, employing the Demonstration Cloning technique. Both models have demonstrated their effectiveness in radioactive target localization. Lastly, Wang et al. [85] introduced a novel two-stage DRL decision-making framework aimed at multi-UAV target tracking. At the core of their approach is a sample generator that produces expert experience data. The generated data is then used to pre-train both the policy and critic networks using behavior cloning loss and Q-value loss. The outcome is a big boost in exploration and accelerated learning.

Machine learning and deep learning methods offer adaptability to changing environments, and automation of the path-planning process for UAVs. They can learn from patterns, and make informed decisions in real time, resulting in optimized flight paths considering factors such as terrain, weather, and obstacles. With the use of reinforcement learning methods, it is possible for UAVs to dynamically adjust their flight paths based on real-time information, allowing them to respond to changing situations and mission objectives. However, there are also challenges associated with the use of machine learning methods for UAV path planning. These include the requirement for substantial amounts of data for training, the complexity and interpretability of the algorithms, the computational requirements that may exceed the onboard computing capabilities of UAVs, and the potential issues with overfitting and generalization. Careful consideration and validation are necessary to address these challenges and ensure the effective and safe use of such methods in UAV path planning applications.

F. HYBRID METHODS

A large number of hybrid approaches combine algorithms from the above groups to tackle the problems of path planning and obstacle avoidance for various types of vehicles. Focusing on UAVs, Richards et al. [86] proposed a hybrid approach that divides the planning process into offline and online phases. Their approach combines A* for path planning in a discrete space and motion primitives for automating trims and maneuvers in order to efficientlty avoid obstacles. Yao et al. [87] introduced a 3D real-time path planning technique specifically designed for UAVs, integrating two mathematical models, namely the enhanced Lyapunov Guidance Vector Field with the Interfered Fluid Dynamical System. They later expanded their work to address the challenges of three-dimensional cooperative path planning for multiple UAVs [88]. Zhang et al. [89] presented a technique that combines Improved Particle Swarm Optimization with an Artificial Potential Field, focusing on efficient multi-robot formations in environments with unknown obstacles. Yafei et al. [90] developed a UAV penetration path planning algorithm that merges the artificial potential field with the RRT algorithm, aiming to distinguish between threats and obstacles. Sangiovanni et al. [91] proposed a hybrid dual-mode architecture using a sample-based algorithm (SBL[1]) for motion planning and Deep Reinforcement Learning for obstacle avoidance in order to eliminate the complexities of obstacle consideration in robot path planning. Most recently, Kiani et al. [92] introduced three novel variants of the Rapidly Exploring Random Tree (RRT) algorithm for 3D path planning in autonomous robots. These hybrid methods, named Adapted-RRT-GWO, Adapted-RRT-I-GWO, and Adapted-RRTEx-GWO, incorporate Grey Wolf Optimization (GWO) metaheuristics to overcome the limitations of both sampling-based algorithms. Beyond UAVs, some studies have extended their focus to Unmanned Surface Vehicles (USVs) and maritime transportation. Chen et al. [93] introduced a hybrid approach that combines an ant colony optimization algorithm with artificial potential fields. Abebe et al. [94] concentrated on maritime transportation, introducing a hybrid ARIMA-LSTM model for accurate ship trajectory estimation. Finally, Shin et al. [95] proposed a novel path planning method centered on positioning accuracy for Unmanned Vehicles, ensuring that paths maintain their positioning capability.

All aforementioned methods represent the state of the art in UAVs autonomous navigation. However, when classic AI is applied to such problems, computational complexity and energy demands may compromise the efficiency and autonomy of the BVLOS flights. This work proposes an innovative reinforcement learning approach for efficient autonomous navigation in dynamic environments which offers a holistic approach to UAV path planning problems.

III. THE PROPOSED APPROACH

Problem Definition: As depicted in Figure 1, we assume that a UAV D wants to move from the starting point A to the destination point B, without colliding in any of the obstacles that may occur in its route and without crashing into the floor. The flight area is not known in advance and there is no prior information about the location or size of the obstacles. The navigation algorithm has to use the input from a limited set of distance sensors in order to plan the path for the drone to follow.

The proposed approach aims to create a system that takes advantage of the various onboard sensors to solve the path planning problem and perform collision avoidance tasks. In order to achieve this, the system must be able to detect obstacles and perform avoidance maneuvers in real-time, while heading to its destination. The conjunction of real-time

[1]Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking.

TABLE 1. Comparison of path planning techniques for UAVs.

Techniques	Methods	Advantages	Disadvantages
Sampling-Based	Probabilistic Roadmaps, Rapid-exploring Random Trees, A-star, Voronoi diagrams, Dijkstra's algorithm, Potential Field Method	Wide usage, promising results in path planning and obstacle avoidance, ability to navigate in high-dimensional spaces	Increase in complexity in dynamic environments
Artificial Intelligence	Brute-force search, Local search algorithms, Genetic algorithms	Adaptable, ability to solve optimization problems, exhaustive search	Risk of local minima, computationally expensive
Bio-inspired	Particle Swarm Optimization, Ant Colony Optimization, Grey Wolf Optimization	Balance between global exploration and local exploitation, ability to avoid local optima	High computational and memory requirements, reliance on heuristics
Mathematical	Bezier curves, Markov decision process, Integer, and Mixed Integer Linear Programming, Dynamic Programming	Systematic search, flexibility, interpretable solutions	May not capture real-world complexities, performance affected by uncertainties
Machine Learning	Supervised Learning, Artificial Neural Networks, Deep Neural Networks, Unsupervised Learning, Reinforcement Learning	Adaptability, automation of path-planning, ability to learn from patterns	Need for extensive data, computational requirements, risk of overfitting

sensor data processing with a path planning algorithm allows interpreting sensor data to identify potential obstacles and then make decisions and navigate around them.

The Sensor Setup: The UAV is equipped with four distance sensors, an IMU sensor, and a GPS that provides its location at any time. The first three distance sensors are placed at the front of the drone and the fourth at the belly. The measurement capabilities of the sensors are up to 40 meters. From the sensors at the front, one sensor is placed directly in the center, and the other two at +45 and −45 degrees angle. This arrangement allows the drone to detect obstacles not just directly in front, but also at the edges. The fourth sensor is located at the bottom of the drone, enabling it to detect obstacles below the vehicle and the ground itself. Our system assesses the distance to an obstacle using these three front-facing sensors. If the drone approaches an object and all three sensors return consistent distance readings, it indicates that the obstacle is continuous. If, however, one of the sensors does not detect the obstacle, there is likely a gap that the drone can navigate through. The strategic placement of the left and right sensors not only gives insight into the continuity of an obstacle but also aids the drone in deciding whether to move towards a potential opening or to steer clear of tight corners. All the data from the sensors feeds into our model. From there, the model, throughout its training, formulates a strategy. The exact workings and decision-making processes of the model are, by the nature of neural networks, a black-box approach.

We assume that at every moment of the flight, the UAV knows its position and the destination position, and uses the sensors as a way to detect surrounding obstacles. The UAV has certain movement options that can be limited to moving forward (accelerating and decelerating), moving upward or downward (for take-off, landing, and altitude change), and rotating to the z-axis (clockwise and counter-clockwise). We also assume that in each flight the drone begins without any knowledge of the position of the obstacles, and does not keep any information about the obstacles' position until it reaches its destination, or in consecutive flights. In order to remove the potential bias of the UAV learning a specific route in this area, we make sure that the start and destination points are randomly selected for each flight, thus asking the UAV to learn to move in all directions in order to avoid obstacles and reach its goal. Figure 1 provides a bird's eye view of a UAV's path among obstacles.

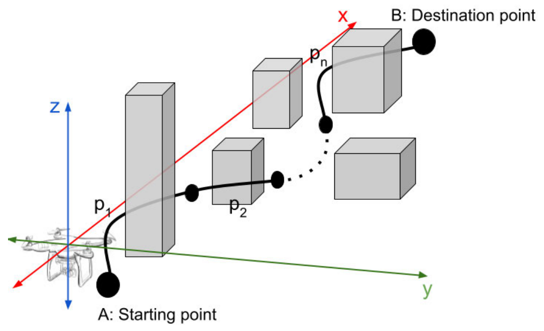


FIGURE 1. An abstract view of the path planning approach.

A. FORMULATING THE REINFORCEMENT LEARNING TASK

In order to apply any reinforcement learning (RL) algorithm to the drone navigation task, it is necessary to follow the general definition of the algorithm and translate it to the specific task. This involves identifying the key components of the RL solution, namely the Observation space, the Action space, and the Reward function. By formulating the problem in this way, it becomes possible to apply any RL algorithm to the task and use it to effectively navigate the drone.

The **Observation space** pertains to a structured depiction of the environment and the drone within it, encompassing the diverse variables that determine the current state of the drone. This state represents a snapshot of these variables at a particular moment. It is essential that the state encompasses the maximum necessary information about the environment that is needed to support the task at hand.

The **Action space**, which comprises the complete range of actions that an agent can take at any given time. As the drone has to move in all three dimensions, six distinctive actions have been established. The actions are divided into three groups, each consisting of two actions, an increasing, and a decreasing action. The first group is responsible for controlling the forward velocity of the drone in the direction it is heading. The second group is responsible for controlling the drone's altitude by adjusting the speed of its four motors. Finally, the last group is responsible for controlling the drone's yaw movement (rotation around the Z-axis). With this group, the drone can change its heading using predefined steps of one degree. The final movement of the drone in a direction comes from the combination of the drone's forward velocity and heading angle, those values are converted internally to a combination of velocities for X and Y axes.

The **Reward function** is often one of the most challenging parts of any RL application, as it serves the purpose of facilitating the agent in acquiring the task policy through learning. The reward function utilizes information from the observation space every moment after the agent executes an action. The primary goal of the agent is to determine the optimal action at each moment by assessing the state of the UAV. The training procedure evaluates each action subsequent to its execution and calculates the reward that the agent deserves based on a reward function and based on that the weights update. In the scenario of the UAV, the reward function compels the drone to diminish the distance between its current position and the intended destination, while concurrently ensuring that no collisions transpire with any obstacles or the terrain.

In our experiments, we used a reward function that takes into account the drone's distance from the target position and its orientation. Specifically, we define d_{target} as the Euclidean distance between the drone and the target position, and d_{max} is a maximum distance (e.g. the diagonal of the flight area), which is used to normalize d_{target}. We also calculate the heading offset of the drone, $\Delta\theta$, between its current heading and the heading required to reach the target position. Using $\Delta\theta$ and d_{target}, we calculate the length of the arc that the drone should rotate by, denoted as l_{arc}.

$$l_{arc} = d_{target} \cdot \Delta\theta \tag{1}$$

We then calculate the weighted sum of d_{target} and l_{arc}, which serves as the target distance that the drone needs to minimize (see Figure 2).

To further adjust the importance of each distance, we use the weights w_{arc} and w_{eucl}, which are assigned to the arc length and Euclidean distance, respectively.

$$distance = w_{arc} \cdot \frac{l_{arc}}{\pi d_{target}} + w_{eucl} \cdot \frac{d_{target}}{d_{max}} \tag{2}$$

Finally, we calculate the step reward, denoted as r_{obs}, which subtracts the target distance from 1 and multiplies the result by the weights w_{vel} and w_{obst}.

$$r_{obs} = (1 - distance) \cdot w_{vel} \cdot w_{obst} \tag{3}$$

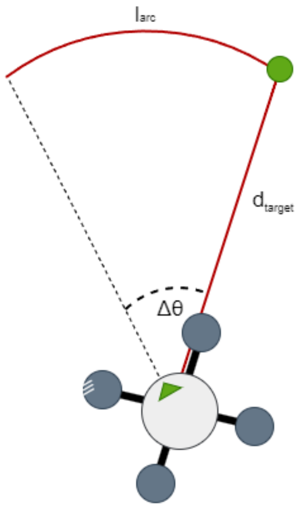


FIGURE 2. An overview of how the reward is calculated.

w_{vel} is used to penalize the drone for not moving forward, where the forward velocity of the drone is denoted as v_f.

$$w_{vel} = 1 - e^{-\alpha_v v_f} \quad (0 \text{ for } v_f < v_{min}) \tag{4}$$

w_{obst} is used to encourage the drone to avoid getting too close to obstacles, which is calculated using d_{obst} which is the distance to the nearest obstacle at each moment.

$$w_{obst} = 1 - e^{-\alpha_o d_{obst}} \quad (0 \text{ for } d_{obst} < d_{min}) \tag{5}$$

In Equations 4 and 5, α_v and α_o are scaling factors for the velocity and obstacle distance weights, respectively. In addition, v_{min} and d_{min} represent the minimum values for the velocity and obstacle distance thresholds. Then, the reward obtained by the drone in each time step is given by:

$$R = \begin{cases} r_{coll\_floor} & \text{if the drone collides with the floor} \\ r_{coll\_obst} & \text{if the drone collides with an obstacle} \\ r_{timeout} & \text{if flight over a given time} \\ r_{overshoot} & \text{if drone exceed the given boundaries} \\ r_{goal} & \text{if target reached within a radius of } r \\ r_{obs} & \text{otherwise} \end{cases} \tag{6}$$

where r_{coll_floor}, r_{coll_obst}, $r_{timeout}$, $r_{overshoot}$, r_{goal} and r_{obs} are the respective reward values for collision with the floor, collision with an obstacle, timeout, flight out of boundaries, reaching the goal position and calculated reward based on the state of the observation space. The r_{coll_floor}, r_{coll_obst}, $r_{timeout}$ and $r_{overshoot}$ are penalties (negative rewards) with fixed values and describe wrong decisions of the agent. The r_{goal} and r_{obs} are positive rewards with the r_{goal} to have a very high fixed value and the r_{obs} to take positive values based on the current state.

B. PROXIMAL POLICY OPTIMIZATION (PPO)

Based on the general formulation of the path planning and obstacle avoidance task with reinforcement learning terms, it is important to choose the appropriate RL algorithm and fine-tune it to the task at hand. For this purpose, we chose the **Proximal Policy Optimization (PPO)** algorithm [96], which offers several benefits that make it an ideal choice for real-time path planning in unknown and unconstrained environments.

PPO is particularly useful in UAV maneuvering applications, where real-time control is critical and a model of the environment may be difficult to obtain. PPO does not require a model of the environment and updates the policy directly. It is an on-policy algorithm, meaning that it uses the most recent data to update the policy. Also has internal mechanisms that allow it to make use of all the data it has collected, which can lead to more efficient learning. PPO works by updating its policy using samples collected from the environment and using a surrogate objective function that approximates the true objective of maximizing the expected reward. PPO can learn from a limited number of interactions with the environment, which is particularly beneficial for UAV path planning and obstacle avoidance tasks, where real-world interactions can be expensive, time-consuming, or even dangerous.

PPO is designed to be stable and efficient, and it achieves this by using two key techniques: **clipping** and **value function fitting**. The clipping technique is used to prevent policy updates from being too aggressive, which can destabilize the learning process. Stable optimization is important in tasks where safety is a critical concern, and sudden changes in the policy can lead to collisions or other undesired behaviors. The usage of the value function based on the bibliography offers an efficient policy learning way and the same time is more suitable for continuous action spaces as in the current problem [96], [97]. In our implementation, the value function fitting technique, on the other hand, involves updating the value function separately from the policy and using it to estimate the advantage of a state-action pair. This advantage estimate is used in the surrogate objective function, which helps to improve the stability of the algorithm.

Value can be used in online learning settings, where the agent learns and updates its policy on-the-fly as it interacts with the environment [97]. This makes it suitable for real-time decision-making tasks, where the environment is dynamic and changes over time. The ability to learn online allows the UAV to adapt to changing conditions and make informed decisions in real time. Finally, value has been shown to be relatively robust to hyperparameter settings, making it less sensitive to tuning compared to some other reinforcement learning algorithms. This is advantageous when the complexity and dynamics of the environment are high.

Figure 3 provides an overview of the PPO algorithm, showing how it adapts to the reinforcement learning process.

The clipped surrogate objective function $L^{clip}(\theta)$ that is maximized during training to improve the policy is defined

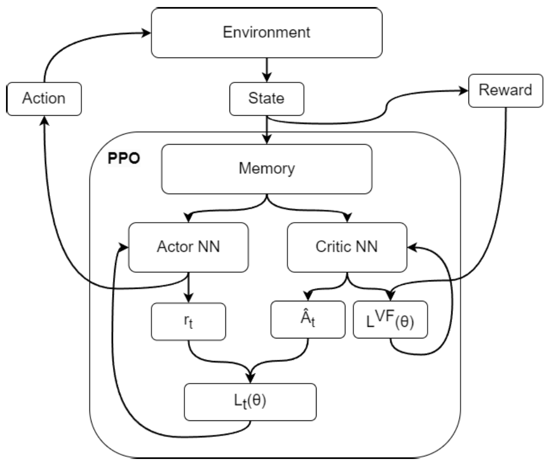


FIGURE 3. PPO diagram.

as the minimum of two terms, as depicted in Equation 7.

$$L^{clip}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \quad (7)$$

The first term is the product of the probability ratio $r_t(\theta)$ and the advantage estimate $\hat{A}_t$. The second term is the clipped probability ratio (which is constrained in a range of $1-\epsilon$ to $1+\epsilon$) multiplied by the advantage estimate, which prevents the new policy from deviating too much from the old policy.

The advantage estimate is calculated for each time step t as the difference between the discounted (by a discount factor γ) sum of rewards from time step t and the estimated value function $V(s_t)$ at time step t. It is given by the following equation:

$$\hat{A}_t = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (8)$$

The value function estimate $V(s_{t+1})$ is also included in the equation to account for future rewards.

The value loss function $L^{VF}(\theta)$ that is minimized during training measures the difference between the estimated value function $V(s_t)$ and the empirical estimate of the value function V_t for each time step t and is defined as follows:

$$L^{VF}(\theta) = \hat{\mathbb{E}}_t[(V(s_t) - V_t)^2] \quad (9)$$

The hat symbol over the expected value (i.e. $\hat{\mathbb{E}}$) denotes an estimate of the expected value.

Finally, the clipped surrogate objective loss for PPO is as follows:

$$L_t(\theta) = \hat{\mathbb{E}}_t[L_t^{clip}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \quad (10)$$

In all equations, θ represents the parameters of the policy network, r_t represents the probability ratio between the new and old policies, V represents the value function estimated by the value network, s_t represents the state at time step t, V_t represents the empirical estimate of the value function at time step t, c_1 and c_2 are coefficients, the S denotes an entropy bonus, π_θ is a stochastic policy and ϵ is a hyperparameter that controls the magnitude of the clipping.

IV. EXPERIMENTAL EVALUATION

Before selecting the algorithms to be used for the UAV path planning in the simulated 3D environment, we choose to evaluate a wider range of algorithms from the families of algorithms presented in Section II, on a 2D setup with static obstacles with positions that are known in advance. For this purpose, we choose at least one algorithm from each family.

A. PRELIMINARY EXPERIMENTS

In order to test the performance of different path-planning algorithms, we evaluated several implementations of selected algorithms that we reviewed in the literature, taking care to choose at least one algorithm per group and a larger number of RL algorithms. More specifically we chose: A*, Potential Field Method (PFM), Randomized Rapidly-Exploring Random Trees (RRT), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Dynamic Programming (DP), Deep Q-Network (DQN), Quantile Regression Deep Q-Network (QRDQN), Advantage Actor-Critic (A2C), Trust Region Policy Optimization (TRPO) and the Proximal Policy Optimization (PPO) algorithm.

For the first set of experiments, we constructed a 46×40 grid-map that resembles the overview of an area with buildings of various sizes and two obstacle-free areas at the bottom and the top. With the use of OpenCV the map has been converted to binary image where obstacles are represented as ones and free spaces as zeros. The only available actions for the agent were to move forward and rotate around itself (clockwise or anti-clockwise by 45 degrees at each step). Additionally, obstacle detection was possible only in the direction the agent was facing. For the experiments, we used the same reward function of Eq. 6 for each step, ignoring any information about altitude (including floor collisions which can't happen in the 2-D scenario). An episode begins at the starting position and ends when a crash occurs when the UAV gets out of map bounds, or when it arrives at the end point. The episode reward is the cumulative sum of the step rewards along the route.

For the reinforcement learning algorithms, that need to be trained beforehand, we ran several episodes until the model learned to consistently find a path to the target. In each training episode, we used random spawn points in the free area at the bottom of the map, as well as random end points in the free area at the top of the map. The neural network model used by the actor had an input layer, followed by 3 dense layers (with 64, 32, 12 neurons using Relu) and an additional dense output layer with 3 neurons using the tanh function. The critic employed a similar network but with a single neuron using the linear function in the output layer.

To check if the model has been trained, every 1000 time steps,[2] we performed an evaluation cycle consisting of 10 episodes. We assume that the model has been trained when the UAV has reached the target in all evaluation episodes

[2]Given the grid-map dimensions, the average duration of a successful episode is 60 time steps.

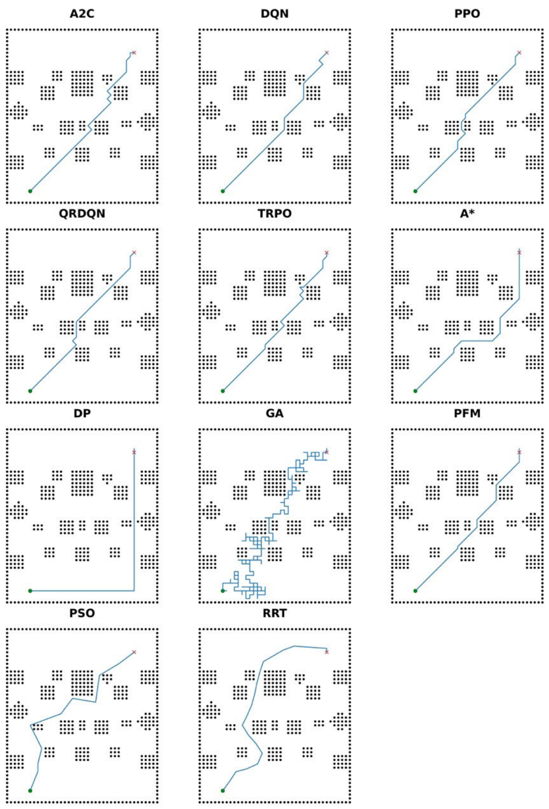


FIGURE 4. Selected paths of algorithms.

for 10 consecutive cycles. For the evaluation of the trained models, we generated a test set of 1000 pairs of random starting and goal points (in the free areas as before) and tested all the algorithms on the same set and the same task which was to successfully find a path from the starting to the end point without colliding to the obstacles. In all cases, the map environment was unknown at the beginning of each route.

An examination of the results of the 2-D path planning and obstacle avoidance task as shown in Table 2 demonstrates the high computational complexity of the bio-inspired approaches, since for PSO the mean time per route for reaching the end point is much higher than all other algorithms although the mean path length is at the scale of the best algorithms. It also shows that the mathematical models (i.e. DP) fail to find the path to the end-point in a reasonable amount of time in 60% of the cases (although they are fast in the cases they manage to find a path. Similarly, the Genetic Algorithms and the Sampling-based algorithms (apart from A*) are either slow or result in very long paths. The RL-based (Reinforcement Learning) approaches combine very low computation times and find paths of reasonable mean length. However, they do not always find a path to the target. More specifically, the best-performing algorithms were PPO and TRPO, which had the fastest path generation time and the shortest path, followed by DQN and A2C. Among the methods from the other categories, A* offered the best combination of low calculation time and acceptable path length. Based on their performance in the

TABLE 2. Performance comparison of algorithms. With gray shade are the algorithms that had very low performance in one of the metrics. * The distance and flight time are averaged over the successful routes only.

Family	Algorithms	Successful Routes	Average Distance (in steps)*	Average Flight Time (in sec)*
Sampling-based	A*	1000	47.9 (± 0.77)	0.0 (± 0.0)
	PFM	1000	31.38 (± 2.34)	6.13 (± 0.01)
	RRT	1000	114.25 (± 4.77)	0.08 (± 0.01)
Artificial Intelligence	GA	1000	523.12 (± 23.03)	12.97 (± 0.33)
Bio-Inspired	PSO	1000	38.78 (± 0.68)	55.3 (± 0.62)
Mathematical-models	DP	392	52.79 (± 1.3)	0.0 (± 0.0)
Reinforcement Learning	A2C	734	54.35 (± 0.66)	0.04 (± 0.0)
	DQN	934	49.33 (± 0.91)	0.02 (± 0.0)
	QRDQN	878	60.97 (± 2.45)	0.03 (± 0.0)
	TRPO	983	46.96 (± 0.72)	0.04 (± 0.0)
	PPO	948	48.16 (± 0.4)	0.04 (± 0.0)

2-D experiment and their popularity, we decided to evaluate A* and two of the most popular RL algorithms, A2C and PPO,[3] in a 3-D setup that is described in the following subsection.

B. SIMULATION ENVIRONMENT

In the absence of a real-world environment and a drone for the training and evaluation of our model, we perform all the experiments in a simulation environment. For this purpose, we employed Microsoft's AirSim[4] simulator, which is a very powerful simulation environment for vehicles and UAVs, that offers many options for landscape setup (within the Unreal Engine), provides an extremely flexible programming environment and allows the creation of realistic scenarios that support the use of reinforcement learning techniques. The AirSim is already used by numerous researchers and has proven to offer a high level of realism in the quadcopter kinematic model and accurate simulation of the included sensors [8], [99], [100]. The environment can be characterized as a medium complexity stage with multiple buildings but it had limited flying area. In the current work, we decided to create two variations of this environment. The first environment has high complexity based on the high density and the short distances between the buildings (formations of blocks), which we further call Hard Area (HA). The second environment has a low density of buildings, which we further call Easy Area (EA). Also in both worlds, we have created two rectangle areas on the edges of the allowing flying zone, in order to use them as starting and ending zones. In every episode, the drone randomly spawns inside the starting area at a random altitude and a new random destination is set inside the end zone. All the buildings are placed between those two areas. Those areas are colored with white color and demonstrated with the two environments in Figures 5 and 6. The usage of these two zones and the randomness in spawn in destination points was important

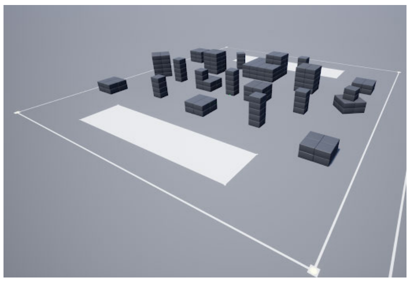


FIGURE 5. **Hard world.**

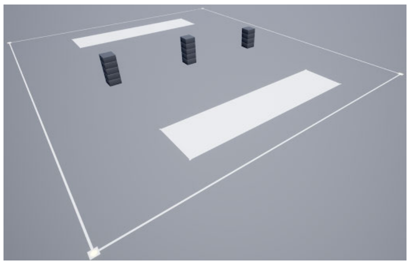


FIGURE 6. **Easy world.**

because eliminates any directional bias and also push the agent to achieve better exploration of the area. In order to guarantee the repeatability of the experiments, with the various algorithms that we tested, we generated 10,000 routes (pairs of random take-off and landing points) for the hard area, which could be used for route discovery during training. Furthermore, we produced an additional 1,000 pairs of take-off and landing points for the evaluation of the path-planning algorithms in each area.

AirSim offers a wide variety of environment-sensing sensors and allows users to attach any number of sensors in any location on the vehicle [99]. For our experiments, we used the default quad copter provided by the AirSim simulator and attached four distance sensors to it in the following positions: three sensors were placed in the middle front of the drone, with two of them rotated by +45 and −45 degrees to detect the edges of obstacles. These three sensors were responsible for detecting any obstacle in front of the vehicle. The fourth distance sensor was placed in the belly of the drone and was responsible for measuring the distance of the floor or any

[3]PPO is a first-order approximation of TRPO and has proven to be faster than TRPO in more complex neural networks and tasks [98]. Although TRPO achieves better path length and is comparable in execution time to PPO in this small grid-map setup, we decided to use PPO instead for the 3-D path planning task.

[4]https://microsoft.github.io/AirSim/

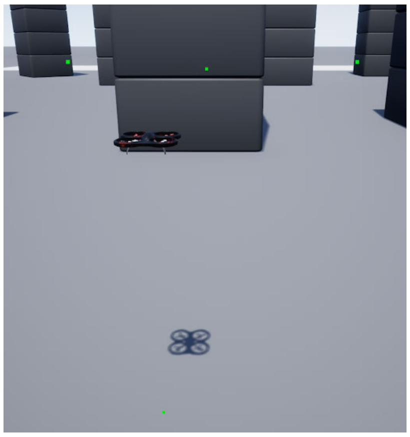


FIGURE 7. Drone setup. The four single-point laser distance sensors are depicted with green dots.

TABLE 3. Hyper-parameters per algorithm.

	A2C	PPO
Hidden Size	52.0	109.00
Learning Rate	0.0001	0.0001
Gamma	0.966549	0.959604
GAE Lamda	0.913836	0.933405
Activation Function	ReLu	ReLU
Optimizer	RMSprop	Adam

obstacle below. In Figure 7 we presented how the distance sensor was placed. Additionally, we used the IMU sensor and GPS sensor of the drone to provide extra information to the model and also to obtain positional information.

C. TRAINING PARAMETERS

1) RL MODELS HYPERPARAMETER TUNING

To tune the depth of the neural network and other training hyperparameters, we utilized the open-source hyperparameter optimization framework, Optuna.[5] Within this framework, we created two instances of the environment, one for each algorithm. Both A2C and PPO used the same underlying neural network models. The neural network model used by the actor had an input layer, 3 batch normalization layers followed by a dense layer each (with 128, 64, 32 neurons using Relu respectively), and an additional dense output layer with 6 neurons using the tanh function. The critic employed a similar network but with a single neuron using the linear function in the output layer. In these environments, we executed 100 runs of 1,000 time steps for each RL algorithm to fine-tune its parameters. We chose Optuna's default sampler, which utilizes the Tree-structured Parzen Estimator algorithm. After the completion of the sampling rounds, we identified the optimal training parameter for each algorithm. The results of the study and all the hyperparameters are presented in Table 3.

[5]https://optuna.org/

2) REWARD FUNCTION

In the reward function, we utilized penalties of $r_{coll_{floor}}$, $r_{coll_{obst}}$, $r_{timeout}$, and $r_{overshoot}$ with a fixed value of -100, while r_{goal} was assigned a value of 500. To calculate r_{obs}, we employed weights w_{arc} and w_{eucl} with an empirical value of 0.5. Additionally, the weights w_{vel} and w_{obst} were calculated using exponential functions with scaling factors a_o and a_v set to -5 and -0.2, respectively. During training, we generated 10,000 hard-area random routes. These training routes were solely used to train the PPO and A2C algorithms, as the A* algorithm does not require a training phase.

D. COMPARATIVE ANALYSIS

Our point of reference for comparison is an implementation of the A* algorithm, which is considered a challenging benchmark since the drone in the A* scenario is equipped with a highly dense LiDAR that delivers abundant depth information regarding obstacles. Based on the LiDAR information the drone is able to update the information about obstacles in its route and move or deviate accordingly.

The second method we used for comparison is another reinforcement learning algorithm from the Actor-Critic family of algorithms, more specifically the Advantage Actor-Critic (A2C). A2C employs an advantage function to estimate the advantage of each action, which quantifies the improvement of an action compared to the average action in a given state. This advantage guides policy updates and leads to more informed and efficient exploration. The "critic" component of A2C estimates the state-value function and provides a baseline for the advantage estimation. This helps reduce the variance of policy updates, resulting in more stable learning. PPO is from the same family of Actor-Critic algorithms but differs from A2C in a few critical points. It combines the minimization of a surrogate objective function, which maximizes the expected reward with limited change in the policy, and has a clipping mechanism that limits the amount by which the new policy can deviate from the old one [97].

E. EVALUATION METRICS

To assess the efficacy of our approach and compare it with other methods, we use a set of metrics that evaluate the ability of the planning algorithm to safely guide the UAV to its destination, the efficiency of the algorithm in terms of the time needed to compute the right path and its efficacy in quickly navigating the UAV on target. The first group of metrics that examine whether the route is completed successfully or resulted in failure (such as collision with the ground or an obstacle, getting stuck in a repetitive movement for a prolonged period, or crossing the world boundaries) are defined as follows:

- Reach target zone (RTZ): the number of routes that the drone safely reached the target zone.
- Reach target (RT): the number of routes that the drone reached the designated target.

- Crash: the number of routes that the drone collided with an obstacle.
- Out of limits: the number of routes that the drone went out of the area limits.
- Flight timeout: the number of routes that the drone failed to reach the target after a certain time after take off.

The second group of metrics pertains to flight time and distance. We also computed the total computational time required for each algorithm to determine the next move, as well as the clean flight time (flight time minus computational time). These metrics were only measured for successfully completed routes for all three algorithms. The metrics per route are as follows:

- RTZ time: the time from take off until entering the target zone.
- RT time: the time from take off until reaching the final target.
- RT flight time: the time needed for performing the necessary moves to reach the final target, excluding any path calculation time.
- RT calculation time: the time needed to perform all the path calculation steps during a flight.
- Route length: the total length of the route from the take-off until the final target.

Since all the routes have similar lengths and duration, we calculate the average time for all the successfully completed routes and the standard deviation.

V. RESULTS

To evaluate the performance of the PPO and A2C algorithm, we analyzed the performance of 10,000 randomly generated routes over a period of 500 episodes. From this analysis, we selected the top-performing routes based on their ability to maintain stable performance without any further improvement for at least 500 episodes during the training. We selected PPO route 6,300 and A2C route 4,446 as the top-performing routes using this criterion. The next step was to test all the algorithms in the 1,000 randomly generated evaluation routes. These evaluation routes were not used during training and were generated separately. The comparative evaluation of PPO against the alternative RL method of A2C and the A* algorithm is presented in three distinct parts.

A. FLIGHT SUCCESS

The first part is depicted in Figures 8 and 9 and shows how the three algorithms perform on the easy and hard stage setup.

In Figure 8 we see the performance of the three algorithms in terms of safely guiding the drone in reaching its target (success rate) in the easy area. First of all, we can see that the percentage of routes in which the *drone crashed* in an obstacle or on the floor is 1% for PPO, 0.4% for A*, and 0% for A2C, which means that both A2C and PPO have learned to avoid sparse obstacles effectively and A* using the LiDAR demonstrates the best performance.

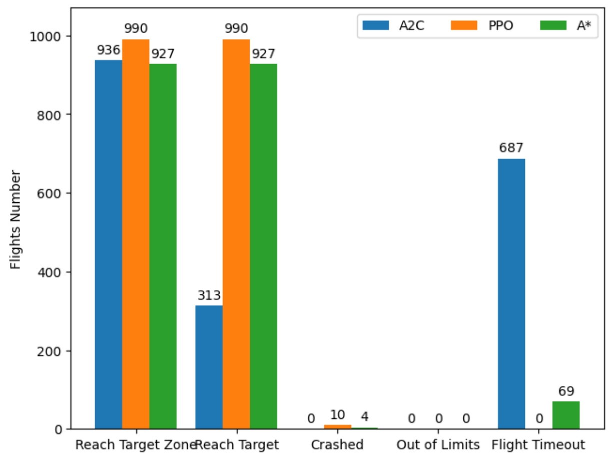


FIGURE 8. Easy area route completion statistics.

Secondly, all three algorithms have a success rate in *reaching the target zone* that is above 90% with PPO being the best at 99% and A2C and A* having a comparable performance with 93.6% and 92.7% respectively. An interesting observation is associated with the *Reach Target* metric. PPO is still the best of the three with 99% and A* is second with 92.7% success rate, whilst A2C has a very low success rate at 31.3%. These results mean that with A2C even though the drone reached the target zone it failed to reach the exact destination point on time. Analyzing the *Flights' timeout* metric we can see where A* and A2C lost to their counterpart algorithm PPO. In contrast to PPO which performed perfectly at a 0% rate, we see that for 68.7% of the time A2C flew for too long and thus terminated its flight reducing massively the success rate of reaching the target zone. A* due to its high computational costs amassed 6.9% of flight timeouts. Lastly, all algorithms performed perfectly when it comes to staying within the world's limits.

Figure 9 presents the performance of the three algorithms in the hard area. First of all, we can see that the crash ratio has increased for all algorithms, compared to the easy area with the sparse obstacles. It is now 4.2% for PPO, 3.7% for A*, and 36.9% for A2C. The results show that while A2C was the best at avoiding sparse obstacles, has the worst performance in the dense obstacle world. PPO and A* come close to each other meaning that PPO has learned to avoid effectively the dense obstacles with slight room for improvement. Secondly, A2C and A* had drastically done worse in *reaching the target zone* with a 60.8% and 59.4% success rate respectively, with only PPO maintaining a high success rate at 95.7%, which is comparable to that of the easy area. The success rate in *reaching the target* compared to that of reaching the target area demonstrates a small drop for all algorithms. Analyzing the flight timeout metric we can see that once again the A2C algorithm has a small timeout rate (9.2% flight timeout rate), whereas PPO performed perfectly with a 0% timeout rate. However, A* had massively increased its flight timeout rate to 36.9% highly due to its increased computational

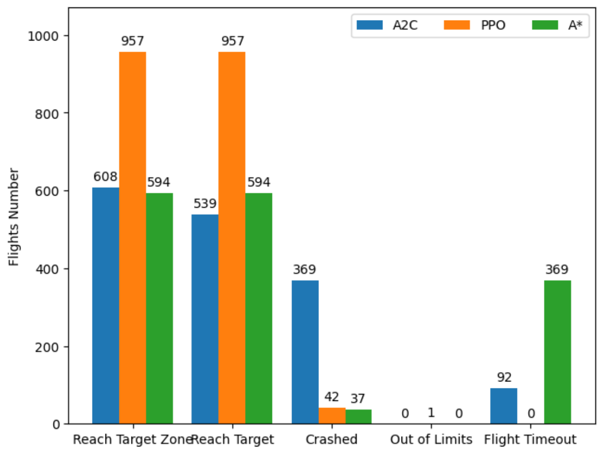


FIGURE 9. Hard area route completion statistics.

cost in dense environments. Lastly, all algorithms performed perfectly when it comes to staying within the world limits with PPO this time having a 0.1% in out-of-limits rate.

B. FLIGHT DURATION

The second set of results, as presented in Tables 4 and 5, illustrates the time performance of the three algorithms in the two areas. In Table 4 we see the values of the time metrics used in the three algorithms in the easy area. First, we observe the Reach Target Zone (RTZ) Time with the fastest algorithms being PPO and A* at 264s and 210s respectively and the slowest being A2C at 600s. Although the average times for all the completed routes are comparable for PPO and A*, the standard deviation of values in the case of PPO is 14s as opposed to A*'s 105s, which shows that with PPO consistently the drone reaches the target zone in less than 4.5 minutes (264 seconds).

By analyzing the time needed to Reach Target, we can see that A2C takes the longest to reach the target destination (844s) with PPO and A* being the fastest (at 280s and 236s respectively). Since the A* algorithm has a standard deviation of 110s it means that PPO is consistently fast, where A* can quickly lead to the target in some routes and can take much more time than PPO in others. By subtracting the RTZ Time from the RT Time we see that A2C, PPO, and A* take 244s, 16s, and 26s respectively on average to guide the drone to the target, once it has reached the target zone. This means that A2C makes a lot of unnecessary movements to reach the target, whereas PPO is the most efficient in tracking and reaching the destination fast.

Next, we break down the RT Time into two subcategories RT Flight Time and RT Calculation Time in order to understand where the algorithms allocate their time. By converting the times to ratios we get that A2C spends about 97.51% of its total flight time in performing the decided action and 2.49% in calculating the action to be taken. Similarly, PPO spends 98.57% for navigation and 1.43% for path calculation, and A* 57.62% and 42.38% respectively. This shows that PPO and A2C are much faster at calculating the appropriate actions whereas A* requires a significant amount of time in performing its calculations. Considering the standard deviation as shown in Table 4 PPO is the most efficient in terms of the overall flight time in the easy area flights.

In Table 5 we see the values of the time metrics used in the three algorithms in the hard area. Firstly, we observe the Reach Target Zone Time with the fastest algorithm being PPO at 277s, and A* with A2C being the slowest with similar times at 494s and 525s respectively. Next by analyzing the Reach Target Time, we can see that A2C takes the longest to reach the target destination at 669s with PPO being the fastest at 292s and A* in between at 509s. Subtracting the RTZ Time from the RT Time we see that A2C, PPO, and A* take 144s, 15s, and 15s to guide the drone to the target once it reaches the target zone. This means that again A2C makes a lot of unnecessary movements to reach the target whereas PPO and A* are the most efficient in tracking and reaching the destination fast.

Once again for the hard area, we break down the RT Time into two subcategories RT Flight Time and RT Calculation Time in order to understand where is the time allocated to. By converting the times to ratios we get that A2C spends about 97.45% of its RT Time performing the decided action and 2.55% calculating the action to be taken. Similarly, PPO spends 98.63% and 1.37%, and A* 34.97% and 65.03%. This shows that PPO and A2C are much faster at calculating the appropriate actions compared to A*. Considering the standard deviation as shown in Table 5 PPO is the most efficient in overall time. Lastly, we see that A* did a lot longer to calculate the path that it must follow as noted in both the final ratios and the overall time meaning that the more complex the world the worse it performs whilst A2C and especially PPO are not affected by the state of the world when it comes to more complex environments.

TABLE 4. Easy area performance statistics.

Method	Mean RTZ Time	Mean RT Time	Mean RT Flight Time	Mean RT Calc Time
A2C	600.115 (± 12.28)	844.35 (± 6.59)	823.054 (± 6.42)	21.296 (± 0.22)
PPO	264.606 (± 2.26)	280.161 (± 2.56)	276.523 (± 2.53)	3.638 (± 0.03)
A*	210.100 (± 17.29)	236.133 (± 17.94)	136.846 (± 2.64)	99.287 (± 16.0)

TABLE 5. Hard area performance statistics.

Method	Mean RTZ Time	Mean RT Time	Mean RT Flight Time	Mean RT Calc Time
A2C	525.12 (± 18.76)	669.663 (± 23.74)	652.219 (± 23.13)	17.444 (± 0.61)
PPO	277.90 (± 4.17)	292.507 (± 4.38)	288.678 (± 4.32)	3.829 (± 0.06)
A*	494.08 (± 28.62)	509.034 (± 28.84)	178.633 (± 3.79)	330.401 (± 26.05)

The better time complexity of the reinforcement learning algorithms is mainly due to the fact that the implementation of A* requires the algorithm to find the complete path to the destination at each step (starting from the current position each time), whereas in the case of A2C and PPO the algorithm evaluates the state and takes only a single action for the next step. This makes the proposed approach more flexible and fast.

C. ROUTE LENGTH

The last set of results presented in Figures 10 and 11 shows the mean and standard deviation of route lengths, for the completed routes of each method.

In Figure 10 we showcase the average path length that each algorithm had with its appropriate standard deviation, in the easy area. We can observe that PPO is the most effective algorithm in finding the shortest path to follow while A2C and A* perform similarly with A* having the largest standard deviation meaning that A2C and PPO are more consistent.

In Figure 11 we showcase the average path length that each algorithm had with the respective standard deviation, in the hard area. We can observe that PPO is the most effective algorithm in finding the shortest path to follow while A2C and A* perform similarly. PPO is still more consistent than both A2C and A* even when having a more difficult environment.

Another noteworthy observation is that while analyzing the timing of the three algorithms, A* proved to be the fastest while performing the actions that were taken. In contrast,

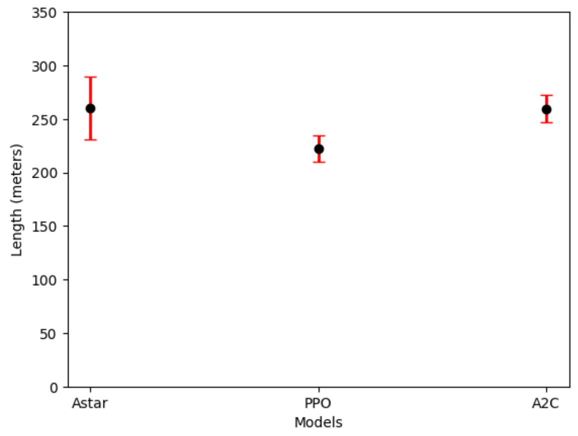


FIGURE 10. Easy area path length.

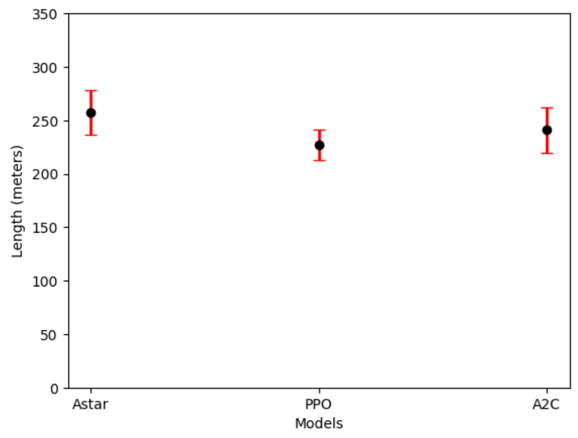


FIGURE 11. Hard area path length.

we see that instead of those actions causing a smaller path than the other two algorithms, it takes a slightly longer path. This means that both PPO and A2C are having much longer Flight Time values not because they are poor at path planning but because the speed at which the UAV navigates to the target is smaller.

VI. CONCLUSION

The experimental evaluation performed in this study investigated the effectiveness of two autonomous navigation algorithms, namely PPO and A2C, which were implemented using low-cost sensors, in comparison to a typical path-planning algorithm, namely A* that employs rich information from a LiDAR for scene perception. Each algorithm has its unique strengths and weaknesses, and it is important to discuss them in detail to understand their suitability in this context. We found that PPO is particularly advantageous in using low-cost sensors due to its ability to learn from past experiences and optimize the policy accordingly. Our sensor setup enabled us to avoid obstacles effortlessly without significantly delaying the route completion time. This was demonstrated by comparing the flight times between the easy and hard areas. However, A* also performed well, approaching the performance of PPO. Nevertheless, the A* implementation utilized a more expensive sensor (LiDAR) and suffered from the high computational time between steps, as well as a limitation in reacting to changes in the environment while moving. In contrast, the A2C exhibited the worst performance, mainly due to the large size of the environment. This led to the problem of catastrophic forgetting, where the A2C did not have a mechanism to utilize the available data optimally and instead relied on the quality of data from the last batch.

This work has demonstrated that using PPO in combination with low-cost sensors can be a highly effective solution for the problem of path planning in unconstrained and unknown environments, without previous knowledge of the obstacles. As demonstrated experimentally in a flight simulation environment the main advantage of the proposed solution is that it can successfully guide the drone to the target and avoid obstacles using only a few low-cost sensors, and has a small computational time compared to other algorithms, which results in low computational and hardware costs, which makes it a practical solution for real-world applications. In the proposed solution, the primary focus was to find a balance between computational demands and path length. As such, the generated paths and our metrics do not guarantee absolute optimality, although they can approximate an optimal solution. Given the ever-changing and unpredictable nature of real-world scenarios, an optimal solution might not exist. However, this will be investigated in future work with a specially designed metric and experiments for that purpose.

In summary, our approach utilizes only four sensors to perceive the environment and combines their input with an RL algorithm to make navigation decisions. The unique reward function that we crafted makes optimal use of this limited information while guiding the model's policy. Since no detailed information on the drone environment is used, the model avoids overfitting and enhances its generalization.

As part of our next steps in this work, we plan to repeat the same set of experiments, this time using other types of sensors, such as cameras, to leverage additional information from the environment. Finally, we aim to investigate the robustness of our methodology in continuously changing environments with moving objects, as all experiments conducted in this study were performed in static environments.

REFERENCES

[1] M. Moshref-Javadi and M. Winkenbach, "Applications and research avenues for drone-based models in logistics: A classification and review," *Expert Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114854.

[2] M. Hassanalian and A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review," *Prog. Aerosp. Sci.*, vol. 91, pp. 99–131, May 2017.

[3] M. Hussein, R. Nouacer, F. Corradi, Y. Ouhammou, E. Villar, C. Tieri, and R. Castiñeira, "Key technologies for safe and autonomous drones," *Microprocessors Microsyst.*, vol. 87, Nov. 2021, Art. no. 104348.

[4] P. M. Kornatowski, A. Bhaskaran, G. M. Heitz, S. Mintchev, and D. Floreano, "Last-centimeter personal drone delivery: Field deployment and user interaction," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3813–3820, Oct. 2018.

[5] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Robot. Res.*, vol. 40, nos. 4–5, pp. 698–721, Apr. 2021.

[6] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," 2018, *arXiv:1803.11347*.

[7] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 1205–1212.

[8] K. Kersandt, G. Muñoz, and C. Barrado, "Self-training by reinforcement learning for full-autonomous drones of the future," in *Proc. IEEE/AIAA 37th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2018, pp. 1–10.

[9] U. Ates, "Long-term planning with deep reinforcement learning on autonomous drones," in *Proc. Innov. Intell. Syst. Appl. Conf. (ASYU)*, Oct. 2020, pp. 1–6.

[10] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data," *Neural Comput. Appl.*, vol. 33, no. 6, pp. 2015–2033, Mar. 2021.

[11] E. Balestrieri, P. Daponte, L. De Vito, F. Picariello, and I. Tudosa, "Sensors and measurements for UAV safety: An overview," *Sensors*, vol. 21, no. 24, p. 8253, Dec. 2021.

[12] M. Jones, S. Djahel, and K. Welsh, "Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–39, Nov. 2023.

[13] E. Politi, A. Garyfallou, I. Panagiotopoulos, I. Varlamis, and G. Dimitrakopoulos, "Path planning and landing for unmanned aerial vehicles using AI," in *Proc. Future Technol. Conf. (FTC)*, vol. 1. New York, NY, USA: Springer, 2022, pp. 343–357.

[14] P. Pandey, A. Shukla, and R. Tiwari, "Aerial path planning using meta-heuristics: A survey," in *Proc. 2nd Int. Conf. Electr., Comput. Commun. Technol. (ICECCT)*, Feb. 2017, pp. 1–7.

[15] F. Mumuni, A. Mumuni, and C. K. Amuzuvi, "Deep learning of monocular depth, optical flow and ego-motion with geometric guidance for UAV navigation in dynamic environments," *Mach. Learn. Appl.*, vol. 10, Dec. 2022, Art. no. 100416.

[16] J. L. Giesbrecht, "Global path planning for unmanned ground vehicles," Defence Res. Develop., Suffield, CT, Canada, Tech. Rep. 2004-272, 2004.

[17] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Comput. Commun.*, vol. 149, pp. 270–299, Jan. 2020.

[18] M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-PRM: Temporal probabilistic roadmap for path planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 10320–10327.

[19] P. Švestka and M. Overmars, "Probabilistic path planning," in *Robot Motion Planning and Control*. New York, NY, USA: Springer, 2005, pp. 255–304.

[20] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symp. (ICRA)*, vol. 2, Apr. 2000, pp. 995–1001.

[21] W. Wang, H. Gao, Q. Yi, K. Zheng, and T. Gu, "An improved RRT* path planning algorithm for service robot," in *Proc. IEEE 4th Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, vol. 1, Jun. 2020, pp. 1824–1828.

[22] 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.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[23] P. Bhattacharya and M. Gavrilova, "Roadmap-based path planning—Using the Voronoi diagram for a clearance-based shortest path," *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, Jun. 2008.

[24] A. Candra, M. A. Budiman, and K. Hartanto, "Dijkstra's and A-Star in finding the shortest path: A tutorial," in *Proc. Int. Conf. Data Sci., Artif. Intell., Bus. Anal. (DATABIA)*, Jul. 2020, pp. 28–32.

[25] A. Budiyanto, A. Cahyadi, T. B. Adji, and O. Wahyunggoro, "UAV obstacle avoidance using potential field under dynamic environment," in *Proc. Int. Conf. Control, Electron., Renew. Energy Commun. (ICCEREC)*, Aug. 2015, pp. 187–192.

[26] H. Sharma, T. Sebastian, and P. Balamuralidhar, "An efficient backtracking-based approach to turn-constrained path planning for aerial mobile robots," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2017, pp. 1–8.

[27] Y. Xu and C. Che, "A brief review of the intelligent algorithm for traveling salesman problem in UAV route planning," in *Proc. IEEE 9th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, Jul. 2019, pp. 1–7.

[28] K. Peng, J. Du, F. Lu, Q. Sun, Y. Dong, P. Zhou, and M. Hu, "A hybrid genetic algorithm on routing and scheduling for vehicle-assisted multi-drone parcel delivery," *IEEE Access*, vol. 7, pp. 49191–49200, 2019.

[29] R. Shivgan and Z. Dong, "Energy-efficient drone coverage path planning using genetic algorithm," in *Proc. IEEE 21st Int. Conf. High Perform. Switching Routing (HPSR)*, May 2020, pp. 1–6.

[30] Y. V. Pehlivanoglu and P. Pehlivanoglu, "An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems," *Appl. Soft Comput.*, vol. 112, Nov. 2021, Art. no. 107796.

[31] S. Poudel, M. Y. Arafat, and S. Moh, "Bio-inspired optimization-based path planning algorithms in unmanned aerial vehicles: A survey," *Sensors*, vol. 23, no. 6, p. 3051, Mar. 2023.

[32] J. L. Foo, J. Knutzon, V. Kalivarapu, J. Oliver, and E. Winer, "Path planning of unmanned aerial vehicles using B-splines and particle swarm optimization," *J. Aerosp. Comput., Inf., Commun.*, vol. 6, no. 4, pp. 271–290, Apr. 2009.

[33] A. Sharma, S. Shoval, A. Sharma, and J. K. Pandey, "Path planning for multiple targets interception by the swarm of UAVs based on swarm intelligence algorithms: A review," *IETE Tech. Rev.*, vol. 39, no. 3, pp. 675–697, May 2022.

[34] G. M. Nayeem, M. Fan, and Y. Akhter, "A time-varying adaptive inertia weight based modified PSO algorithm for UAV path planning," in *Proc. 2nd Int. Conf. Robot., Electr. Signal Process. Techn. (ICREST)*, Jan. 2021, pp. 573–576.

[35] 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. 7, pp. 105086–105099, 2019.

[36] U. Cekmez, M. Ozsiginan, and O. K. Sahingoz, "A UAV path planning with parallel ACO algorithm on CUDA platform," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2014, pp. 347–354.

[37] S. Konatowski and P. Pawłowski, "Ant colony optimization algorithm for UAV path planning," in *Proc. 14th Int. Conf. Adv. Trends Radioelecrtron., Telecommun. Comput. Eng. (TCSET)*, Feb. 2018, pp. 177–182.

[38] B. Li, X. Qi, B. Yu, and L. Liu, "Trajectory planning for UAV based on improved ACO algorithm," *IEEE Access*, vol. 8, pp. 2995–3006, 2020.

[39] J. Chen, F. Ye, and Y. Li, "Travelling salesman problem for UAV path planning with two parallel optimization algorithms," in *Proc. Prog. Electromagn. Res. Symp. Fall (PIERS FALL)*, 2017, pp. 832–837.

[40] C. Qu, W. Gai, J. Zhang, and M. Zhong, "A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (UAV) path planning," *Knowl.-Based Syst.*, vol. 194, Apr. 2020, Art. no. 105530.

[41] R. K. Dewangan, A. Shukla, and W. W. Godfrey, "Three dimensional path planning using grey wolf optimizer for UAVs," *Appl. Intell.*, vol. 49, no. 6, pp. 2201–2217, Jun. 2019.

[42] W. Jiang, Y. Lyu, Y. Li, Y. Guo, and W. Zhang, "UAV path planning and collision avoidance in 3D environments based on POMPD and improved grey wolf optimizer," *Aerosp. Sci. Technol.*, vol. 121, Feb. 2022, Art. no. 107314.

[43] K. Wu, T. Xi, and H. Wang, "Real-time three-dimensional smooth path planning for unmanned aerial vehicles in completely unknown cluttered environments," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2017, pp. 2017–2022.

[44] J. Faigl and P. Váňa, "Surveillance planning with Bézier curves," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 750–757, Apr. 2018.

[45] S. Ragi and E. K. P. Chong, "UAV path planning in a dynamic environment via partially observable Markov decision process," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 4, pp. 2397–2412, Oct. 2013.

[46] V. Darbari, S. Gupta, and O. P. Verma, "Dynamic motion planning for aerial surveillance on a fixed-wing UAV," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2017, pp. 488–497.

[47] M. Radmanesh, M. Kumar, A. Nemati, 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, no. 9, pp. 1668–1682, Jul. 2016.

[48] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annu. Rev. Control*, vol. 51, pp. 65–87, Jan. 2021.

[49] B. Bakker, Z. Zivkovic, and B. Krose, "Hierarchical dynamic programming for robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Aug. 2005, pp. 2756–2761.

[50] K. Shin and N. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Trans. Autom. Control*, vol. AC-31, no. 6, pp. 491–500, Jun. 1986.

[51] A. L. Jennings, R. Ordonez, and N. Ceccarelli, "Dynamic programming applied to UAV way point path planning in wind," in *Proc. IEEE Int. Conf. Comput.-Aided Control Syst.*, Sep. 2008, pp. 215–220.

[52] L. Wirth, P. Oettershagen, J. Ambühl, and R. Siegwart, "Meteorological path planning using dynamic programming for a solar-powered UAV," in *Proc. IEEE Aerosp. Conf.*, Mar. 2015, pp. 1–11.

[53] R. Diankov and J. Kuffner, "Randomized statistical path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 1–6.

[54] M. Menner, K. Berntorp, M. N. Zeilinger, and S. D. Cairano, "Inverse learning for data-driven calibration of model-based statistical path planning," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 1, pp. 131–145, Mar. 2021.

[55] S. Y. Choi and D. Cha, "Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art," *Adv. Robot.*, vol. 33, no. 6, pp. 265–277, Mar. 2019.

[56] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 161–168.

[57] H. Jiang and Y. Liang, "Online path planning of autonomous UAVs for bearing-only standoff multi-target following in threat environment," *IEEE Access*, vol. 6, pp. 22531–22544, 2018.

[58] M. Kang, Y. Liu, and Y. Zhao, "A threat modeling method based on Kalman filter for UAV path planning," in *Proc. 29th Chin. Control Decis. Conf. (CCDC)*, May 2017, pp. 3823–3828.

[59] Z. Wu, J. Li, J. Zuo, and S. Li, "Path planning of UAVs based on collision probability and Kalman filter," *IEEE Access*, vol. 6, pp. 34237–34245, 2018.

[60] H. B. Makineci, H. Karabörk, and A. Durdu, "ANN estimation model for photogrammetry-based UAV flight planning optimisation," *Int. J. Remote Sens.*, vol. 43, nos. 15–16, pp. 5686–5708, Aug. 2022.

[61] Y. Zhang, Y. Zhang, Z. Liu, Z. Yu, and Y. Qu, "Line-of-sight path following control on UAV with sideslip estimation and compensation," in *Proc. 37th Chin. Control Conf. (CCC)*, Jul. 2018, pp. 4711–4716.

[62] Y. Ma, Q. Khan, and D. Cremers, "Multi agent navigation in unconstrained environments using a centralized attention based graphical neural network controller," 2023, *arXiv:2307.16727*.

[63] W. Shi, Z. Dong, and L. Zhang, "Path planning of unmanned aerial vehicle based on supervised learning," in *Proc. IEEE 8th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2022, pp. 747–751.

[64] G. Tartaglione and M. Ariola, "Obstacle avoidance via landmark clustering in a path-planning algorithm," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 2776–2781.

[65] L. Zheng, P. Zhang, J. Tan, and F. Li, "The obstacle detection method of UAV based on 2D LiDAR," *IEEE Access*, vol. 7, pp. 163437–163448, 2019.

[66] C. Teuliere, E. Marchand, and L. Eck, "3-D model-based tracking for UAV indoor localization," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 869–879, May 2015.

[67] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–8.

[68] L. Tai and M. Liu, "Towards cognitive exploration through deep reinforcement learning for mobile robots," 2016, *arXiv:1610.01733*.

[69] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[70] Z. Yao, X. Liang, G.-P. Jiang, and J. Yao, "Model-based reinforcement learning control of electrohydraulic position servo systems," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 3, pp. 1446–1455, Jun. 2022.

[71] S. Luan, Y. Yang, H. Wang, B. Zhang, B. Yu, and C. He, "3D G-learning in UAVs," in *Proc. 12th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2017, pp. 953–957.

[72] Z. Yijing, Z. Zheng, Z. Xiaoyi, and L. Yang, "Q learning algorithm based UAV path learning and obstacle avoidence approach," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 3397–3402.

[73] C. Chronis, G. Anagnostopoulos, E. Politi, A. Garyfallou, I. Varlamis, and G. Dimitrakopoulos, "Path planning of autonomous UAVs using reinforcement learning," *J. Phys., Conf. Ser.*, vol. 2526, no. 1, 2023, Art. no. 012088.

[74] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "UAV path planning for wireless data harvesting: A deep reinforcement learning approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[75] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "Autonomous UAV navigation: A DDPG-based deep reinforcement learning approach," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[76] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.

[77] I. Kim, S. Shin, J. Wu, S.-D. Kim, and C.-G. Kim, "Obstacle avoidance path planning for UAV using reinforcement learning under simulated environment," in *Proc. 3rd Int. Conf. Electron., Electr. Eng., Comput. Sci. (IASER)*, Okinawa, Japan, 2017, pp. 34–36.

[78] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous UAV navigation using reinforcement learning," 2018, *arXiv:1801.05086*.

[79] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2125–2140, Apr. 2019.

[80] C. Yan, X. Xiang, and C. Wang, "Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments," *J. Intell. Robot. Syst.*, vol. 98, no. 2, pp. 297–309, May 2020.

[81] Y.-M. Wang and D.-L. Peng, "A simulation platform of multi-sensor multi-target track system based on STAGE," in *Proc. 8th World Congr. Intell. Control Autom.*, Jul. 2010, pp. 6975–6978.

[82] L. He, N. Aouf, J. F. Whidborne, and B. Song, "Deep reinforcement learning based local planner for UAV obstacle avoidance using demonstration data," 2020, *arXiv:2008.02521*.

[83] X. Fu, J. Zhu, Z. Wei, H. Wang, and S. Li, "A UAV pursuit-evasion strategy based on DDPG and imitation learning," *Int. J. Aerosp. Eng.*, vol. 2022, pp. 1–14, May 2022.

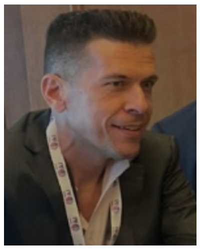
[84] A. Alagha, R. Mizouni, J. Bentahar, H. Otrok, and S. Singh, "Multi-agent deep reinforcement learning with demonstration cloning for target localization," *IEEE Internet Things J.*, vol. 10, no. 15, pp. 13556–13570, Aug. 2023.

[85] J. Wang, P. Zhang, and Y. Wang, "Autonomous target tracking of multi-UAV: A two-stage deep reinforcement learning approach with expert experience," *Appl. Soft Comput.*, vol. 145, Sep. 2023, Art. no. 110604.
[86] N. Richards, M. Sharma, and D. Ward, "A hybrid A*/automaton approach to on-line path planning with obstacle avoidance," in *Proc. AIAA 1st Intell. Syst. Tech. Conf.*, Sep. 2004, p. 6229.
[87] P. Yao, H. Wang, and Z. Su, "Real-time path planning of unmanned aerial vehicle for target tracking and obstacle avoidance in complex dynamic environment," *Aerosp. Sci. Technol.*, vol. 47, pp. 269–279, Dec. 2015.
[88] P. Yao, H. Wang, and Z. Su, "Cooperative path planning with applications to target tracking and obstacle avoidance for multi-UAVs," *Aerosp. Sci. Technol.*, vol. 54, pp. 10–22, Jul. 2016.
[89] T. Zhang, J. Xu, and B. Wu, "Hybrid path planning model for multiple robots considering obstacle avoidance," *IEEE Access*, vol. 10, pp. 71914–71935, 2022.
[90] L. Yafei, W. Anping, C. Qingyang, and W. Yujie, "An improved UAV path planning method based on RRT-APF hybrid strategy," in *Proc. 5th Int. Conf. Autom., Control Robot. Eng. (CACRE)*, Sep. 2020, pp. 81–86.
[91] B. Sangiovanni, G. P. Incremona, M. Piastra, and A. Ferrara, "Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 397–402, Apr. 2021.
[92] F. Kiani, A. Seyyedabbasi, R. Aliyev, M. U. Gulle, H. Basyildiz, and M. A. Shah, "Adapted-RRT: Novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms," *Neural Comput. Appl.*, vol. 33, no. 22, pp. 15569–15599, Nov. 2021.
[93] Y. Chen, G. Bai, Y. Zhan, X. Hu, and J. Liu, "Path planning and obstacle avoiding of the USV based on improved ACO-APF hybrid algorithm with adaptive early-warning," *IEEE Access*, vol. 9, pp. 40728–40742, 2021.
[94] M. Abebe, Y. Noh, Y.-J. Kang, C. Seo, D. Kim, and J. Seo, "Ship trajectory planning for collision avoidance using hybrid ARIMA-LSTM models," *Ocean Eng.*, vol. 256, Jul. 2022, Art. no. 111527.
[95] Y. Shin and E. Kim, "Hybrid path planning using positioning risk and artificial potential fields," *Aerosp. Sci. Technol.*, vol. 112, May 2021, Art. no. 106640.
[96] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[97] Y. Wang, H. He, and X. Tan, "Truly proximal policy optimization," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 113–122.
[98] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep RL: A case study on PPO and TRPO," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–14.
[99] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. New York, NY, USA: Springer, 2018, pp. 621–635.
[100] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, "Federated transfer reinforcement learning for autonomous driving," in *Federated and Transfer Learning*. New York, NY, USA: Springer, 2022, pp. 357–371.

CHRISTOS CHRONIS received the Bachelor of Science (B.Sc.) degree in informatics and telematics from the Harokopio University of Athens, where he is currently pursuing the Ph.D. degree with the Department of Informatics and Telematics. With a primary focus on recommendation systems utilizing reinforcement learning, he strives to optimize personalized user experiences. He has actively contributed to numerous EU projects, showcasing his expertise in machine learning, system architectures, autonomous vehicles, robotics, and the IoT. Through his dedication and passion, he has actively participated in multiple publications featured in international journals and conferences.

GEORGIOS ANAGNOSTOPOULOS received the B.Sc. degree in informatics and telematics from Harokopio University. He is currently a Junior Researcher with Harokopio University, where he specializes in developing reinforcement learning models for object avoidance and path planning on UAVs. He has coauthored a publication on *Path Planning of Autonomous UAVs Using Reinforcement Learning* that has been accepted for publication and is pending release, in 2023. The paper was presented at the 12th EASN International Conference on Innovation in Aviation & Space for Opening New Horizons in Barcelona, in October. He is committed to advancing the field of UAVs through his research and contributions to academia.

ELENA POLITI received the degree in physics from the National and Kapodistrian University of Athens, and the M.Sc. degree in telecommunication networks and telematic services from the Harokopio University of Athens, where she is currently pursuing the Ph.D. degree. Her research interests include intelligent transport systems, AI functionalities, and optimization algorithms for autonomous vehicles. She has also participated in several international research projects and she has published several scientific papers in prestigious journals and conferences.



GEORGE DIMITRAKOPOULOS received the bachelor's degree in electrical and computer engineering from the National Technical University of Athens, in 2002, and the Ph.D. degree from the University of Piraeus, in 2007. He is currently an Associate Professor with the Department of Informatics and Telematics, Harokopio University of Athens. He is the author of about 100 publications in international journals and conferences, whereas he has been involved in numerous internationally funded research and development projects. His research interests include the design and development of strategies for the optimization of wireless networks based on cognitive networking principles, and the design of intelligent transportation systems (ITS) and novel ICT healthcare applications.

IRAKLIS VARLAMIS (Member, IEEE) received the M.Sc. degree in information systems engineering from UMIST, U.K., and the Ph.D. degree from the Athens University of Economics and Business, Greece. He is currently an Associate Professor in the area of data management with the Department of Informatics and Telematics, Harokopio University of Athens (HUA). His research interests include data mining and social network analytics to recommender systems for social media and real-world applications. He has more than 200 articles published in international journals and conferences and more than 4000 citations on his work. He holds a patent from the Greek Patent Office for a system that thematically groups web documents using content and links. He is the scientific coordinator for HUA in several EU (H2020, ECSEL, REC) projects and in national projects.