eFlight: RL Scheme for Autonomous Drones to Efficiently Fly through Obstacles

Yihan Xu
Department of Electrical and Computer Engineering
New York Institute of Technology
New York City, New York State, USA
yxu66@nyit.edu

Chuan-Bi Lin
Department of Information and Communication Engineering
Chaoyang University of Technology
Taichung, Taiwan
cblin@cyut.edu.twg

Cong Wang
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, New Jersey, USA
cong.wang@njit.edu

Wen Zhang
Department of Civil and Environmental Engineering
New Jersey Institute of Technology
Newark, New Jersey, USA
wen.zhang@njit.edu

Ziqian Dong
Department of Electrical and Computer Engineering
New York Institute of Technology
New York City, New York State, USA
ziqian.dong@nyit.edu

Roberto Rojas-Cessa
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, New Jersey, USA
rojas@njit.edu

Abstract

The flight time of uncrewed autonomous vehicles (UAVs) is constrained by its battery capacity, restricting its application in long-duration missions. To address this challenge, we propose eFlight, a hybrid scheme that uses a reinforcement-learning heuristic to augment A* for path finding. eFlight reduces both node expansions and computation time while finding energy-efficient paths in obstacle-dense 3D airspace. We compare eFlight with conventional path-planning algorithms for point-to-point flights on areas of various dimensions and with various obstacle densities. The results show that eFlight achieves a dual advantage: finding low-energy paths with short computation times. In high-density obstacle environment, eFlight identifies the lowest energy consumption path in 89.5% of the trials. Compared to the baseline scheme, eFlight reduces computation time by 90.6% ± 26.6% and energy by 7.13% ± 9.96%.

CCS Concepts

• **Computing methodologies** → Motion path planning.

Keywords

Reinforcement learning, A*, UAV, path finding, energy efficiency

ACM Reference Format:
Yihan Xu, Chuan-Bi Lin, Cong Wang, Wen Zhang, Ziqian Dong, and Roberto Rojas-Cessa. 2025. eFlight: RL Scheme for Autonomous Drones to Efficiently Fly through Obstacles. In *Proceedings of SEC'25.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3769102.3774246

SEC '25, Arlington, VA, USA

ACM ISBN 979-8-4007-2238-7/2025/12.
https://doi.org/10.1145/3769102.3774246

1 Introduction

Uncrewed Aerial Vehicles (UAVs) are becoming an increasingly integral part of remote sensing and monitoring tasks, such as environmental sensing and infrastructure inspection. However, their limited battery capacity restricts both flight duration and monitoring time, and so the variety and scope of real applications. To maximize mission time, drones must minimize energy expenditure while collecting data over extended missions but mostly during flight.

There is substantial work on flight path planning aimed at covering areas defined by numerous waypoints. However, those approaches neglect the energy expenditure of displacement vector changes that occur when flying from one waypoint to another, mostly in low altitude flights. Such point-to-point travel often faces obstacle-filled environments. Examples include buildings in urban settings or trees and hills in suburban and forest settings. Incorporating low-energy path finding in obstacle-dense settings would make the use of autonomous drones within a more practical reach [9].

Path-finding algorithms such as A*, Dijkstra's algorithm, genetic algorithms, and machine learning-based techniques have been recently explored [7, 8, 11]. While A* and Dijkstra algorithms perform well in static environments, they can be computationally intensive and of limited efficiency in dynamic and obstacle-rich scenarios [14]. Recent approaches dynamically adjust path weights to improve adaptability [4], yet they still struggle with abrupt and variable terrain changes.

Another research work has focused on active obstacle avoidance and reliable wireless connectivity during flight [1, 16]. However, many existing models consider drone energy expenditure depends solely on travel distance [12]. Therefore, factors such as acceleration, deceleration, and altitude changes needed for actual maneuvering remain neglected despite playing important roles in energy consumption and trajectory selection under realistic conditions.

To address this energy study gap, we introduce eFlight, a hybrid method that combines the A* algorithm with reinforcement learning

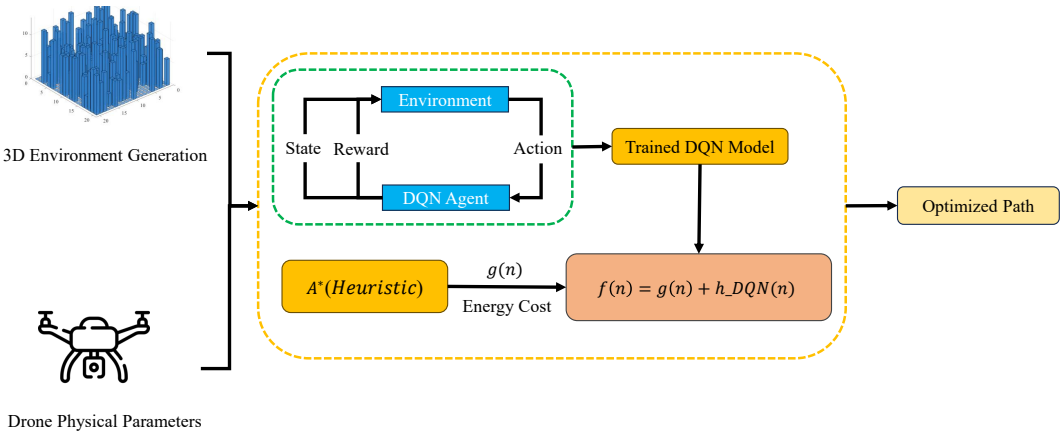

Figure 1: eFlight model for selection of energy efficient paths. The algorithm combines Reinforcement Learning and path pre-calculation with A*.

(RL) to optimize energy consumption by identifying the most efficient flight paths. Unlike traditional A*, eFlight accounts for altitude changes, speed, and acceleration, enabling dynamic path selection. We train an RL model offline to distill a heuristic function that can be integrated into A* to find the path with the lowest energy cost, thereby reducing node expansions and runtime. Node expansion is the process of finding all available immediate next moves from the current location [15]. Minimizing node expansion allows the algorithm to reduce computation time. Compared with the traditional RL strategy, our approach finds paths with a 100% success rate in experiments with at least one available path. We compared eFlight with various baseline algorithms in simulations and found that eFlight effectively adapts to complex 3D terrains with obstacles of varying heights. The scheme also achieves high computational efficiency, which allows it to be deployed in resource-constrained embedded systems.

The remainder of this paper is organized as follows. Section 2 introduces eFlight, the proposed hybrid path-finding algorithm for 3D environments. Section 3 presents performance features of eFlight and performance comparisons with other efficient path-finding algorithms. Section 4 presents our conclusions.

2 Proposed Approach

We propose eFlight, a framework that combines the A* algorithm with reinforcement learning to enable 3D UAV path planning in obstacle cluttered urban grids. Figure 1 summarizes the overall process and the main components of the eFlight framework.

First, a 3D environment is generated, represented by a mesh and a set of obstacles, along with the drone's physical parameters and energy consumption characteristics for different flight behaviors. This information is used as input to the reinforcement learning model. The reinforcement learning loop, shown in the yellow dashed box in Figure 1, enables the agent to learn by receiving rewards and penalties through interaction with the environment. The outcome of this training process is a Deep Q-Learning (DQN) model that encodes energy-efficient navigation policies. Subsequently, the A* algorithm uses the trained DQN model as a heuristic function to guide its path search. While the original A* does not require learning, it leverages the learned heuristic to efficiently identify an energy-optimal path through the 3D space.

2.1 Energy Model and 3D environment

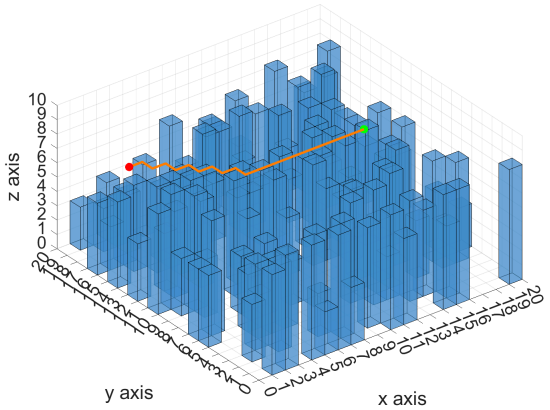

Figure 2: Simulation environment. A 3D environment with obstacles of various heights. The model represents a 20x20x10 m^3 sample space, with an obstacle density of 30% on the $x-y$ plane. Each voxel corresponds to a 1 m^3 area. The green and red markers indicate the start and end points, respectively, and the planned flight path between them is indicated in orange.

For testing, the 3D environment is modeled as a grid over a plane, where each unit square can either be empty or occupied by a portion of an obstacle, which is assigned a height. The drone's physical parameters and energy consumption model for different flight maneuvers are defined accordingly. Figure 2 illustrates a

sample 3D environment with dimensions of 20x20x10 m^3 and a grid resolution of 1x1 m^2. The obstacle density in this example is 30%, or 120 out of the 400 ground locations with obstacles. The plot provides a 3D perspective, where obstacles are shown as blue vertical bars of varying heights, resembling buildings in an urban setting or trees in an open field setting.

Physical parameters, such as drone mass, gravity, and aerodynamic drag are considered for accurate estimations. The force vector-based approach is used to calculate the energy consumption of each segment of the flight path. The goal is to verify the accuracy and effectiveness of our energy-optimized hybrid path-finding algorithm. The energy consumption model in the simulator uses gravitational and kinematics equations that describe the energy consumption on drone flight dynamics.

2.1.1 Parameters. Table 1 shows the variables of the physical parameters used to model the estimation of energy expenditure.

Table 1: Terminologies and Parameters.

Parameter	Symbol	Unit
Mass of drone	m	kg
Gravitational acceleration	g	m/s^2
Air density	ρ	kg/m^3
Drag coefficient	C_d	Dimensionless
Cross-sectional area	A	m^2
Cruise speed	v_c	m/s
Acceleration	a	m/s^2

2.1.2 Force Vector Calculations. Gravity, thrust and drag forces are defined in [2] (1), (2), and (3), respectively.

$$\vec{F_g} = -mg \tag{1}$$

$$\vec{F_t} = m\vec{a}, \quad \vec{a} = \frac{\vec{v_{\text{new}}} - \vec{v_{\text{old}}}}{\Delta t} \tag{2}$$

$$F_d = \frac{1}{2} C_d A \rho v^2, \quad \vec{F_d} = -F_d \frac{\vec{v}}{|\vec{v}|} \tag{3}$$

When hovering, $\vec{a}$ is an upward acceleration equal to $\vec{g}$.
The *Net Force* is $\vec{F_{\text{net}}} = \vec{F_g} + \vec{F_d} + \vec{F_t}$.

2.1.3 Energy Calculations. The energy required is calculated as work done by net force, hovering energy [6], drag energy [10], and acceleration energy [13] are defined in (4), (5), (6), and (7), respectively.

$$W = \vec{F_{\text{net}}} \cdot \vec{d}, \quad \vec{d} = \vec{p_{\text{new}}} - \vec{p_{\text{old}}}, \tag{4}$$

$$E_h = mgd_t, \tag{5}$$

$$E_d = F_d \cdot d, \tag{6}$$

$$E_a = \frac{1}{2} m (v_{\text{new}}^2 - v_{\text{old}}^2), \tag{7}$$

where d_t is the hovering time, as a function of distance and velocity. The energy consumed in each section of the path, or segment, is calculated: Total Energy for Segment i is then:

$$E_{s_i} = E_h + E_d + E_a \tag{8}$$

The total energy for a path consisting of N segments [5] is:

$$E_{\text{total}} = \sum_{i=0}^{N-1} E_{s_i} \tag{9}$$

The energy calculations presented here provide a foundation for directly comparing the efficiency of different flight paths. By quantifying the energy expenditure of each path, this approach enables a clear evaluation of the proposed method's performance relative to alternative algorithms. The framework ensures that path selection prioritizes minimal energy consumption without compromising traversal efficiency.

We formulate the drone path planning energy consumption as the following optimization problem:

$$\min E_{total} = \sum_{i \epsilon D \cup V} \sum_{j \epsilon V} \sum_{k \epsilon D} x_{ijk} E_{s_i} \tag{10}$$

Subject to:

$$\sum_{i \epsilon D \cup V} \sum_{k \epsilon D} x_{ijk} \leq 1, \quad \forall\, j \subseteq V \tag{11}$$

$$\sum_{i \epsilon V} \sum_{j \epsilon D} x_{ijk} = 1, \quad \forall\, k \epsilon D \tag{12}$$

$$\sum_{i \epsilon D \cup V} \sum_{j \epsilon D \cup V} d_{ij} \sum_{k \epsilon D} x_{ijk} \leq Q_k \tag{13}$$

The objective is to minimize the energy consumption of the drone by reducing the number of segments it traverses, as defined in (10). The binary decision variable $x_{ijk} \in \{0, 1\}$ indicates whether the segment between nodes i and j is selected as part of the route from starting location k. Here, D denotes the set of starting points, V denotes the set of waypoints, Q_k denotes the distance capacity from the starting point k, and d_{ij} is the distance between waypoints i and j. Constraint (11) ensures that each waypoint is visited at most once. Constraint (12) ensures that each drone returns to its depot. Constraint (13) ensures that the drone travels within its distance capacity from the starting point k.

2.2 Reinforcement Learning Method

The objective of the RL model is to learn the state–action value function, defined as:

$$h(s) = -\max_a Q\big(S(n),\, a\big) \tag{14}$$

where $h(s)$ is the heuristic function for the A*, and $\max_a Q\big(S(n),\, a\big)$ is the maximal Q-value for the current state $S(n)$. The action a defines the UAV's movement direction $(\pm x, \pm y, \pm z)$. The state S contains all information the drone uses to make decisions, such as its current and target positions in 3D space, as follows:

$$S = [\frac{x_i}{G}, \frac{y_i}{G}, \frac{z_i}{G}, \frac{x_g}{G}, \frac{y_g}{G}, \frac{z_g}{G}, r_1, ..., r_{12}] \tag{15}$$

This 18-dimensional vector is observed at each time step, where (x_i, y_i, z_i) denotes the current position and (x_g, y_g, z_g) denotes the goal position. Both are normalized by the grid dimension G. The

remaining 12 components, r_j, represent ray distances in six axis-aligned directions ($\pm x, \pm y, \pm z$) and six diagonal directions. These ray distances provide the drone with multi-directional awareness of nearby obstacles, enabling it to learn complex behaviors such as navigating narrow corridors or avoiding dead ends. Although Eqn. (14) limits the drone to six discrete actions, the inclusion of diagonal rays provides foresight, allowing the drone to detect and avoid obstacles around corners. For example, if there is an obstacle or a dead end along the path from $+x$ to $+y$, the model assigns a lower Q value to the $+x + y$ direction, thus discouraging travel in that direction.

To guide learning, we defined the step reward as follows:

$$R_t = \lambda\,\Delta d_t + R_{\text{step}}\,C_{\text{step}} + R_{\text{goal}}\,\mathbf{1}\{\text{Reach}\} + R_{\text{coll}}\,\mathbf{1}\{\text{Collision}\}, \quad (16)$$

with the progress term:

$$\Delta d_t = \frac{d(p_t, g) - d(p_{t+1}, g)}{D}. \quad (17)$$

R_t is the immediate reward at time t, λ is the weight of the progress term; The indicator functions $\mathbf{1}\{\text{Reach}\}$ and $\mathbf{1}\{\text{Collision}\}$ take the value of 1 when the event occurs, and take the value of 0 otherwise. R_{goal}, R_{step}, and R_{coll} is the different behavior reward; Table 2 lists the reward setting and description used in this study; p_t and p_{t+1} are the agent's positions before and after the action, and g is the goal. The function $d(p, g)$ denotes the distance metric. The normalization constant D is set to the diagonal of the space such that $\Delta d_t \in [-1, 1]$, where positive values indicate a travel towards the target.

Table 2: Parameters for the reward function

Parameter	Value	Description
R_{goal}	50	Terminal bonus for reaching the target destination
R_{coll}	-5.0	A penalty for moving into an obstacle or out of bounds
R_{step}	-0.01	Per-step cost to encourage efficiency

To address this issue, the learning algorithm is enhanced with Hindsight Experience Replay (HER) [3]. Whenever a training episode fails to reach the desired goal, HER retroactively relabels the trajectory by treating the actual final state reached as the goal. This strategy of redefining "failures" as virtual successes generates a dense and informative training signal and significantly reduces the number of episodes required to learn an effective policy.

After training, the learned Q-function $Q\big(S(n), a\big)$ serves as the primary output of the model. First, it can be directly deployed as a greedy navigation policy, where the agent chooses the action with the highest Q-value at each step. More importantly, in the eFlight framework, the Q-function acts as a learned heuristic for A* search. By prioritizing actions with higher Q-values during node expansion, it focuses the search on energy-efficient paths while significantly reducing computational overhead.

eFlight searches for the lowest-energy cost path by evaluating each node n with

$$E_{eFlight}(n) = g(n) + h(n) = \min E_{total} + -\max_a Q\big(S(n),\ a\big) \quad (18)$$

The path cost $g(n)$ is the accumulated flight energy from the start to the current position n, which is the sum of the hovering, aerodynamic drag, and acceleration energy terms. The heuristic $h(n)$ is based on reinforcement learning. We use the negative of the state–action value $S(n)$ as an estimate of the action and energy amount.

The search begins by inserting the start node into a set, OpenSet, with $g = 0$ and another empty set, ClosedSet. Iteratively, the algorithm selects the node from the OpenSet with the lowest f-value, moves it to the ClosedSet, and evaluates its neighbors. For each neighbor, it calculates the tentative cumulative cost from the start; if this path is more efficient than any previously recorded one, the algorithm updates that neighbor's g and f scores and sets its predecessor to the current node. Any newly discovered neighbor is added to the OpenSet for future consideration. This process continues until the destination node is selected, at which point the optimal path is reconstructed by following the chain of predecessors back to the start. If the OpenSet becomes empty before the goal is reached, the algorithm concludes that there is no valid path.

3 Experimental Results

We implemented eFlight in Python and tested it under four 3D test environments with dimensions 50x50x10 m^3 and 200x200x10 m^3 with 30% and 50% obstacle densities, respectively, through simulations. Obstacle heights were randomly set using a uniform distribution in the range of 3 to 8 meters.

We compared eFlight against three baselines: the direct, rise and traverse, and A* schemes. The direct method assumes that the drone flies in a straight line from the starting point to the target, ignoring all obstacles. It serves as the theoretical lower bound for energy consumption. The rise and traverse method first guides the drone to ascend vertically above the obstacle field, followed by a horizontal A* search at that safe altitude. The A* approach is the classical A* implementation that finds the shortest-length path through a 3D grid.

For each environment, we tested 1,000 trials. The starting and ending points of each trial were randomized to ensure a comprehensive evaluation across a variety of path-finding scenarios. The results are shown in Figure 3, organized into a 4x3 grid, where each row corresponds to one of the four environments and each column presents a different analysis. In Figures 3(a), (d), (g), and (j), we compare each method to the direct scheme baseline (i.e., $y = x$), which represents the minimum energy path when using the same start and end points. A* performs comparably to the baseline algorithm in small and sparse areas. The rise and traverse scheme also reduces the number of detours. However, as map size and obstacle density increase, eFlight's learned heuristic performs closest to the straight-line energy lower bound, and its advantage becomes more pronounced in denser environments.

Figures 3 (b), (e), (h), and (k) show the relationship between mean energy (+/- one standard deviation) and path length. While

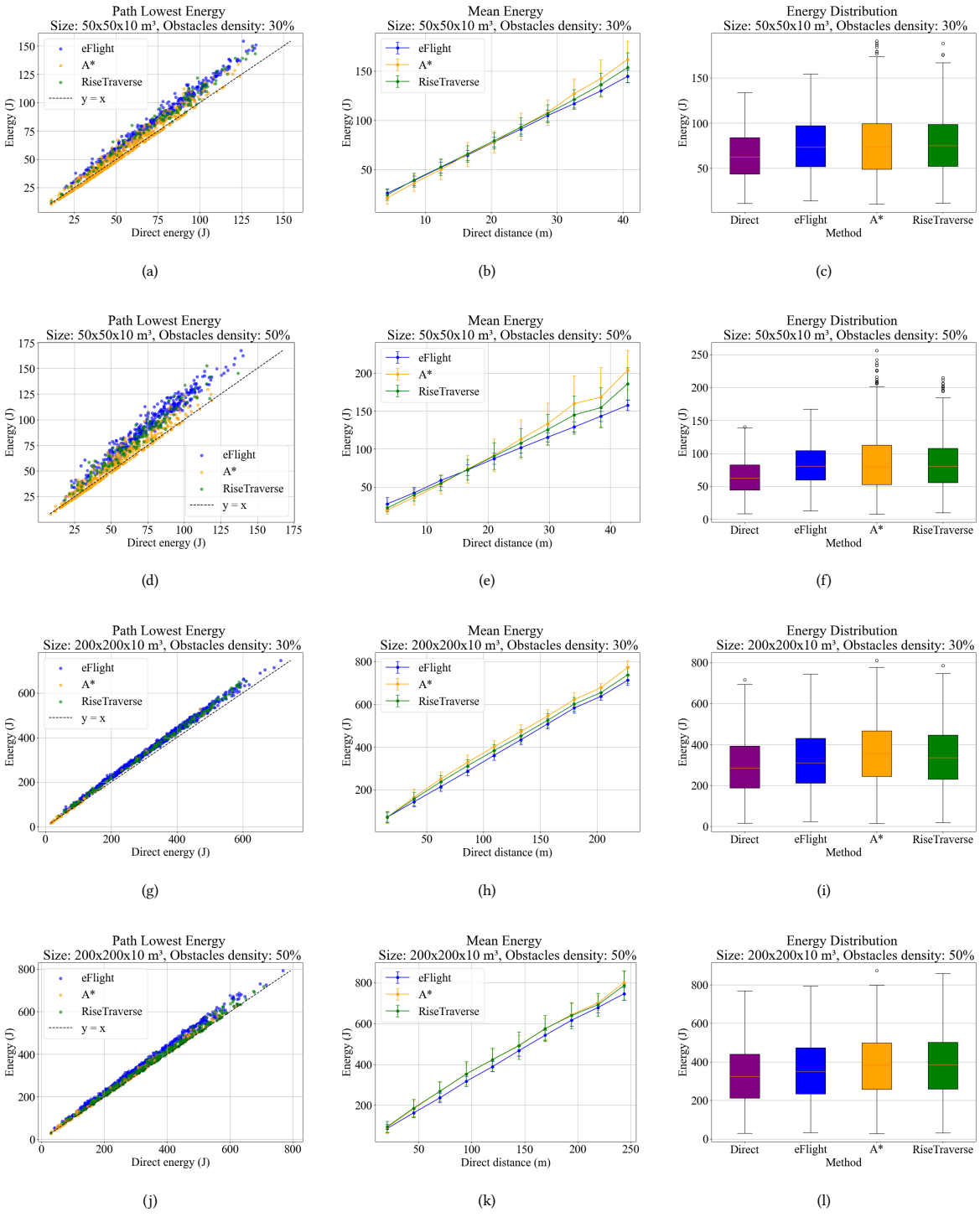

Figure 3: Comparative performance of UAV path-planning methods. Each row is a unique environment, from top to bottom: 50x50x10 m^3 with 30% obstacle density, 50x50x10 m^3 with 50% obstacle density, 200x200x10 m^3 with 30% obstacle density, and 200x200x10 m^3 with 50% obstacle density.

Table 3: Benchmark of path-finding methods under 200x200x10 m^3 with 50% obstacle density environment

Method	OPT (%))	Time Reduction (%)	Node Reduction (%)	Energy Reduction (%)
A*	2.4%	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Rise Traverse	8.1%	29.1 ± 40.0	35.4 ± 44.3	-0.62 ± 10.72
eFlight	89.5 %	90.6 ± 26.6	68.2 ± 45.6	7.13± 9.96

the compared methods scale mostly linearly with distance, eFlight consistently achieves the lowest mean energy and with the smallest errors, indicating more stable and predictable performance across a variety of distances and obstacle configurations. Figures 3 (c), (f), (i), and (l) display box plots of energy distributions. eFlight exhibits the lowest median energy, smallest inter-quartile range, and fewest extreme outliers, confirming it as the most robust and energy-efficient choice for UAV missions in both small and large, sparse and cluttered environments.

In addition to its exceptional energy efficiency, eFlight also offers a key advantage in computational performance. The conventional A* method is often too slow for dynamic applications, as its exhaustive graph search evaluates a large number of nodes. To evaluate these, we use the A* as the baseline, with the following three evaluation metrics:

- OPT (Optimality / Minimum-Energy Rate): The proportion of methods that achieve the minimum total energy among all methods
- Time Reduction: The average time reduction relative to A*, which is expressed as mean ± standard deviation
- Node Reduction: The average reduction in node expansion times relative to A* is reported as mean ± standard deviation.
- Energy Reduction: The energy reduction relative to the A* is reported as mean ± standard deviation.

Table 3 lists the results. Using the A* as the baseline, eFlight has the best performance. It attains the highest OPT and achieves the largest average reductions in both runtime and node expansions. Rise Traverse offers modest improvements but with substantial variability across cases. Overall, eFlight is the preferred choice for energy-optimal path finding, combining near-optimal energy performance with significant efficiency gains and rapid solution times, making it suitable for deployment in resource-constrained embedded systems.

4 Conclusion

In this paper, we propose eFlight, a hybrid algorithm for energy-efficient drone path planning. It is integrated with a trained reinforcement learning heuristic that uses A* search to achieve low-energy path finding and reduce computation time compared to conventional path-finding algorithms. We compare eFlight against three baselines; the A*, the Rise and Traverse, and a direct path algorithms, across varying size areas and for various obstacle densities. The results show that eFlight not only finds energy-efficient paths more reliably than other obstacle-aware methods, but also computes them faster. In the most complex environment, eFlight returns the lowest-energy path in 89.5% of the trials. Relative to the baseline algorithm, eFlight achieves 90.6 ± 26.6% reduction in computation time, 68.2 ± 45.6% reduction in node expansion, and 7.13% ± 9.96% energy reduction (mean ± standard deviation), highlighting its promise for real-time, resource-constrained UAV deployment.

5 Acknowledgments

This work was partially supported by the US National Science Foundation under Grants No. 1841558 and 1856032 and NJIT Faculty Seed grant.

References

[1] Shubhani Aggarwal and Neeraj Kumar. 2020. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer communications* 149 (2020), 270–299.
[2] J. D. Anderson. 2010. *Introduction to Flight* (7th ed.). McGraw-Hill, New York, NY, USA.
[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in neural information processing systems* 30 (2017).
[4] Rashiya Bekmurzaeva, Rustem Kalimullin, and Fatima Aguzarova. 2024. Application of drones and artificial intelligence to monitor and protect natural ecosystems. In *BIO Web of Conferences*, Vol. 140. EDP Sciences, 01008.
[5] Thor I. Fossen. 2011. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley.
[6] David Halliday, Robert Resnick, and Jearl Walker. 2013. *Fundamentals of Physics* (10th ed.). Wiley.
[7] Jordan Leyva, Nahim Moran, Christopher Romero, Adrien Durasno, Tendai Chimuka, Ziqian Dong, and Roberto Rojas-Cessa. 2025. EcoFlight: Low-Energy Path Finding Through Obstacles for Autonomous Sensing Drones. In *IEEE MIT Undergraduate Research Technology Conference*.
[8] Fangyu Li, Sisi Zlatanova, Martijn Koopman, Xueying Bai, and Abdoulaye Diakité. 2018. Universal path planning for an indoor drone. *Automation in Construction* 95 (2018), 275–283.
[9] Jingya Liu, Roberto Rojas-Cessa, and Ziqian Dong. 2016. Sensing, calculating, and disseminating evacuating routes during an indoor fire using a sensor and diffusion network. In *2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC)*. IEEE, 1–6.
[10] Bruce R. Munson, Donald F. Young, and Theodore H. Okiishi. 2013. *Fundamentals of Fluid Mechanics* (7th ed.). Wiley.
[11] Kun Shen, Rutuja Shivgan, Jorge Medina, Ziqian Dong, and Roberto Rojas-Cessa. 2022. Multidepot drone path planning with collision avoidance. *IEEE Internet of Things Journal* 9, 17 (2022), 16297–16307.
[12] Rutuja Shivgan and Ziqian Dong. 2020. Energy-efficient drone coverage path planning using genetic algorithm. In *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 1–6.
[13] Paul A. Tipler and Gene Mosca. 2007. *Physics for Scientists and Engineers* (6th ed.). W. H. Freeman.
[14] Yu Wu and Kin Huat Low. 2020. An adaptive path replanning method for coordinated operations of drone in dynamic urban environments. *IEEE Systems Journal* 15, 3 (2020), 4600–4611.
[15] Ryo Yonetani, Tatsunori Taniai, Mohammadamin Barekatain, Mai Nishimura, and Asako Kanezaki. 2021. Path planning using neural a* search. In *International conference on machine learning*. PMLR, 12029–12039.
[16] Yong Zeng, Rui Zhang, and Teng Joon Lim. 2016. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Communications magazine* 54, 5 (2016), 36–42.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009