

Real-Time Drone Data Processing for Aerial Photography Using Edge Computing and AI-Powered Algorithms

Nuraj Indika Rajapakse^{ORCID}, Ismail Ladele Ladipo*

Faculty of Engineering, Higher Colleges of Technology, Sharjah, United Arab Emirates

Email: *iladipo@hct.ac.ae

How to cite this paper: Rajapakse, N.I. and Ladipo, I.L. (2026) Real-Time Drone Data Processing for Aerial Photography Using Edge Computing and AI-Powered Algorithms. *Advances in Internet of Things*, 16, 124-142.

<https://doi.org/10.4236/ait.2026.164008>

Received: March 14, 2026

Accepted: October 24, 2026

Published: October 27, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

The rapid advancement of Unmanned Aerial Vehicles (UAVs) has made the field of aerial photography a dynamic, data-driven environment, requiring real-time data analysis and intelligent decision-making. However, the traditional cloud computing paradigm for processing UAV data has been found to be prone to latency, bandwidth constraints, and connectivity issues. This research is presented to address the problems in the traditional cloud computing paradigm for processing UAV data at a conceptual level, and it focuses on the integration of edge computing and AI-based algorithms for real-time data processing. This paper presents the use of edge computing and AI for real-time UAV data processing in aerial photography under obstacle constraints, emphasizing edge-AI, MDP and Q-learning concepts while providing a comparative study in evaluating various field-related examples. In the proposed framework, the intelligent decision-making for the UAV is achieved through the deployment of lightweight neural networks and computer vision models completely onboard or near-edge devices. This kind of system enables immediate object detection, scene classification, and adaptive flight control without entirely relying on continuous cloud connectivity. The proposed framework indicates the potential for improvements in processing speed, decision latency, and energy efficiency based on theoretical analysis and system design considerations. However, these benefits are not empirically validated in this study and are left for future experimental evaluation. The primary objective of this paper is to develop a unified edge-AI framework that enables real-time data processing and autonomous decision-making for UAVs operating in obstacle-constrained environments.

Keywords

Surveillance, Autonomous Area Mapping, Cloud & Edge Computing, Edge

1. Introduction

Surveillance, mapping, and inspection are some of the main areas of aerial photography that constantly utilize modern technological advancements to ensure increased accuracy as well as speed. Two main distributed computing models currently in use in this field include cloud computing as well as edge computing. The traditional cloud computing model mainly relies on the use of remote data centers for the purpose of storing as well as processing the data. However, the use of edge computing for the purpose of data processing allows for the creation of a faster, more dependable, as well as responsive environment. The use of edge computing enables the minimization of the degradation of performance caused by the latency of the network, thus allowing for the improvement of the real-time responsiveness of the system. This is because the use of edge computing allows the data to be processed in the devices or servers that are located at the edge of the network, thus eliminating the need for the data to be sent to the cloud servers for the purpose of analysis.

In addition, the inclusion of Artificial Intelligence (AI) in aerial photography has significantly enhanced the efficiency and accuracy of data processing, particularly when dealing with large quantities of high-resolution images. AI-based imaging technologies can identify, classify, and segment objects within aerial photographs, facilitating faster data interpretability even when computational resources are scarce. Compared to conventional manual data analysis, AI-based processing provides higher accuracy, reduced human error, and increased efficiency.

This paper is primarily focused on the use of the technology known as Edge Computing and Artificial Intelligence algorithms, collectively known as Edge AI, for the real-time processing of data in the context of aerial photography. The use of such technologies enables the real-time processing of data, reducing the energy required for the same. The proposed framework includes the use of intelligent cameras with IoT sensors, which enables the real-time collection of data. The IoT-enabled camera makes use of embedded AI algorithms that are capable of analyzing the collected data, making decisions, and even executing the imaging process autonomously. IoT technologies are generally known for collecting sensor data and providing continuous connectivity for remote monitoring. The use of AI enables the IoT technology to effectively process the collected data.

Furthermore, the role of effective path planning requires fast and accurate aerial imaging for the optimizing of flight trajectories while avoiding obstacles. A well-established approach in machine learning, such as the Markov Decision Process (MDP), provides a powerful framework for optimizing the path planning of drones in dynamic environments. By combining Edge AI with path planning algorithms based on MDP, the efficiency of aerial photography systems can be significantly

enhanced because of the fact that intelligent decisions can be made by the drone during navigation. This paper aims to discuss how the formulation of MDP for drone mapping can be practically achieved, including the definition of states, actions, and reward functions, and the possible solution approaches for various scenarios, including some cases that are modeled as a Partially Observable Markov Decision Process (POMDP) for real-world uncertainty handling.

2. Literature Review

The recent developments in Unmanned Aerial Vehicles (UAVs) have been very instrumental in improving the capabilities of aerial photography. However, some limitations have been noted with the conventional use of clouds as a means of processing data. To reduce latency, bandwidth dependency, and scalability issues that are inherent with conventional means of processing data from aerial photography, researchers have been motivated to use edge computing with Artificial Intelligence (AI) as a means of processing visual data directly from the source [1] [2]. Moreover, a real-time UAV object detection framework that combines Vision Transformer (ViT) architecture and edge computing has been presented, directly addressing real-time aerial image processing and edge AI deployment [3].

The literature has identified that there are three major edge processing architectures that have been dominant with regard to processing data from drones. They are Onboard edge AI processing, where data processing takes place onboard of UAV; Near-edge or Micro-Edge processing, where processing takes place near the unmanned aerial vehicle; and a hybrid model that involves both clouds and edge servers [1] [4].

A major line of research focuses on tailoring AI algorithms for edge devices with hardware constraints. That kind of research work has utilized model compression, quantization, and pruning techniques to implement deep learning algorithms within hardware constraints of UAV devices [2] [5]. Lightweight deep learning models such as MobileNet, YOLOv5-Nano, and CSPDNet are used for object detection, scene classification, and semantic segmentation tasks. Additionally, AI-based techniques for dynamic exposure compensation, image stabilization, and framing have been proposed to improve the quality of images captured by UAVs [5].

UAV edge computing is typically carried out on hardware such as NVIDIA Jetson Nano/Xavier, Google Coral TPU, and Intel Movidius [6]. Such hardware allows for real-time processing at 15 - 30 frames per second for medium-resolution images. Middleware such as ROS and ROS2 are commonly used to integrate AI with flight control systems. Researchers have shown that such hardware-software stacks can be used to effectively manage data processing for UAVs [4] [6]. Further, a system in processing a real-time aerial image under environmental noise to address AI robustness in drone imagery has been proposed to improve accuracy and robustness compared to standard models, which provide interpretable outputs [7].

The evaluation metrics for this field are usually processing latency (ms/frame),

energy efficiency (W or mAh), and model accuracy (mAP). Datasets such as Vis-Drone and UAVDT are used for benchmarking object detection and tracking for drones in dynamic flight conditions [5]. These datasets help to quantify the trade-offs between computation efficiency and visual quality for edge-based drones.

Edge AI enables drones to conduct dynamic and adaptive aerial photography, where drones can change their exposure, position of the gimbal, and framing based on their understanding of the scene or tracking of subjects of interest. Various other applications of UAVs with edge AI include environmental, agricultural, and infrastructure inspection, where drones can conduct analysis to support field-level decision-making [4] [6]. Related precision-agriculture applications are discussed in [8]-[12]. The use of edge AI with drones enables UAVs to be dynamically optimized rather than becoming simple data collectors.

Despite these encouraging achievements, there are also a few limitations. For instance, energy constraints and thermal issues persist in the prolonged real-time processing of UAVs onboard [1]. Moreover, AI models trained on standard datasets may not generalize well under different lighting conditions, altitudes, and weather conditions. Communication issues also persist, especially when UAVs depend on unstable 4G/5G connections for hybrid edge-cloud computing [2]. These issues indicate that there is a strong need for more robust, adaptive, and energy-efficient AI systems and frameworks.

Some of the emerging trends in UAV-based AI research include the application of tiny AI and NAS (Neural Architecture Search) for designing AI models that can specifically operate on UAVs. Additionally, there is also a growing interest in FL (Federated Learning), which enables UAVs to learn collaboratively without data transfer [2]. Furthermore, the application of MEC (Multi-access Edge Computing) and 5G connectivity for enabling ultra-low latency processing of large-scale UAVs also represents an emerging trend [4]. These issues indicate that the next step in UAV-based AI research involves the application of hardware-aware AI and context-driven data processing. To address the identified gaps, this study adopts reinforcement learning, specifically Q-learning, as a decision-making framework.

3. Core Methods for Real-Time Data Processing

3.1. Onboard Data Acquisition and Pre-Processing

One of the most significant techniques for processing data in real-time for drones is data acquisition and pre-processing. While drones take clear images and videos, data is immediately processed to eliminate lens distortion, reduce noise, and normalize lighting conditions. Pre-processing data is significant in reducing the complexity of processing data in subsequent stages. It is vital to ensure that data is processed correctly to ensure accurate analysis. With data pre-processing, drones are able to process data in real-time without having to maintain communication with other devices to achieve this task.

3.2. Onboard Artificial Intelligence and Computer Vision Inference

The use of artificial intelligence and computer vision inference is another important approach for data processing for aerial photography. In this approach, deep learning models are optimized and embedded in edge processors, which are then used for object detection, object tracking, feature recognition, and scene segmentation in real-time. This is a very speedy process in analyzing images and scenes, which in turn helps in intelligent decision-making, properly adjusting camera angles and fast-tracking objects while navigating autonomously. This immensely helps in reducing latency, which is important in dynamic scenarios like aerial photography.

3.3. Region-of-Interest (ROI) Extraction and Event-Driven Processing

Region of interest, as well as event-driven processing, can be used to improve the efficiency of real-time systems. In traditional real-time systems, the entire frame is constantly being analyzed. In edge computing, the analysis is done only on the region or event of interest. This improves the energy efficiency of the system while still providing high accuracy in the analysis. This means that the drones will be able to fly for a longer time without compromising real-time performance.

3.4. Edge-Cloud Collaboration and Task Offloading

Finally, edge-cloud collaboration and task offloading also have a significant role in the scalability of the real-time processing of drone data. Real-time tasks, like vision inference and navigation, are processed in the edge, while less urgent tasks, like high-resolution mapping, are offloaded to edge servers/cloud platforms, depending on the connectivity status. Resource-aware scheduling algorithms are also utilized to determine the execution location of tasks in a distributed manner, depending on the constraints and requirements of the tasks and the resources. This ensures a balanced and efficient processing paradigm for intelligent, autonomous, and real-time aerial photography systems.

4. AI-Driven Approaches to Real-Time Drone DATA Processing for Aerial Photography

4.1. The Markov Decision Process

The formal MDP is a collection of items of tuple $\langle S, A, P, R, \gamma \rangle$, where S = State Space, A = Action Space, P = State Transition Probability, R = Reward Function and $\gamma \in [0, 1)$ = Discount Factor.

The objective is to find policy $(a | s)$ to choose actions in maximizing expected discounted returns.

States: S

State Transition Probability: $P(s' | s, a) = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$

Reward Function: $R(s, a, s')$

Policy: $\pi(a|s) = P(A_t = a | S_t = s)$

Return (Total Discounted Reward): $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

State Value Function: $V^{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right]$

where,

s —Current state;

s' —Next state;

a —Action taken;

a' —Possible next action.

Q-Learning Formulation for Autonomous Area Mapping

Below is the Q-learning formulation (Action Value Function) of the autonomous area mapping MDP. The Q-value represents expected future reward.

Action Value Function (Q-function):

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]$$

Q-Learning Update Rule (Core Equation from Bellman Equation)

The quality of an action is given by the Q-learning update rule, and the update rule adjusts the old value as follows:

New Q-value = Old Q-value + Learning Rate $\times$ Temporal Difference Error

$$\text{Temporal Difference (TD) Error} = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$$

$$Q_{new}(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

$$Q_{new}(s, a) = (1 - \alpha) Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') \right)$$

where,

r —Reward received after taking action (actual reward);

α —Learning rate ($0 < \alpha \leq 1$);

γ —Discount factor ($0 \leq \gamma < 1$);

$Q(s, a)$ —Current estimate of value;

$\max_{a'} Q(s', a')$ —Maximum Q-value for the next state;

$\gamma \max_{a'} Q(s', a')$ —Best future reward.

The Q-learning update equation is used to learn the optimal action-value function $Q(s, a)$ in a Reinforcement Learning (RL) environment without knowing the model (transition probabilities or rewards). The above equation moves the current Q-value toward a better estimate, and it directs the agent in the process of updating its knowledge after each action.

Further, TD error is the difference between what you expected to happen and what actually happened and also the addition of the reward received after taking action. It can also be viewed as the learning signal in reinforcement learning. The positive TD error indicates that the outcome is better than expected and the negative TD error indicates that the outcome is worse than expected. Moreover, zero TD error indicates that the prediction mostly is perfect.

4.2. Partially Observable Markov Decision Process

It is worth mentioning that the Partially Observable Markov Decision Process (POMDP) is an extension of an MDP, which addresses the uncertainty of the agent's perception of the state. This happens when the coverage depends on the entire history. Therefore, by incorporating the mapped state clearly in the state space ($\mathcal{S}$), the process of aerial photography can be represented in the MDP. It's proposed to formalize the UAV mission as a sequential decision-making problem under uncertainty. Depending on sensing completeness, the problem can be modeled as either a MDP or a POMDP. In realistic deployments, the UAV does not have full access to the true state and therefore it's recommended to model the problem as a POMDP. Moreover, the MDP formulation works well when the state variables can be accurately estimated and known, whereas the POMDP formulation is necessary when sensing is incomplete and noisy, requiring reasoning over uncertainty via the observation model, justifying the use of a POMDP in realistic UAV deployments.

As depicted in **Figure 1**, a discrete MDP defines state and action spaces as finite or countable sets, making it suitable for problems where variables like position, altitude and battery charge level can be reasonably discretized into grids or levels. Such a definition allows for the application of exact dynamic programming techniques like value iteration and policy iteration, providing convergence guarantees and a more computationally friendly problem definition. By contrast, a continuous MDP defines state and action spaces as real-valued, which is necessary when fine-grained control, precise dynamics, or safety constraints cannot be well-approximated using discretization. Continuous MDPs often require approximate solution techniques like policy gradients and MPC (Model Predictive Control), which, although more difficult to analyze, can provide more scalable and realistic models. In fact, a common approach in practice is to use a combination of a discrete MDP for high-level planning and a continuous control approach for low-level control.

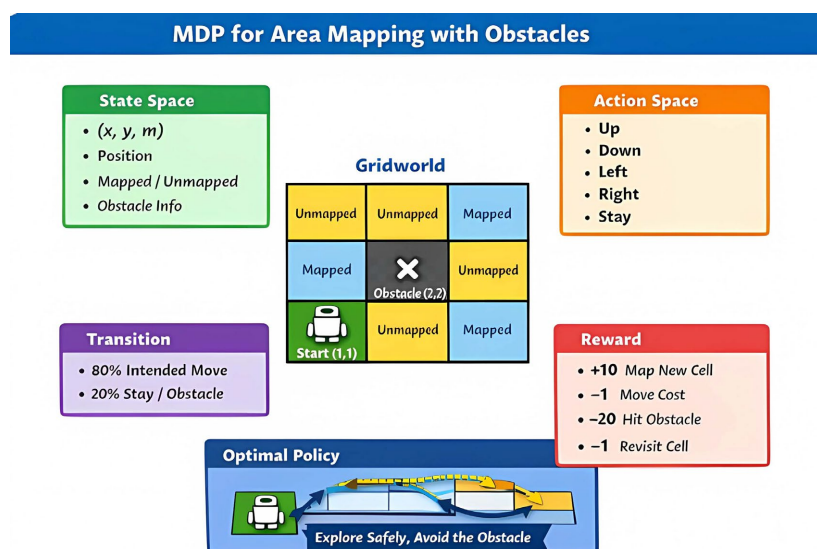


Figure 1. MDP for area mapping with obstacles.

5. A Collection of Items for a Small Grid-World Numerical Model of Autonomous Area Mapping

State Space, $S = (x, y, z, b, m)$

where:

x, y —Position (x is a row position and y is a column position);

z —Altitude;

b —Battery level;

m —Mapped/coverage state.

1) Action Space, $A: a \in \{\text{move, change altitude, return}\};$

2) For the discrete model, $a \in \{\text{N/E/S/W/up/down}\};$

3) For the continuous model, $a \in \{\text{velocity, climb rate}\};$

4) Transition Model $P(s' | s, a)$ converts information such as motion dynamics, wind disturbance, sensor & actuation uncertainty, etc. Therefore, the next state probability is modeled as the addition of the system dynamics and also the noise.

$$P(s' | s, a) = \text{Dynamics}(s, a) + \epsilon$$

where,

$\text{Dynamics}(s, a)$ —Deterministic system model predicting the next state;

ϵ —Small disturbance or the noise term.

Therefore, the optimal Policy π^* produces an efficient path under uncertainty as below:

$$\pi^*(s) = \arg \max_a \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R_t \right]$$

Reward function or the total reward, $R(s, a, s')$ is the weighted combination of several reward components such as coverage, quality, energy and time.

$$R = w_c \cdot \text{Coverage} + w_q \cdot \text{Quality} - w_e \cdot \text{Energy} - w_t \cdot \text{Time}$$

6. Practical Solution Approaches of Fully Numerical Grid-World MDP for Autonomous Area Mapping

It's important that a drone must map an unknown area which is represented as a grid. The objective is to maximize the mapped area (information) while avoiding collisions and returning to base before battery runs out.

Case 1: Autonomous area mapping for a 2×2 grid with Q-learning in absence of an obstacle

Grid Definition

Index cells row-wise (Vector size = 4):

0 1

2 3

So: Mapped state, $M \in \{0, 1\}^4$ (For each entry, explored is taken as one (1) and unexplored is taken as zero (0)).

State Definition

$s = (p, M)$ —total of 64 states and $|S| = 4 \times 2^4 = 64$

where:

- $p \in \{0, 1, 2, 3\}$;
- $M \in \{0, 1\}^4$.

Action Space Table

Action	Code
Up	0
Down	1
Left	2
Right	3

Reward Function

$$R(s, a, s') = \begin{cases} +5 & \text{new cell explored} \\ -1 & \text{visited cell} \\ -5 & \text{invalid move (wall)} \end{cases}$$

Initial State—start at cell 0:

$$p_0 = 0$$

$$M_0 = [1, 0, 0, 0]$$

Transition Steps

Step 1: Action: Right (3)

$0 \rightarrow 1$ —valid move

$$p_1 = 1$$

$$M_1 = [1, 1, 0, 0]$$

Reward = +5

Step 2: Action: Down (1)

$1 \rightarrow 3$ —valid move

$$p_2 = 3$$

$$M_2 = [1, 1, 0, 1]$$

Reward = +5

Step 3: Action: Left (2)

$3 \rightarrow 2$ —valid move

$$p_3 = 2$$

$$M_3 = [1, 1, 1, 1]$$

Reward = +5

It's noted that all cells are now explored.

Q-Learning Update Equation

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Step 1: From state 0, the agent has taken an action “Right” (action code is 3).

Current state, $s = 0$ and action taken, $a = 3$ (Right). From state 0, moving Right leads to the next state, $s' = 1$. Since state 1 is new, $r = +5$.

Since no learning has happened yet, it's reasonable to consider current Q-value at the start as $Q(0,3)=0$ and the maximum future Q-value from next state (1) as $\max Q(1,a)=0$.

Also consider the learning parameters as follows:

Learning rate: $\alpha = 0.5$;

Discount factor: $\gamma = 0.9$.

Q-learning update for step 1:

$$Q(0,3) \leftarrow Q(0,3) + \alpha [r + \gamma \max Q(1,a) - Q(0,3)]$$

Substituting values:

$$Q(0,3) = 0 + 0.5(5 + 0.9(0) - 0)Q(0,3) = 2.5$$

State	Up (0)	Down (1)	Left (2)	Right (3)
0	0	0	0	2.5
1	0	0	0	0
3	0	0	0	0
2	0	0	0	0

Partially updated Q-table for changing status is:

It's important to note that the agent has learnt, going right from state 0 is a better choice and also future selection of actions prefers going right. This way of learning is driven by the positive reward. Moreover, a negative TD error indicates that the action was worse than expected and the high Q-value indicates that the direction of movement is in the right direction.

Step 2: From state 1, the agent has taken an action "Down" (action code is 1).

Current state, $s = 1$ and action taken, $a = 1$ ("Down").

From state 1, moving down leads to the next state, $s' = 3$.

Since state 3 is new, $r = +5$.

Also consider current Q-value at the start, $Q(1,1)$ is zero (0).

The maximum future Q-value from next state (3), $\max Q(3,a)$ can be assumed as 0 since no learning has happened yet.

Q-learning update for step 2:

$$Q(1,1) \leftarrow Q(1,1) + \alpha [r + \gamma \max Q(3,a) - Q(1,1)]$$

Substituting values:

$$Q(1,1) = 0 + 0.5(5 + 0.9(0) - 0)Q(1,1) = 2.5$$

State	Up (0)	Down (1)	Left (2)	Right (3)
0	0	0	0	2.5
1	0	2.5	0	0
3	0	0	0	0
2	0	0	0	0

Partially updated Q-table

Step 3: From state 3, the agent has taken an action “Left” (action code is 2).

Current state, $s = 3$ and action taken, $a = 2$ (“Left”).

From state 3, moving left leads to the next state, $s' = 2$.

Since state 2 is new, $r = +5$.

Also consider current Q-value at the start as $Q(3, 2) = 0$.

The maximum future Q-value from next state (2), $\max Q(2, a)$ can be assumed as 0 since no learning has happened yet.

Q-learning update for step 3:

$$Q(3, 2) \leftarrow Q(3, 2) + \alpha [r + \gamma \max Q(2, a) - Q(3, 2)]$$

Substituting values:

$$Q(3, 2) = 0 + 0.5(5 + 0.9(0) - 0)Q(3, 2) = 2.5$$

State	Up (0)	Down (1)	Left (2)	Right (3)
0	0	0	0	2.5
1	0	2.5	0	0
3	0	0	2.5	0
2	0	0	0	0

Partially updated Q-table for the above state changes

Case 2: Autonomous area mapping for the same 2×2 grid with Q-learning in the presence of an obstacle

Index cells row-wise from 0 to 3 (Vector size = 4):

Let **cell 1** be an obstacle so that the robot cannot enter cell 1.

0 X

2 3

where:

- $p \in \{0, 2, 3\}$ (cell 1 is forbidden);
- $M \in \{0, 1\}^4$.

Updated Reward Function

$$R(s, a, s') = \begin{cases} +5 & \text{new cell explored} \\ -1 & \text{revisiting a cell} \\ -10 & \text{hit obstacle or wall} \end{cases}$$

Initial State—start at cell 0:

$$p_0 = 0$$

$$M_0 = [1, 0, 0, 0]$$

Valid Transitions from cell 0:

Right $\rightarrow$ obstacle $\rightarrow$ stay in 0, down $\rightarrow$ cell 2, left $\rightarrow$ wall and up $\rightarrow$ wall

Transition Steps

Step 1: Action: Right (3)

0 $\rightarrow$ 1—hit obstacle (blocked), therefore the next state = 0 (no movement)

$p_1 = 0$
 $M_1 = [1, 0, 0, 0]$
 Reward = -10
 Step 2: Action: Down (1)
 $0 \rightarrow 2$ —valid move
 $p_2 = 2$
 $M_2 = [1, 0, 1, 0]$
 Reward = +5
 Step 3: Action: Right (3)
 $2 \rightarrow 3$ —valid move
 $p_3 = 3$
 $M_3 = [1, 0, 1, 1]$
 Reward = +5
 Step 4: Action: Up (0)
 $3 \rightarrow 1$ —blocked (obstacle)
 $p_4 = 3$
 $M_4 = [1, 0, 1, 1]$
 Reward = -10
 Map after exploration:

```

+-----+-----+
X
+-----+-----+
+-----+-----+

```

Q-Learning update (numerical)

Assume:

$\alpha = 0.5$, $\gamma = 0.9$, $Q(s, a) = 0$ and $\max Q(s', a') = 0$ since no learning has happened yet.

Q-learning update for step 1—Action: Right (3):

$$Q(0, 3) \leftarrow Q(0, 3) + \alpha [r + \gamma \max_a Q(0, a) - Q(0, 3)]$$

$$Q(0, 3) = 0 + 0.5 [-10 + 0.9(0)] = -5$$

Q-learning update for step 2—Action: Down (1):

$$Q(0, 3) \leftarrow Q(0, 3) + \alpha [r + \gamma \max_a Q(0, a) - Q(0, 3)]$$

$$Q(0, 1) = 0 + 0.5 [5 + 0.9(0)] = 2.5$$

Q-learning update for step 3—Action: Right (3):

$$Q(2, 3) \leftarrow Q(2, 3) + \alpha [r + \gamma \max_a Q(3, a) - Q(2, 3)]$$

$$Q(2, 3) = 0 + 0.5 [5 + 0.9(0)] = 2.5$$

Q-learning update for step 4—Action: Up (0):

$$Q(3, 0) \leftarrow Q(3, 0) + \alpha [r + \gamma \max_a Q(3, a) - Q(3, 0)]$$

$$Q(3, 0) = 0 + 0.5 [-10 + 0.9(0)] = -5$$

Q-learning update for step 5—Action: Left (2):

$$Q(3,2) \leftarrow Q(3,2) + \alpha [r + \gamma \max_a Q(2,a) - Q(3,2)]$$

$$Q(3,2) = 0 + 0.5[-1 + 0.9(0)] = -0.5$$

State	Up (0)	Down (1)	Left (2)	Right (3)
0	0	2.5	0	-5
1	-	-	-	-
2	0	0	0	2.5
3	-5	0	-0.5	0

Updated Q-table

State Space Size is given by the multiplication of reachable positions (3) and mapped states (16).

State Space Size, $|S| = 3 \times 2^4 = 48$.

It's understood that the agent has strongly avoided walls due to obstacle penalty and also the looping has prevented due to revisit penalty. In here, the exploration is encouraged, but it's finite to a great extent.

Case 3: Autonomous area mapping for 5 × 5 grid with Q-learning in the presence of obstacles

Grid Definition

Index cells row-wise (vector size = 25): (index 0 - 24)

0 1 2 3 4

5 6 7 8 9

10 11 12 13 14

15 16 17 18 19

20 21 22 23 24

Let **cells 7, 13, and 21** be obstacles (3 obstacle cells), so that the robot cannot enter those cells. A penalty should be imposed if the robot attempts those cells. It's also counted as an obstacle if action moves outside the grid and, as a result, the agent stays in same state.

$S = \{(x, y), M \in \{0, 1\}^{25}\}$ — Robot position is given by (x, y) and the mapped state by M

where:

(x, y) — Robot position (x is a row position and y is a column position);

M — Mapped/coverage state.

$$A = \{\text{up, down, left, right, stay}\}$$

Action Space Table

Action	Code
Up	0
Down	1
Left	2
Right	3

Updated Reward Function

Typical rewards for area mapping are to encourage exploration and also to avoid unnecessary movement.

$$R(s, a, s') = \begin{cases} +10, & \text{if } s' \text{ is a new (unvisited) cell} \\ -1, & \text{if } s' \text{ is already visited} \\ -5, & \text{if } s' \text{ is an obstacle} \\ -0.1, & \text{every step (energy cost)} \end{cases}$$

Rewards for various actions:

If agent enters a new cell: $R = +10 - 0.1 = +9.9$. If it revisits a cell:

$R = -1 - 0.1 = -1.1$. If it hits obstacle: $R = -5 - 0.1 = -5.1$.

Q-learning formulation

Update Rule— $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$. Assume

Learning Rate, $\alpha = 0.5$ and Discount Factor, $\gamma = 0.9$.

The exploration objectives are:

- 1) The agent learns to maximize coverage (visit all non-obstacle cells);
- 2) Avoid obstacles;
- 3) Avoid unnecessary revisits; and
- 4) Minimize movement cost.

Various actions by the agent

Assume $Q(s_t, a_t) = 0$ and $\max_{a'} Q(s_{t+1}, a') = 0$.

Various Transition Steps

Action: Move from state 0 to 1—moving right (3)

$0 \rightarrow 1$ is a valid move

$p_0 = 0$ and $p_1 = 1$

$M_0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ —Note that the robot has not explored any cells yet.

$M_1 = [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

Reward = +9.9

Q-learning update for the above step:

$$Q(0, 3) = 0 + 0.5(9.9 + 0.9 \cdot 0) = 4.95$$

Action: Move from state 6 to 7—moving right (3)

$6 \rightarrow 7$ is not a valid move (obstacle)

$p_6 = 6$ and $p_7 = 6$

$M_6 = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ —the robot has explored 6 cells.

Reward = -5.1

Q-learning update for the above step:

$$Q(6, 3) = 0 + 0.5(-5.1) = -2.55$$

Although the above-described architecture can minimize latency, maximize energy efficiency, and boost processing speed using onboard computation and hier-

archical task execution, these advantages are anticipated on the basis of the theoretical framework that underpins the architecture. They are not empirically established in this research.

7. IoT ML Models for Autonomous Area Mapping

Reinforcement Learning (RL) models have been used to carry out autonomous area mapping in Internet of Things (IoT) settings, as this allows agents to efficiently explore unknown spaces with limited resources. In this case, IoT technology is used to create intelligent agents, like robots, drones, and sensors, that can collect spatial information and interact with edge or cloud nodes for processing. Among the different reinforcement learning models, Q-learning is considered very appropriate for autonomous area mapping as it does not require previous models of the environment and the optimal navigation policies can be learnt directly from interaction with the surrounding of the environment.

In an autonomous mapping scenario based on IoT, the physical world is typically represented as a grid-world, where each cell of the grid corresponds to a state. Obstacles like walls, furniture, or restricted areas are represented either as invalid states or as states with high negative rewards. The IoT agent explores the environment using its onboard sensors (LiDAR, ultrasonic, infrared, and cameras) to identify obstacles and unvisited areas in real-time [13]. The sensor data is converted into state transitions for the Q-learning algorithm, where moving up, down, left, or right actions are assessed based on their long-term expected reward. This close integration of sensing (IoT layer) and learning (ML layer) makes it possible to learn adaptive maps even in dynamic or partially observable environments [14]-[16].

The reward function is of significant importance in guiding the Q-learning algorithm in autonomous mapping. For example, positive reward is usually provided when entering unexplored cells, thus maximizing coverage, while negative reward is provided in cases of re-visiting areas, collision with obstacles, or excessive energy usage. In the case of IoT networks, since the energy level of batteries is important, step penalty is usually integrated into the algorithm to represent battery usage or communication overheads. Therefore, the inclusion of Q-values to balance exploration, energy cost and safety is very important, resulting in an optimal navigation policy while avoiding obstacles and efficiency.

From a system perspective, IoT systems frequently utilize Q-learning in combination with edge computing to achieve efficient autonomous mapping. This is because, instead of transmitting raw sensor data to the cloud for processing, edge nodes may update Q-values and partial maps in parallel, transmitting only higher-level information if necessary. This is particularly beneficial in multi-agent systems where multiple IoT devices may cooperatively map large areas. Each agent is able to manage its own Q-table while sharing its learning with other agents to achieve faster convergence of the learning process.

In conclusion, Q-learning-based ML models in IoT systems provide an efficient

and clear framework for autonomous area mapping in the presence of obstacles for applications such as smart buildings, environmental monitoring, and warehouse automation. This is because they have low computational complexity with model-free nature, and can handle discrete state-action spaces, making them suitable for practical applications in IoT systems. Although more advanced models such as DRL (Deep Reinforcement Learning) can handle large and complex spaces, Q-learning is still an efficient solution for autonomous mapping in the presence of obstacles in IoT systems, even with limited resources.

8. A Potential System Design Model

Building on the decision-making model, the system-level architecture required to support real-time deployment is described. A potential system design model that can support the suggested approach can be based on a lightweight drone such as a quadcopter that is dedicated to imaging. The quadcopter will be equipped with a high-resolution RGB camera, along with a multispectral thermal camera, as well as IMU (Inertial Measurement Unit). IMU could be a set of hardware sensors capable of measuring the movements and rotations of the drone generally made up of accelerometers, gyros, and a magnetometer. The edge computing device should also be attached to the quadcopter.

Vision transformers and other efficient deep learning architectures that enhance images, detect objects, and interpret scenes can also be deployed in a hierarchical computing system. Tasks with tight latency requirements, such as capturing, stabilization, and enhancement of images and object detection, are performed locally on the UAV. All other tasks requiring more resources, such as stitching multiple frames, performing sophisticated filtering, and tracking features, can be assigned to a nearby edge server through a low-latency connection. The most computationally intensive tasks, such as analyzing big data sets, training machine learning models, and storing data for archival purposes, are delegated to the cloud.

9. Conclusions

In summary, incorporating Q-learning in autonomous area mapping provides an impressive and intuitive approach for enabling intelligent agents to explore unknown environments with obstacles. By modeling the area mapping problem as a Markov Decision Process, the agent learns optimal navigation policies by interacting with the environment directly, without the need for priori knowledge. The addition of obstacles as penalized states or forbidden transitions enables the learned policy to seek safe and feasible paths naturally, while simultaneously improving map coverage over time.

Reinforcement learning is very effective in situations where uncertainty and partial observability are common. By formulating a thoughtfully planned reward function that favors the exploration of new regions while discouraging collisions, redundant movement, and high energy usage, the agent can effectively explore and exploit the environment. In the long run, this learning mechanism enables efficient

area coverage, reduced navigation errors, and improved adaptability to complex environments. The interpretability of Q-learning also makes it easier to validate for real-world scenarios.

The presence of obstacles underscores the significance of adaptive decision-making for autonomous mapping systems. Static and dynamic obstacles play an important role in influencing state transitions and reward functions, thus forcing the system to be constantly updated. However, Q-learning is able to handle such challenges through incremental updates to the action value estimates, thus enabling the system to be constantly updated as more information is available from the environment. This is highly advantageous for indoor and urban settings where obstacles are common.

In practical terms, Q-learning-embodied autonomous area mapping is highly appropriate for platforms such as mobile robots and IoT devices, even with limited resources. This is due to the low computational complexity associated with discrete state-action spaces, thus enabling real-time learning and decision-making for such platforms at the edge. Alternatively, it is also possible to use Q-learning as a basis for mapping with automated decision-making, thus enabling seamless integration with more complex planning and cooperative multi-agent mapping strategies.

Although the framework has proven to be effective in a controlled numeric environment, some limitations emerge when it is used for actual UAV operations. Firstly, scaling beyond small or grid-based problems poses certain computational problems, because the state space and action space increase significantly in size, requiring approximation techniques or planning in hierarchical form. Secondly, the current model works in discrete time, which may not entirely reflect the continuous dynamics, control inputs, and aerodynamics of real UAVs. Thirdly, the current framework is based on idealized sensory inputs. In a real situation, sensors will have noise, inaccuracies in perception, and limited observation capability, which will affect the operation efficiency of the algorithm. Lastly, there are some other important safety requirements that should be taken into account. Collision avoidance, no-fly zones, and regulatory limitations, although they are implicitly included, should be considered explicitly during the implementation process.

Overall, Q-learning-embodied autonomous area mapping presents a strong balance of efficiency, robustness, and simplicity of implementation for aerial photography using edge computing and AI-powered algorithms. Although there is a need to extend this to a wider scope with function approximation and deep reinforcement learning, the basic principles of this case are very relevant. Thus, Q-learning in reinforcement learning paradigm remains a trusted and clear approach for obstacle-aware autonomous mapping and it provides a good foundation for intelligent navigation and exploration systems.

Author Contributions

Conceptualization, Nuraj Indika Rajapakse; methodology, Nuraj Indika Rajapakse

and Ismail Ladele Ladipo; software, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; validation, Nuraj Rajapakse and Ismail Ladele Ladipo; formal analysis, Nuraj Indika Rajapakse; investigation, Nuraj Rajapakse and Ismail Ladele Ladipo; resources, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; data curation, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; writing—original draft preparation, Nuraj Indika Rajapakse; writing—review and editing, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; visualization, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; supervision, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; project administration, Nuraj Indika Rajapakse and Ismail Ladele Ladipo; funding acquisition, N/A. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

- [1] Xia, X., Fattah, S.M.M. and Babar, M.A. (2023) A Survey on UAV-Enabled Edge Computing: Resource Management Perspective. *ACM Computing Surveys*, **56**, 1-36. <https://doi.org/10.1145/3626566>
- [2] Chang, Z., Liu, S., Xiong, X., Cai, Z. and Tu, G. (2021) A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things. *IEEE Internet of Things Journal*, **8**, 13849-13875. <https://doi.org/10.1109/jiot.2021.3088875>
- [3] Zhu, W. and Chen, K. (2026) Real-Time Object Detection for Unmanned Aerial Vehicles Based on Vision Transformer and Edge Computing. *Scientific Reports*, **16**, Article No. 6814. <https://doi.org/10.1038/s41598-026-37938-5>
- [4] Koubaa, A., Ammar, A., Abdelkader, M., Alhabashi, Y. and Ghouti, L. (2023) AERO: AI-Enabled Remote Sensing Observation with Onboard Edge Computing in UAVs. *Remote Sensing*, **15**, Article No. 1873. <https://doi.org/10.3390/rs15071873>
- [5] Cao, Z., Kooistra, L., Wang, W., Guo, L. and Valente, J. (2023) Real-Time Object Detection Based on UAV Remote Sensing: A Systematic Literature Review. *Drones*, **7**, Article No. 620. <https://doi.org/10.3390/drones7100620>
- [6] Hakani, R. and Rawat, A. (2024) Edge Computing-Driven Real-Time Drone Detection Using YOLOv9 and NVIDIA Jetson Nano. *Drones*, **8**, Article No. 680. <https://doi.org/10.3390/drones8110680>
- [7] Manoj, H.M., Shanthi, D.L., Lakshmi, B.N., Archana, K.J., Venkata Naga Jyothi, E. and Archana, K. (2025) AI-Driven Drone Technology and Computer Vision for Early Detection of Crop Disease in Large Agricultural Areas. *Scientific Reports*, **16**, Article No. 2479. <https://doi.org/10.1038/s41598-025-32384-1>
- [8] Katarya, R., Raturi, A., Mehndiratta, A. and Thapper, A. (2020) Impact of Machine Learning Techniques in Precision Agriculture. 2020 3rd International Conference on Emerging Technologies in Computer Engineering. Machine Learning and Internet of Things (ICETCE), Jaipur, 7-8 February 2020, 1-6. <https://doi.org/10.1109/icetce48199.2020.9091741>
- [9] Torai, S., Chiyoda, S. and Ohara, K. (2020) Application of AI Technology to Smart

- Agriculture: Detection of Plant Diseases. 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Chiang Mai, 23-26 September 2020, 1514-1519. <https://doi.org/10.23919/sice48898.2020.9240353>
- [10] Harmani, V.P., Himawan, B.M., Alhadi, M.A., Gunawan, A.A.S. and Anderies (2022) Systematic Literature Review: Implementation of Artificial Intelligence in Precision Agriculture. 2022 5th International Conference on Information and Communications Technology (ICOIAC), Yogyakarta, 24-25 August 2022, 479-484. <https://doi.org/10.1109/icoiact55506.2022.9971917>
 - [11] Baghel, S.S., Rawat, P., Singh, R., Akram, S.V., Pandey, S. and Baghel, A.S. (2022) AI, IoT and Cloud Computing Based Smart Agriculture. 2022 5th International Conference on Contemporary Computing and Informatics (IC3I), Uttar Pradesh, 14-16 December 2022, 1658-1661. <https://doi.org/10.1109/ic3i56241.2022.10072567>
 - [12] Kuradusenge, M., Hitimana, E., Hanyurwimfura, D., Rukundo, P., Mtonga, K., Mukasine, A., *et al.* (2023) Crop Yield Prediction Using Machine Learning Models: Case of Irish Potato and Maize. *Agriculture*, **13**, Article No. 225. <https://doi.org/10.3390/agriculture13010225>
 - [13] Ullo, S.L. and Sinha, G.R. (2021) Advances in IoT and Smart Sensors for Remote Sensing and Agriculture Applications. *Remote Sensing*, **13**, Article No. 2585. <https://doi.org/10.3390/rs13132585>
 - [14] Sharma, A., Jain, A., Gupta, P. and Chowdary, V. (2021) Machine Learning Applications for Precision Agriculture: A Comprehensive Review. *IEEE Access*, **9**, 4843-4873. <https://doi.org/10.1109/access.2020.3048415>
 - [15] Liakos, K., Busato, P., Moshou, D., Pearson, S. and Bochtis, D. (2018) Machine Learning in Agriculture: A Review. *Sensors*, **18**, Article No. 2674. <https://doi.org/10.3390/s18082674>
 - [16] Yu, J., Sharpe, S.M., Schumann, A.W. and Boyd, N.S. (2019) Deep Learning for Image-Based Weed Detection in Turfgrass. *European Journal of Agronomy*, **104**, 78-84. <https://doi.org/10.1016/j.eja.2019.01.004>