

Article

A Context-Aware Adaptive Framework for UAV-Based Target Detection and Tracking

Tolga Berberoglu ¹ and Buket Kaya ^{2,*} ¹ Ministry of Education, Elazig 23200, Türkiye; tberberoglu23@gmail.com² Department of Electronics and Automation, Firat University, Elazig 23119, Türkiye

* Correspondence: bkaya@firat.edu.tr

Highlights

What are the main findings?

- A context-aware UAV framework that dynamically selects detection and tracking algorithms based on terrain type and mission requirements significantly improves tracking robustness.
- The proposed system achieves ~85% mAP50 while reducing computational overhead through lightweight terrain analysis and adaptive algorithm selection.

What are the implications of the main findings?

- Incorporating environmental awareness into UAV vision systems enhances autonomy and enables more efficient use of limited onboard resources in dynamic scenarios.
- The adaptive detection–tracking strategy with feedback mechanisms provides a scalable foundation for real-time, reliable UAV operations in complex environments.

Abstract

Unmanned Aerial Vehicles (UAVs) have become critical platforms for missions such as surveillance, reconnaissance, and target tracking, which require real-time decision-making, reliable sensing, and efficient resource utilization. However, limited onboard computing capacity, energy constraints, variable terrain conditions, and situations where targets are partially or fully obscured limit the performance of traditional fixed-configuration sensing and tracking approaches. In this study, we propose a context-aware and adaptive UAV-based target detection and tracking framework that dynamically selects the most appropriate detection and tracking algorithm by jointly evaluating terrain characteristics and mission requirements. The proposed system includes a three-stage terrain analysis module supported by HSV color space filtering, Canny edge detection, Laplacian texture variance, and contrast-based features. In cases where color-based classification is insufficient, Random Forest-based classification is used to distinguish between vegetation, bare ground, and urban areas; the terrain classification model achieves approximately 90% accuracy during the training and testing process. In the target detection phase, a YOLOv11-based model was trained on a specialized tank dataset created from various sources and labeled in YOLO format, achieving an mAP50 performance of approximately 85%. In the tracking phase, single-object and multi-object tracking algorithms are selected via a scoring-based decision mechanism depending on the terrain type and mission scenario. Additionally, a hybrid anomaly detection mechanism that evaluates target loss, sudden bounding box changes, and view inconsistencies was integrated into the system, thereby enhancing tracking reliability and enabling the re-detection or algorithm switching process when necessary. Experimental results demonstrate that the proposed context-aware approach can reduce



Academic Editors: Higinio González-Jorge, Fernando Veiga López and Enrique Aldao Pensado

Received: 11 May 2026

Revised: 30 June 2026

Accepted: 2 July 2026

Published: 8 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

computational load while maintaining tracking robustness under various environmental conditions. These findings highlight that environmental awareness and adaptive algorithm selection can make significant contributions to autonomy, operational efficiency, and real-time reliability in UAV-based imaging systems.

Keywords: unmanned aerial vehicles; deep learning; image processing; YOLOv11 algorithm; object detection; object tracking; OpenCV

1. Introduction

Unmanned Aerial Vehicles (UAVs) have become indispensable platforms in a wide range of applications, including environmental monitoring, disaster response, surveillance, and reconnaissance. Their ability to acquire high-resolution data and operate autonomously in dynamic environments has significantly expanded their role in both civilian and defense-related domains. In such applications, reliable object detection and tracking are essential for enabling situational awareness, real-time decision-making, and mission success [1].

However, deploying robust computer vision algorithms on UAV platforms remains a challenging problem. UAVs are inherently constrained by limited onboard computational resources, energy capacity, and payload restrictions. In addition, maintaining continuous communication with a ground control station introduces security vulnerabilities and operational risks. Previous studies have shown that radio frequency (RF)-based communication can be exploited for UAV detection and identification with high accuracy, raising concerns regarding operational security [2,3]. Furthermore, cyberattacks and hijacking attempts targeting UAV control systems may disrupt mission execution, highlighting the importance of onboard autonomy [4–7]. These challenges emphasize the need for vision-based systems that can operate independently of external communication links. However, the issue of energy consumption comes into play here as well.

The massive energy consumption and rising costs associated with information and communication technology (ICT) services have garnered significant attention in both the academic community and industry [8–10]. Minimizing energy consumption for the efficient management of existing resources is among today's most critical objectives. In this context, designing sustainable artificial intelligence applications has become one of the fundamental technical challenges facing modern engineering [11].

Recent advancements in deep learning and hardware acceleration have made it possible to deploy convolutional neural networks (CNNs), which are fundamental deep learning architectures widely adopted in computer vision applications, within resource-aware and sustainable artificial intelligence systems—on embedded platforms, thereby enabling real-time object detection for UAV applications [12]. In particular, the YOLO (You Only Look Once) family of algorithms has gained significant attention due to its favorable trade-off between accuracy and computational efficiency [13]. Hybrid approaches, such as the YOLOFlow framework proposed by Ajmera and Singh [14], combine detection with motion estimation techniques to improve tracking performance. In parallel, research on resource-aware systems has focused on optimizing computational efficiency through model-level improvements and system-level adaptations [15–19]. For example, lightweight architectures and transformer-based models aim to reduce computational complexity while maintaining acceptable detection accuracy. Despite these advancements, most existing UAV-based detection and tracking systems rely on fixed algorithmic configurations that do not adapt to environmental conditions. In real-world scenarios, however, visual characteristics such as terrain type, texture complexity, and occlusion patterns vary significantly

and directly influence the performance of detection and tracking algorithms. Systems that ignore such contextual information may suffer from inefficient resource utilization or degraded tracking accuracy, especially in complex environments such as urban areas or densely vegetated regions.

To address this limitation, this study proposes a context-aware and adaptive target detection and tracking framework for UAVs. The proposed system incorporates environmental information into the decision-making process by analyzing terrain characteristics using a lightweight image processing approach based on HSV color space filtering and Canny edge detection. Based on the extracted contextual features, a scoring-based decision mechanism dynamically selects the most suitable algorithm among multiple detection and tracking methods, including YOLO-based object detection and various single-object and multi-object tracking algorithms. In addition, a feedback loop is integrated into the system to reinitialize detection when tracking performance deteriorates, thereby improving robustness.

The main contributions of this paper are summarized as follows:

1. A context-aware UAV vision framework that integrates terrain analysis into the detection and tracking decision process.
2. A dynamic scoring-based mechanism that adaptively selects the most appropriate tracking algorithm based on mission requirements and environmental conditions.
3. An integrated detection–tracking pipeline with a feedback mechanism to enhance robustness against target loss.
4. Experimental validation demonstrating that the proposed approach achieves reliable performance while maintaining low computational cost.

The remainder of this paper is organized as follows. Section 2 reviews related work on UAV-based detection and tracking systems. Section 3 presents the proposed methodology. Section 4 provides experimental results and discussion. Finally, Section 5 concludes the paper and outlines future research directions.

2. Related Works

Object detection and tracking for Unmanned Aerial Vehicles (UAVs) has been extensively studied in recent years, driven by the increasing demand for autonomous and intelligent aerial systems. However, the limited computational capacity and energy constraints of UAV platforms require a careful balance between accuracy and efficiency in the design of vision-based algorithms.

2.1. UAV Autonomy and Security Considerations

One of the key challenges in UAV operations is the reliance on continuous communication with ground control stations. Recent studies have demonstrated that radio frequency (RF)-based signals emitted by UAVs can be detected and classified with high accuracy using deep learning techniques, posing significant security risks [2,3]. In addition, cyberattacks and hijacking attempts targeting UAV control systems can compromise mission integrity and operational safety [4–7]. These findings highlight the importance of increasing onboard autonomy and reducing dependence on external communication by enabling UAVs to perform critical tasks such as detection and tracking using onboard processing capabilities.

2.2. Object Detection and Tracking Approaches

Deep learning-based object detection methods have become the dominant approach in UAV applications. In particular, the YOLO (You Only Look Once) family of algorithms offers a favorable trade-off between detection accuracy and computational efficiency, making it suitable for real-time deployment on embedded systems [13]. Hybrid approaches have also been proposed to improve tracking performance. For example, Ajmera and Singh [14]

introduced the YOLOFlow framework, which combines object detection with optical flow techniques to reduce computational cost while maintaining tracking continuity.

In addition to detection, various tracking algorithms have been widely used in UAV-based systems. Traditional single-object trackers such as KCF, MOSSE, and CSRT are known for their computational efficiency, whereas more advanced multi-object tracking approaches [15], including DeepSORT and ByteTrack, provide improved robustness in complex scenarios [16]. However, most existing methods operate under fixed algorithmic configurations and do not dynamically adapt to changing environmental conditions, operational objectives (e.g., single-target tracking, multi-target tracking, surveillance, and target counting), or computational resource constraints encountered during UAV missions.

2.3. Resource-Aware and Adaptive Systems

To address resource constraints in UAV platforms, recent research has focused on developing resource-aware and adaptive systems. These approaches can generally be divided into two categories: model (algorithm)-level optimization and system-level adaptation. Model-level optimization aims to reduce computational complexity by designing lightweight architectures. For example, transformer-based models such as Lite-DETR have been proposed to provide efficient object detection with reduced computational requirements [19]. Many studies focus on energy-aware motion planning and control to reduce energy consumption and extend flight durations; for instance, they aim to plan energy-efficient flight trajectories and prioritize maneuvers with low energy consumption [20–24]. Most of these studies focus on the goal of minimizing battery consumption by determining the most efficient trajectory rather than on how to most efficiently track a moving target in a combat environment. Another study, however, aims to improve detection accuracy using fewer parameters [18].

On the other hand, system-level adaptation methods dynamically adjust parameters such as image resolution, frame rate, or processing frequency based on hardware-related factors such as battery level, processor load, and temperature [17]. These methods have shown promising results in extending UAV mission duration and improving energy efficiency. Additionally, examples of structural redesign [25,26] have been proposed to optimize the UAV's structure and design—for instance, by using lightweight manufacturing materials, carefully selecting components, or redesigning the structure—to reduce weight. While hardware-based optimization typically requires extensive development, algorithm-based optimization is more flexible and can be applied to existing systems [24].

Despite these developments, most studies on the efficiency of UAVs at the model or system level mentioned above have focused on calculating the most efficient routes for the UAV to use its energy source efficiently. Khalili et al. have proposed a framework that simultaneously optimizes trajectory design and resource allocation to minimize energy consumption in UAV-based integrated sensing and communication (ISAC) systems [20]. In this study, to enhance sensing accuracy and reduce the adverse effects caused by UAV vibrations, a constraint requiring the UAV to hover over the target during the sensing task was applied. This significantly limits the project's efficiency. Cheng et al. investigate an autonomous vision-based tracking system using a rotor-equipped UAV with an onboard gimbal camera to track a maneuvering target [27]. In this paper, a Kernel Correlation Filter (KCF) tracker is combined with a re-localization method within the system. An IMM-EKF-based estimator is used to estimate the target's states. Experimental results show that while the proposed real-time vision-based tracking system offers robust and reliable tracking performance, this image-based tracking system faces significant challenges in re-establishing tracking capabilities when an object moves rapidly or a jump loss occurs. In full occlusion scenarios, the traditional KCF framework becomes unable to track targets

accurately [28]. In the system we developed, the goal is to design a more efficient system by addressing the issues of low success rates in target detection by tracking algorithms and high performance issues in object detection algorithms. The related paper [29] discusses a hybrid method developed specifically for tracking dynamic targets. The system uses correlation filters for tracking, SVM for locating lost targets, and optical flow supported by EKF for position estimation. Although field tests were conducted outdoors, it was observed that the system could not detect obstacles and avoid them due to a lack of situational awareness. Paper [30] presents a comprehensive framework for autonomous vision-based target tracking in UAVs. The proposed system uses an onboard RGB camera to detect and track a target using an HSV-based color segmentation approach. An Extended Kalman Filter (EKF) was used to estimate the target's state, thereby ensuring continuous tracking even in the presence of temporary obstacles. Although high frame rates were achieved, the experiment was conducted entirely in a closed environment. Additionally, colors more suited to the environment were used to make tracking objects in a combat zone more challenging. Consequently, the success rate of target detection using HSV color segmentation is very low.

The adaptive systems discussed in this article are largely “environmentally insensitive,” as they rely primarily on hardware-level indicators rather than understanding the scene. In practice, however, environmental characteristics such as terrain type, texture complexity, and levels of occlusion significantly affect detection and tracking performance. For example, selecting a computationally lightweight tracking algorithm in a complex urban environment may result in the target being lost, whereas in simple open terrain, more complex algorithms may be unnecessary.

Table 1 systematically summarizes the key characteristics of the related studies.

Table 1. Comparison of studies based on different criteria.

Study	Optimization Type	Environment-Aware	Energy-Aware	Adaptive Decision Mechanism	Real-Time Focus	Main Limitation
Ajmera & Singh (YOLOFlow)	Algorithm-level	No	Partial	No	Yes	Fixed configuration
Lite-DETR	Algorithm-level	No	Partial	No	Yes	Detection only
Energy-aware trajectory planning studies	System-level	No	Yes	Partial	Yes	Focus on flight path
Cheng et al.	Tracking-level	No	No	No	Yes	Occlusion sensitivity
Proposed Framework	Algorithm + System-level	Yes	Yes	Yes	Yes	Limited embedded validation

2.4. Research Gap and Contribution

A review of the literature indicates that most existing UAV-based detection and tracking systems focus either on improving accuracy or reducing computational cost, but rarely consider environmental context as a key factor in algorithm selection. While resource-aware systems adapt to hardware constraints, they generally lack the ability to incorporate scene-level information into the decision-making process.

To address this gap, the present study proposes a context-aware framework that integrates environmental analysis with a dynamic decision-making mechanism. By combining lightweight terrain classification with a scoring-based algorithm selection strategy, the proposed system enables UAVs to adapt their detection and tracking behavior according to both mission requirements and environmental conditions. This approach extends existing resource-aware methods by introducing environmental awareness as an additional dimension of adaptation, leading to improved robustness and efficiency in real-world scenarios.

3. Materials and Methods

When unmanned aerial vehicles (UAVs) fly at high altitudes, they are easily obstructed by objects such as trees and buildings and encounter various backgrounds. This makes it difficult for UAVs to detect and track targets. UAVs are limited by their payloads and energy consumption, so the demand on computational resources must be kept as low as possible. Therefore, algorithms must minimize computational complexity and data transmission while ensuring performance. The primary objective of this study is to design a system that enables the UAV to analyze its current mission status and the surrounding environment, determine the most suitable algorithm for mission execution, and optimize the use of the UAV's resources. The system design is outlined below.

The system design shown in Figure 1 above consists of three main components. 1. A terrain classification algorithm that makes decisions regarding the geographic area where the task will be performed. 2. A fine-tuned YOLOv11-m model was trained using a custom-curated dataset consisting of 2500 UAV-oriented aerial images for tank detection tasks. 3. An algorithm that selects which detection algorithm to use based on the task status and geographic conditions and manages the task. All experiments were conducted using Python 3.10 with OpenCV (opencv-contrib-python) version 4.11.0.80.

Algorithm 1 summarizes the operational workflow illustrated in Figure 1 and explicitly describes the adaptive decision-making procedure of the proposed framework.

Algorithm 1. Context-Aware Adaptive Detection and Tracking Framework

```

Input: UAV video stream V, mission type M
Output: Selected tracking algorithm T and tracking results R
1: while UAV mission is active do
2:   Acquire current frame  $F_t$ 
3:   Perform terrain analysis
4:    $terrain \leftarrow \text{classifyTerrain}(F_t)$ 
5:   if target is not detected then
6:      $target \leftarrow \text{YOLO\_Detection}(F_t)$ 
7:   end if
8:   Calculate mission and terrain scores
9:    $T \leftarrow \text{selectHighestScoreAlgorithm}()$ 
10:   $tracking\_result \leftarrow T(target)$ 
11:  if  $tracking\_failure\_detected$  then
12:    Re-run YOLO detection
13:    Update algorithm scores
14:     $T \leftarrow \text{selectNextBestAlgorithm}()$ 
15:  else
16:    Reward successful algorithm
17:    Store results in database
18:  end if
19: end while
20: return tracking results

```

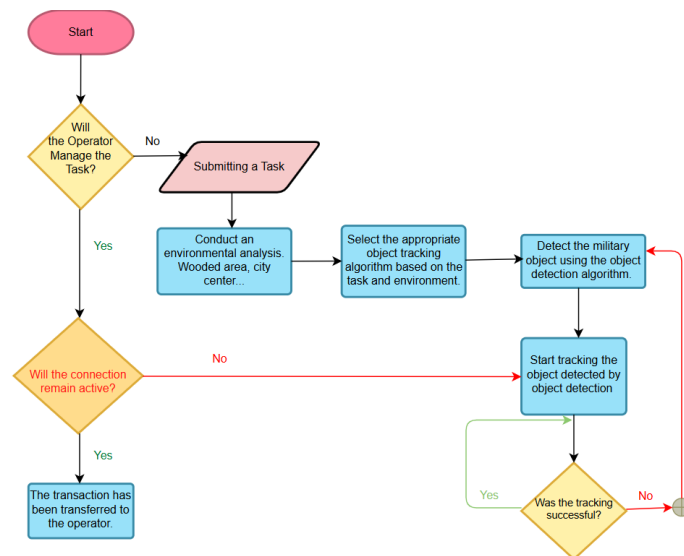


Figure 1. Flowchart of the Proposed UAV-Based Target Detection and Tracking System.

3.1. Land Classification System

The terrain classification module consists of three stages. These stages ensure that the device activates in different situations; otherwise, they remain in sleep mode to conserve energy.

1. Stage (HSV Color Space-Based Pre-Classification)

In this stage, the image captured by the UAV's camera is first analyzed using HSV color filtering, as shown in Figure 2. The HSV (Hue, Saturation, Value) color space is an important representation widely used in the fields of image processing and computer vision [31]. This model defines colors based on three fundamental components: hue, saturation, and value. Hue refers to the type of color; saturation refers to the purity or intensity of the color; and value refers to the brightness level of the color. This enables a clearer and more effective analysis of color characteristics in visual data. The HSV color space offers a structure that is more resistant to changes in lighting compared to RGB. Since the color components are independent of one another, a more consistent color analysis can be performed on images, and color-based detection of target objects yields more accurate results. In our study, to extract the color percentage of forested and grassy areas (green), steppe and soil areas (brown), and concrete areas such as city centers (gray), we masked the HSV image within the range of colors we needed to filter.



Figure 2. The process of labeling images.

Over 10 frames, the algorithm sums the mask ratios and stores them in a 3×3 ratio matrix. These are then compared against predefined threshold values. The mathematical and index equivalents of the threshold values are as follows.

- Grass/forest (index0) dominance check: To check whether the ratio of the cumulative green area accumulated over 10 frames on the scene to the Bare Ground ratio is greater than 2 and the urban ratio is greater than 2.2, use the [row_index, column_index] slicing:

$$Ratio_matr[1, 0 :] \rightarrow \left[\frac{Grass}{Bare\ Ground} \quad \frac{Grass}{Urban} \right]$$

- Bare ground (Index 1) Slicing the cumulative bare ground weight accumulated on the stage over 10 frames into the interval [1, [0, 2]] to check whether the ratio to grass is greater than 2 and the ratio to urban areas is greater than 2.2:

$$Ratio_matr[1, [0, 2]] \rightarrow \left[\frac{Bare\ Ground}{Grass} \quad \frac{Bare\ Ground}{Urban} \right]$$

- Urban (Index 2): Based on the cumulative weight accumulated over 10 frames on the scene, the grass ratio was set to 2.8 and the urban ratio to 3. Since urban areas are densely populated, a higher ratio was selected to account for the potential for misclassification. To verify this, the range was sliced into [2, [0, 1]]:

$$Ratio_matr[1, [0, 1]] \rightarrow \left[\frac{urban}{Grass} \quad \frac{Urban}{Bare\ Ground} \right]$$

2. Stage (Land Classification Using Machine Learning-Based Random Forest)

In cases where the color-based classification results obtained in the first stage do not achieve a sufficient level of confidence, the system proceeds to the second stage. In this stage, a Random Forest-based machine learning method is used to more accurately determine the land cover class to which the image belongs.

In this context, the image is first resized to a standard format and converted to the HSV color space. Histograms are generated from the Hue and Saturation channels of the HSV image to obtain features that represent the color distribution of the scene. These features are used specifically to identify color differences between areas with dense vegetation and areas of soil and urban regions.

In addition to color information, various attributes representing the structural characteristics of the image are also calculated. First, the image's textural complexity is determined by calculating the standard deviation of the grayscale image. Next, the edge density in the image is obtained using the Canny edge detection algorithm [32]. Edge density is used as a key parameter representing the density of buildings, roads, and other structural elements found in urban areas. Finally, the average brightness value of the image is calculated to distinguish areas of bare soil and arid terrain from other classes.

As part of the land classification studies, various feature extraction and classification methods were first evaluated. In this context, the Local Binary Patterns (LBP) method was used to represent the textural characteristics of the images, and the extracted features were classified using the Support Vector Machine (SVM) algorithm [33]. Additionally, the extraction of texture characteristics of the land surface using Gabor filters and their incorporation into the classification process were also examined. However, in the experiments conducted, it was observed that these methods could not provide sufficient stability in real-time UAV images due to variations in lighting conditions, camera heights, and land surface structures. In particular, fluctuations in classification performance occurred when distinguishing

between vegetated and bare soil areas. Therefore, a Random Forest-based approach capable of evaluating color, texture, and structural features together was preferred, resulting in more stable outcomes [34].

All the extracted features are combined to form a feature vector, which is then fed into the Random Forest model created during the training phase. The Random Forest algorithm operates through a collective decision-making mechanism involving a large number of decision trees; the final class is determined by voting on the results produced by each tree. This method was chosen because it can produce more reliable results under varying environmental conditions and image variations.

A total of 1193 images representing three different land cover classes were used to train the model. The dataset consists of three classes: areas with vegetation (grass), bare ground areas (Bare Ground), and urban areas (Urban). Eighty percent of the dataset was allocated for training, and 20% for testing. In the tests conducted after training, the model achieved a classification accuracy of 90%. These results demonstrate that the selected features are effective in distinguishing between land classes. The UC Merced Land Use dataset was used to train the land classification model [35].

3. Stage (Grassland–Forest Classification Using Multi-Attribute Decision Fusion)

As soon as the system determines that the underlying terrain consists of vegetation in either Stage 1 or Stage 2, it immediately puts the Bare Ground and Urban filters into sleep mode and routes the data directly to Stage 3: Multi-Attribute Decision Fusion. In this final stage, to determine whether the vegetation is grassland or a dense forested area, the Canny edge detection algorithm calibrated to UAV high-altitude parameters (100, 200, 1), Laplacian texture variance, and global contrast deviation are calculated at the scalar level. The obtained data are subjected to a linear Majority Voting filter, and the final class decision (Grass or Forest) is transmitted to the autopilot.

To quantitatively evaluate the performance of the land classification module, a balanced test set consisting of four classes was prepared. The test set comprises a total of 80 images, with 20 images for each of the following classes: forested, grassland, bare soil, and urban. The evaluation was conducted not only based on the output of the Random Forest stage but also on the final decision of the proposed multi-stage land classification pipeline. In this context, HSV-based pre-classification, the Random Forest-supported decision tree, and the Canny/Laplacian/contrast-based grassland–forested area distinction—applied in cases where vegetation was detected—were evaluated together. The obtained predictions were compared with the true class labels, and the confusion matrix along with class-based precision, recall, and F1-score values were calculated.

An analysis of the confusion matrix reveals that the system correctly classified 70 out of a total of 80 images, achieving an overall accuracy of 87.50% (See Table 2). In the class-based results, higher F1-score values were obtained for the bare soil and urban classes, while partial confusion occurred between the forested and grassland classes (See Table 3). This situation is related to the fact that both classes contain green vegetation and that some grassland scenes include scattered clusters of trees. Nevertheless, the macro-average F1-score of approximately 0.87 indicates that the proposed multi-stage land classification framework produces sufficiently discriminative results for the selection of an adaptive tracking algorithm.

Table 2. Confusion matrix of the terrain classification module.

Actual/Estimate	Forest	Grass	Bare Ground	Urban	Total
Forest	16	3	0	1	20
Grass	2	17	1	0	20

Table 2. *Cont.*

Actual/Estimate	Forest	Grass	Bare Ground	Urban	Total
Bare Ground	0	1	19	0	20
Urban	1	0	1	18	20

Table 3. Class-wise performance results of the terrain classification module.

Class	Precision	Recall	F1-Score	Support
Forest	0.84	0.80	0.82	20
Grass	0.81	0.85	0.83	20
Bare Ground	0.90	0.95	0.93	20
Urban	0.95	0.90	0.92	20

3.2. Object Detection

The study used images obtained from RoboFlow, online sources, and combat videos for tank detection. Frames were extracted from the video sources to create a raw image pool of approximately 15,000–20,000 images; from this pool, 2500 images were selected that were suitable for the UAV's viewpoint and varied in terms of the target's position, scale, background, and field of view. Care was taken to ensure that consecutive and highly similar frames were not heavily represented in the dataset.

To reduce the risk of frame leakage and overfitting between the training and test sets, data splitting was performed under manual control rather than through automatic random distribution. During this process, care was taken to ensure that consecutive frames or visually nearly identical frames from the same video sequence did not end up in both the training and test sets simultaneously, and the final dataset was split into training and test sets at a 90/10 ratio. While a strict video-based separation cannot be fully guaranteed for all sources, a manual selection and verification process was applied to reduce the risk of similar frame leakage.

After the dataset was created, images containing tanks were labeled according to the YOLO format, while images without tanks were excluded from the labeling process. The text files in YOLO format obtained after labeling were saved for use in the training process.

After the images to be trained using the YOLOv11 algorithm were collected into a dataset, the dataset was divided into training, validation, and test sets. Ninety percent of the final dataset was used for the training/validation process, while 10 percent was used for testing. The validation set, separated from the training/validation portion, was used to monitor model performance during training and to control overfitting. The image sources used to create the dataset and the distribution of images from these sources are shown in Table 4.

Table 4. Data Set Statistics.

Data Source	Total Image	Selected Image	Total Tanks (BBBox)	Resolution Range
RoboFlow	~6.000	1.100	1850	640 × 640–720P–1080P
Internet/War Videos	~12.000	1400	2400	640 × 640–720P–1080P
Total	~20.000	2500	4250	640 × 640–720P–1080P

The model was trained using the Google Cloud infrastructure. In the development environment, the GPU-accelerated version of the PyTorch (2.8.0+cu128) framework was used, and the training process was run on a single GPU. During training, the transfer learning method was applied to increase the convergence rate and leverage pre-trained

features. Accordingly, the model was first trained using the YOLOv11m.pt weight file, which had been pre-trained on the MSCOCO dataset; the system was then optimized using the resulting best.pt weight file. The key hyperparameters and configuration details used during the training process are summarized in Table 5.

Table 5. YOLO11 Model Training Hyper parameters and Values.

Parameter (Code Equivalent)	Description/Responsibilities	Value
Task/Mode	The model's execution task and mode	detect/train
Model	The main weight file used at the start of training	best.pt
Epochs	Maximum number of training epochs	300
Batch Size	Number of images processed per batch	16
Image Size	Input image resolution	640 × 640
Patience	Early stopping patience value	200
Optimizer	Selection of an Optimization Algorithm	Auto
Initial learning rate(lr0)	Initial learning rate value	0.001
Final learning rate factor(lrf)	Final learning rate scaling factor	0.01
Pretrained/Val	Use of a pre-trained model and validation status	True/True
Workers	Number of data loading workers	8

Training took a total of 4 h. The metrics resulting from the training are presented in the graphs below.

Figure 3 demonstrates that the training and validation loss curves converge in a similar manner, suggesting good generalization performance of the trained model. As noted, the slight increase in the val/cls_loss graph shows that the model reached its highest generalization capacity around the 150th epoch in this section. Figure 4 presents the performance metrics of the trained detection model, including Precision, Recall, and mean Average Precision (mAP). The model achieves an mAP50 score of approximately 85%, indicating a strong capability to accurately detect target objects under varying conditions. The Precision and Recall curves exhibit a balanced behavior, which suggests that the model effectively minimizes both false positives and false negatives. This balance is critical in UAV-based applications, where both missed detections and false alarms can negatively impact mission performance. Furthermore, the consistency observed in the evaluation metrics indicates that the model maintains stable performance across different validation samples. These results confirm that the trained detection model provides a reliable foundation for the subsequent tracking and decision-making stages of the proposed framework.

3.3. Object Tracking

The system determines the most suitable object tracking algorithm based on geographical terrain conditions and mission status. A scoring system has been developed for this purpose. Points have been assigned to mission types and terrain characteristics. The system begins operation by selecting the algorithm with the highest score according to this scoring system. In the event of extremely negative feedback, it switches to the next highest-scoring tracking algorithm. Additionally, if tracking is successful, extra points are awarded for the algorithm's mission and geographic area components; the goal is to create a dataset between geographic area and object tracking algorithms for future similar missions. The database structure and scoring system are shown in the table below.

Various task scenarios have been simulated in the system. Scenarios such as tracking and marking a single target, counting targets in an intelligence-gathering area, and multi-target tracking have been assumed. For single-object tracking, we used OpenCV's classic tracking algorithms—KCF Tracker, Median Flow Tracker, MOSSE, and CSRT—along with

lightweight CNN algorithms like GOTURN, and the SiamRPN-based NanoTrack [36] tracking algorithm, which was developed for mobile devices and embedded systems. For multi-object tracking, we used the DeepSORT, ByteTrack, and BoTSORT algorithms.

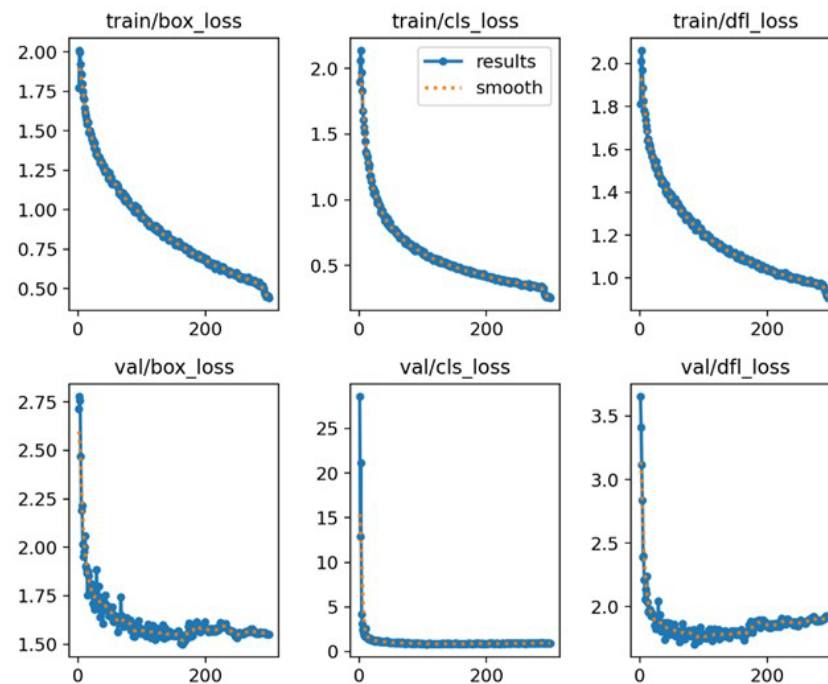


Figure 3. Training and validation losses.

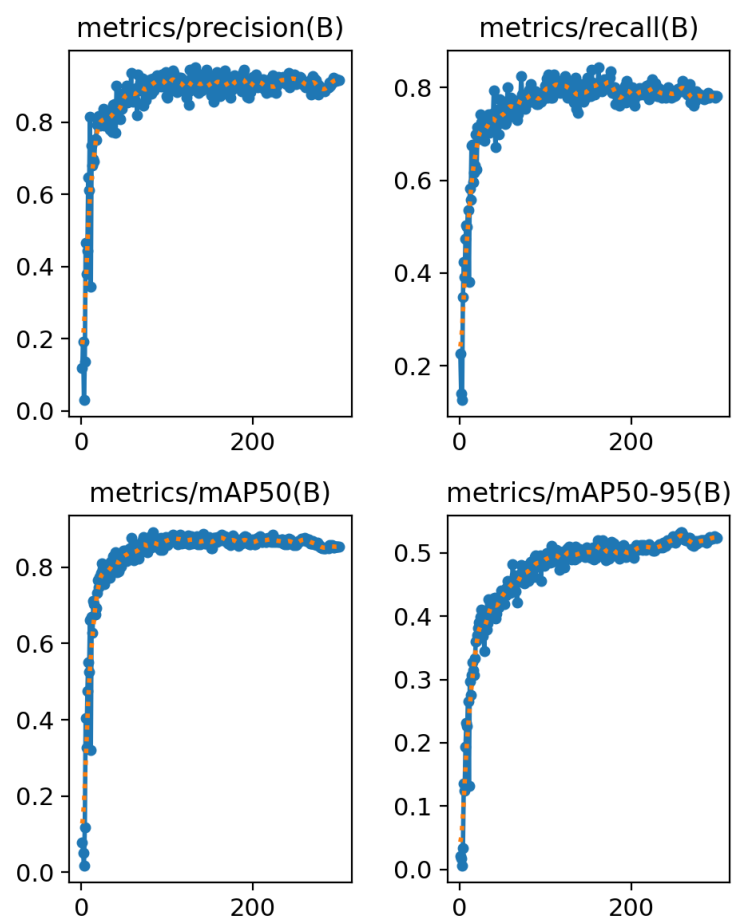


Figure 4. Performance metrics of the trained detection model.

Among the object tracking algorithms, while the KCF Tracker, Median Flow Tracker, MOSSE, and CSRT algorithms are not very efficient, they do not provide feedback when they lose track of the object. The GOTURN algorithm also lacks feedback. The feedback mechanism of the NanoTrack algorithm is also quite poor. In this study, we developed a second control mechanism.

3.4. Security Monitoring System for Object Tracking

In the proposed system, a hybrid anomaly detection mechanism is used to improve the reliability of target tracking and to prevent the tracking algorithm from continuing to track a false target when it loses the target. Conventional tracking algorithms can cause the tracking box to drift into non-target areas, particularly in situations such as brief occlusion of the target, sudden camera movements, changes in scale, low resolution, and when the target and background appear similar. In such cases, the tracking algorithm may generate a failure message in some instances, while in others it may continue to produce a bounding box even after losing the target. For this reason, the system does not rely solely on the tracking algorithm's success output; instead, it evaluates both the temporal behavior of the tracking box and the target's appearance together.

In the hybrid anomaly detection framework, the tracking algorithm's success status is first checked. When the tracking algorithm reports that it has lost the target, this is evaluated as a strong indicator of an anomaly. Additionally, even if the tracking algorithm produces a successful output, the behavior of the bounding box is analyzed separately. In this context, sudden shifts in the box center across consecutive frames, unusual expansion or contraction of the box area, and sudden changes in the aspect ratio were used to evaluate tracking reliability. These metrics were calculated based on historical box information from the most recent frames to determine whether the tracking box remained stably over the target.

The system also compared the target's initial appearance with its current image within the tracking box. For this purpose, a histogram-based similarity metric in the HSV color space was used. However, since the target, road, ground surface, and shaded areas in UAV images can be similar in terms of color distribution, HSV similarity was not used on its own as a definitive decision-making criterion. Instead, HSV-based appearance changes were evaluated as an auxiliary anomaly indicator, in conjunction with disturbances in the behavior of the bounding box.

Rather than relying on a single metric for anomaly detection, a score-based hybrid approach was adopted. Sudden jumps in the box center, area changes, aspect ratio changes, and a decrease in HSV similarity were each treated as separate suspicious conditions, and each contributed to the anomaly score. It was assumed that exceeding the defined score threshold would result in a loss of reliability during the tracking process. However, to prevent single-frame transient deviations from generating false alarms, sequential anomaly checking was implemented. In this context, the condition that the relevant disturbance must persist across a certain number of consecutive checks was required for an anomaly decision to be made.

Additionally, a separate control mechanism was defined for cases where HSV similarity remains unchanged but the bounding box behavior is continuously disrupted. Thus, even in scenes where the target and background are similar in color and HSV similarity remains high, if instability in the box's center or dimensions persists for a certain period, the system evaluates this situation as a persistent tracking anomaly. This ensures that the re-detection process can be triggered even if the tracking algorithm appears to be stable on the wrong region.

When the hybrid anomaly check detects a degradation in tracking reliability, the YOLO-based re-detection mechanism is activated. Since YOLO is run only when a tracking degradation is detected or when the target needs to be re-verified—rather than in every frame—the system’s computational load is reduced.

This approach aims to enhance tracking reliability against target loss while simultaneously reducing unnecessary deep learning-based detection calls in energy-constrained UAV systems.

4. Results

Experimental Validation and Comparative Analysis

The proposed framework was experimentally evaluated under various environmental conditions, including urban, forested, and open-area scenarios. The experiments were conducted using pre-recorded UAV videos obtained from simulated aerial surveillance missions. The study utilized videos in mp4 format at 1080p and 720p resolutions.

The experiments were conducted on a system equipped with an NVIDIA RTX 3050Ti 4GB GPU, an Intel i5 processor, and 16 GB of RAM. In the experiments, the PyTorch library was used to run the code on both the GPU and CPU, and the results were analyzed.

The following criteria were used in the comparison during the experiments.

1. The `uavyolov11.py` file, which uses yolov11 for both detection and tracking, consists of an average of 80 lines of code. This code file was run on both the GPU and the CPU to obtain results.
2. The `‘uavhybrid.py’` file, which contains the hybrid-based code developed for scene evaluation, the YOLO object detection algorithm, and the algorithm for selecting and tracking systems, consists of approximately 500 lines of code. Results were obtained by running this file on both the CPU and GPU.
3. A comparison table is provided to analyze which tracking algorithm yields more successful results in which areas.

The videos we used for testing were purchased from <https://www.vecteezy.com/> (accessed on 15 June 2026). The videos can be viewed at <https://github.com/interface79> (accessed on 15 June 2026).

1. Analysis results obtained by running the `uavyolov11.py` file, in which the standard Yolov11 is used for both object detection and object tracking.
 - a. Results obtained from GPU-based execution are presented in Table 6. It should be noted that these measurements also include certain CPU operations, as PyTorch continues to execute some fundamental OpenCV functions on the CPU during processing.
 - b. The CPU execution results are presented in Table 7. These results correspond to the entire processing pipeline executed solely on the CPU, including all object detection, tracking, and image-processing operations.
2. Analysis results of the developed hybrid-based code structure
 - a. The GPU-based execution results are presented in Table 8. It should be noted that OpenCV does not natively support GPU acceleration for its tracking algorithms. Therefore, the reported GPU results correspond exclusively to the tracking algorithms integrated within the YOLO framework, namely ByteTrack, DeepSORT, and BoT-SORT.

Table 6. YoloV11's detection and tracking analysis table on the GPU.

Device Type	Tracking Success Rate (%)	mAP50 %	FPS	GPU Usage %	CPU Usage %	GPU Power Watt	CPU Power Watt	Total Power Watt
GPU	65–80	75–87	35–40	49–57	13–17	47–53	13–15	60–68

Table 7. YoloV11's detection and tracking analysis table for the CPU.

Device Type	Tracking Success Rate (%)	mAP50 %	FPS	Cpu Usage %	Power Watt
CPU	60.4–79	75–87	4–7	27–33	33–35

Table 8. Analysis table for detecting and tracking the ByteTrack, DeepSORT, and BoT-SORT algorithms using a GPU.

Device Type	Tracking Success Rate (%)	mAP50 %	FPS	GPU Usage %	CPU Usage %	GPU Power Watt	CPU Power Watt	Total Power Watt
GPU	70–92	75–90	28–30	45–60	15–18	14–16	11–15	35–41

As a result of experiments conducted on flat grassy areas and in rural terrain, it was determined that the tracking algorithms produced similar results in terms of CPU usage, FPS values, power consumption, and tracking performance. For this reason, both environments were evaluated together, and the findings are presented in Table 9. All experiments were conducted at a video resolution of 720 p.

Table 9. Analysis table of running tracking algorithms on a CPU in grassy and rural areas.

Tracking Algorithm	Number of Times Yolo Has Been Activated	Cpu Usage %	FPS	Power Watt
KCF	0–6	6–20	55–65	13–30
Median	0–6	3–20	55–65	12–30
TLD	4–7	12–35	10–20	20–32
Mosse	0–3	2–10	60–65	7–15
CSRT	0–2	9–14	20–30	14–20
Goturn	3–7	7–20	11–15	20–25
NanoTrack	0–2	10–13	55–60	10–13
ByteTrack	Always	30–33	3	25–30
DeepSORT	Always	35–40	2–3	30–35
BoT-SORT	Always	40–45	2–3	40–45

An examination of Table 9 reveals that, in rural environments where there are no dense obstructions or complex background elements that could cause the target object to exit the field of view, the tracking algorithms generally performed well. Thanks to the clear-line-of-sight conditions, most algorithms were able to maintain target tracking throughout the mission, and the need for re-acquisition remained low. However, the two-stage tracking safety mechanism developed for the system ensured that, in the event of target loss, the YOLO-based object detection module was activated to re-locate the target, thereby enabling the tracking task to be handed back to the active tracking algorithm. This approach enhanced tracking continuity and supported the system's operational reliability.

The results show that OpenCV-based tracking algorithms and the proposed hybrid tracking architecture can achieve performance levels that allow them to operate in real time using only CPU resources. In particular, the MOSSE, KCF, Median, and NanoTrack

algorithms delivered balanced results in terms of tracking success, CPU usage, FPS, and power consumption. In contrast, it was observed that the TLD algorithm produced lower FPS values compared to the other algorithms, had a higher computational cost, and failed to deliver the expected performance in terms of tracking stability. Therefore, the contribution of the TLD algorithm to the proposed hybrid architecture was deemed limited, and it was decided to exclude it from the system architecture in subsequent experimental studies.

The findings indicate that tracking algorithms should be evaluated not only in terms of accuracy or tracking success but also in terms of computational load, energy consumption, and operational sustainability criteria.

- b. The results obtained by running tracking algorithms on the CPU in forested areas are presented in Table 10. In the experiments, both low-coverage videos—in which the target tank briefly entered behind trees, bushes, and similar natural obstacles—and high-coverage videos—in which the target tank remained behind obstacles multiple times—were used. The values in the table show the average results of the tests conducted under these different coverage scenarios. The video resolution was set to 720p for all experiments.

Table 10. Analysis table of running tracking algorithms on the CPU in a forested area.

Tracking Algorithm	Number of Times Yolo Has Been Activated	Cpu Usage %	FPS	Power Watt
KCF	4–10	14–30	20–60	15–35
Median	9–12	12–20	60–80	14–30
TLD	8–15	18–24	5–10	18–30
Mosse	3–5	16–20	80–95	14–18
CSRT	2–5	16–20	22–29	17–17
Goturn	6–9	10–18	9–15	15–20
NanoTrack	0–5	10–13	45–50	10–15
ByteTrack	Always	30–33	3	25–30
DeepSORT	Always	35–40	2–3	30–35
BoT-SORT	Always	40–45	2–3	40–45

An examination of Table 10 reveals that, in forested area scenarios, tracking algorithms produced varying results in terms of FPS, power consumption, and the number of times YOLO had to be reactivated. When the target moved behind obstacles such as trees and shrubs, this reduced tracking continuity in some algorithms and increased the need for re-detection. While MOSSE is one of the fastest and most cost-effective methods, with a frame rate range of 80–95 FPS and power consumption of 14–18 W, MedianFlow—despite reaching a frame rate of 60–80 FPS—demonstrated more limited tracking stability in scenes with obstacles due to YOLO being triggered 9–12 times.

The KCF and CSRT algorithms produced more stable results. KCF operated in the 20–60 FPS range and required YOLO to trigger 4–10 times. Although the FPS value for CSRT remained in the 22–29 range, the fact that YOLO triggered only 2–5 times indicates that this method can provide more stable tracking in forested areas. NanoTrack, on the other hand, produced remarkable results in terms of both tracking continuity and energy efficiency, operating at 45–50 FPS, with a power consumption of 10–15 W, and a YOLO activation count ranging from 0 to 5.

Due to the continuous operation of YOLO in the ByteTrack, DeepSORT, and BoT-SORT methods, FPS values dropped significantly and power consumption increased. While ByteTrack operated at 3 FPS, DeepSORT and BoT-SORT remained in the 2–3 FPS range.

In particular, the fact that power consumption in BoT-SORT rose to the 40–45 W range indicates that this method is more costly for energy-constrained UAV systems. Overall, the results show that detection-based methods, where YOLO is run on every frame in forested areas, create a higher computational load; whereas tracking architectures, where YOLO is activated only when necessary, are more advantageous in terms of FPS and energy consumption.

Urban area experiments were conducted using a small number of urban tank videos obtained from online sources, due to the limited availability of suitable UAV-based tank imagery. Since the target tank in these videos is typically located on a street and within an open field of view, concealment scenarios involving the tank moving in and out from behind buildings, vehicles, or similar urban obstacles could not be comprehensively evaluated. The results obtained by running the tracking algorithms on a CPU are presented in Table 11. The video resolution was set to 720 p for all experiments.

Table 11. Analysis table of running tracking algorithms on a CPU in an urban environment.

Tracking Algorithm	Number of Times Yolo Has Been Activated	Cpu Usage %	FPS	Power Watt
KCF	2–8	10–25	35–60	14–32
Median	4–8	8–20	58–75	13–30
TLD	6–11	15–28	8–15	19–31
MOSSE	1–4	8–15	70–85	10–16
CSRT	1–4	12–18	25–30	15–18
GOTURN	4–8	8–18	10–15	18–23
NanoTrack	0–3	10–13	50–55	10–14
ByteTrack	Always	30–33	3	25–30
DeepSORT	Always	35–40	2–3	30–35
BoT-SORT	Always	40–45	2–3	40–45

Table 11 presents the results obtained by running tracking algorithms on a CPU in urban areas. Since the target is typically located on a street and within an unobstructed line of sight in these scenarios, the need for re-detection is seen to be more limited compared to forested areas. However, due to the limited availability of suitable test videos featuring urban tank imagery, occlusion scenarios—where the target moves in and out from behind buildings, vehicles, or other urban obstacles—could not be evaluated with sufficient diversity.

In classical and lightweight tracking algorithms, where YOLO is activated only when necessary, FPS values remain higher while power consumption stays at lower levels. In contrast, in detection-based tracking methods such as ByteTrack and DeepSORT, FPS values have dropped significantly and power consumption has increased due to YOLO running continuously. This demonstrates that while the YOLO-based continuous detection approach may offer advantages in terms of tracking continuity, it also results in higher computational costs for energy-constrained UAV systems.

In general, results from urban environments indicate that lightweight tracking algorithms offer advantages in terms of real-time performance and energy efficiency when the target is tracked in open scenes. However, the limited availability of video data for urban occlusion scenarios highlights the need for more comprehensive evaluations of these environments in future studies.

5. Discussion and Limitations

5.1. Discussion

The experimental results demonstrate that the proposed context-aware adaptive framework is capable of maintaining reliable detection and tracking performance under varying environmental conditions while operating with limited computational resources. By incorporating terrain information into the decision-making process, the system is able to dynamically adjust its behavior and select the most appropriate tracking algorithm for a given scenario. This approach contrasts with conventional fixed-configuration systems and highlights the importance of environmental awareness in UAV-based vision applications.

One of the key strengths of the proposed framework is its lightweight design. Instead of remaining constantly active, the various control systems are activated only when needed and remain in sleep mode at other times. Although popular algorithms such as the CNN-based SiamRPN or the transformer-based TransT and OTrack are available, these algorithms are capable of tracking only a single object. To use them in multi-object scenarios, the system must restart for every new object entering the frame, which further limits the use of these algorithms—already characterized by high computational and energy consumption—in embedded systems. Since these models lack a general “object detector” (such as YOLO), the model typically cannot detect that it has lost track of the object when the object is completely obscured. It may continue to track the wrong location by mistaking the pixels of the tree behind the object or the background for the target object. If the object disappears completely and the model locks onto the wrong background, even if the object re-enters the frame unobstructed, the model will never be able to find it on its own again. This is because it only examines pixels within a limited search area. To find it again, manual feedback such as “the target object went there” must be provided to the system from the outside. Modern Transformer models like OTrack are much more resilient to occlusions compared to older-generation SiamRPNs thanks to their attention mechanisms. However, if the object completely exits the frame and does not return for a long time, they too will fail permanently. Whether using lightweight or powerful object tracking algorithms, the common issue is losing track of the target. Therefore, we found that using a lighter system supported by a superior object detection algorithm like YOLO is more efficient in terms of resource and energy usage than using a powerful algorithm that heavily burdens the system’s resources.

Despite these strengths, the achieved performance indicates that there is still room for improvement. While the system demonstrates promising detection accuracy and stable tracking behavior, its effectiveness is influenced by the quality of terrain classification and the design of the scoring mechanism. In particular, the rule-based nature of the terrain analysis and decision process may limit the system’s adaptability in highly complex or unseen environments.

5.2. Limitations

Several limitations of this study should be acknowledged.

First, the experimental evaluation is primarily conducted using simulated environments and video data rather than real-world UAV deployments. Although these experiments provide useful insights, they may not fully capture real-world challenges such as sensor noise, motion blur, communication delays, and varying weather conditions. Therefore, the generalizability of the results to real operational scenarios remains limited. Second, the dataset used for object detection is relatively small and focuses on a single object class. This restricts the ability of the system to generalize to more diverse targets and complex multi-class scenarios. In addition, the lack of publicly available datasets or standardized benchmarks makes it difficult to directly compare the proposed method

with existing approaches. Third, the scoring-based decision mechanism relies on manually defined weights and heuristic rules. While this approach provides interpretability and simplicity, it may not always yield optimal decisions across all scenarios. A more data-driven or learning-based strategy could potentially improve adaptability and performance. Fourth, the terrain classification module is based on simple image processing techniques, which may be sensitive to illumination changes, seasonal variations, and visual ambiguities between different terrain types. This may lead to incorrect environmental classification and suboptimal algorithm selection. Finally, the system has not been evaluated in terms of energy consumption, latency, or real-time performance on embedded hardware. These factors are critical for UAV applications and should be considered in future work to fully assess the practical applicability of the proposed framework.

6. Conclusions

In this study, a context-aware and adaptive target detection and tracking framework for UAVs has been proposed to address the challenges imposed by limited onboard resources and dynamic environmental conditions. Unlike conventional approaches that rely on fixed algorithmic configurations, the proposed system integrates terrain analysis and mission-aware decision-making into a unified framework. By utilizing lightweight image processing techniques for environmental classification and a scoring-based mechanism for adaptive algorithm selection, the system dynamically determines the most suitable detection and tracking strategy during operation. The experimental results demonstrate that the proposed approach is capable of maintaining stable detection and tracking performance while reducing computational overhead. The integration of a feedback mechanism further improves robustness by enabling the system to recover from tracking failures through automatic re-detection and algorithm switching. These findings highlight the potential of context-aware strategies in enhancing the autonomy and operational efficiency of UAV-based vision systems.

Future work will focus on expanding the proposed framework in various ways. First, we will focus on the problem of target tracking systems continuing to track a target even after it has been lost or when tracking the wrong target. When the target is lost and the object detection algorithm is activated, if another object enters the frame, object detection algorithms like YOLO analyze the entire image; consequently, the detection algorithm inputs the new target into the tracking system instead of the previously tracked target, and the object detection algorithm provides this as input to the tracking algorithm. To prevent such issues and ensure the continuity of the tracked target, research will be conducted on a local-to-global image scaling system. Second, the system will be evaluated on actual UAV platforms to measure real-time performance, energy consumption, and robustness under real-world conditions. Finally, to enhance generalization capabilities, the dataset will be expanded to include multiple object classes and a wider variety of scenarios.

Author Contributions: Conceptualization, T.B. and B.K.; methodology, T.B. and B.K.; software, T.B.; validation, T.B. and B.K.; formal analysis, T.B. and B.K.; investigation, T.B.; resources, B.K.; writing—original draft preparation, T.B. and B.K.; writing—review and editing, T.B. and B.K.; visualization, T.B.; supervision, B.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data associated with the study have not been deposited into a publicly available repository. Data will be made available on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once
mAP	mean Average Precision
CNNs	Convolutional Neural Networks
KCF	Kernelized Correlation Filter
MOSSE	Minimum Output Sum of Squared Error
CSRT	Discriminative Correlation Filter with Channel and Spatial Reliability
FPS	Frames Per Second

References

1. Pareek, B.; Gupta, P.; Singal, G.; Kushwaha, R. Person identification using autonomous drone through resource constraint devices. In *Proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*; IEEE: Piscataway, NJ, USA, 2019; pp. 124–129.
2. Mutlu Genç, Y.; Erciyas, E. İnsansız Hava Araçları (İHA) Tehditleri ve Güvenlik Yönetimi. *Türkiye İnsansız Hava Araçları Derg.* **2020**, *2*, 36–42.
3. Sazdić-Jotić, B.; Bondžulić, B.; Pokrajac, I.; Bajčetić, J.; Mokhtari, M. Drone classification based on radio frequency: Techniques, datasets, and challenges. In *Proceedings of the 10th International Scientific Conference on Defensive Technologies (OTEH 2022)*, Belgrade, Serbia, 13–14 October 2022.
4. Al-Sa'd, M.F.; Al-Ali, A.; Mohamed, A.; Khattab, T.; Erbad, A. RF-based drone detection and identification using deep learning approaches: An initiative towards a large open source drone database. *Future Gener. Comput. Syst.* **2019**, *100*, 86–97. [[CrossRef](#)]
5. Alam, S.S.; Chakma, A.; Rahman, M.H.; Bin Mofidul, R.; Alam, M.M.; Utama, I.B.K.Y.; Jang, Y.M. Rf-enabled deep-learning-assisted drone detection and identification: An end-to-end approach. *Sensors* **2023**, *23*, 4202. [[PubMed](#)]
6. Feng, Z.; Guan, N.; Lv, M.; Liu, W.; Deng, Q.; Liu, X.; Yi, W. Efficient drone hijacking detection using two-step GA-XGBoost. *J. Syst. Archit.* **2020**, *103*, 101694. [[CrossRef](#)]
7. Medaiyese, O.O.; Ezuma, M.; Lauf, A.P.; Guvenc, I. Wavelet transform analytics for RF-based UAV detection and identification system using machine learning. *Pervasive Mob. Comput.* **2022**, *82*, 101569. [[CrossRef](#)]
8. U. P. POST. *Energy Consumption of ICT*; Technic Report; UK Parliament: London, UK, 2022.
9. Lange, S.; Pohl, J.; Santarius, T. Digitalization and energy consumption. Does ICT reduce energy demand? *Ecol. Econ.* **2020**, *176*, 106760. [[CrossRef](#)]
10. Fonseca, A.; Kazman, R.; Lago, P. A manifesto for energy-aware software. *IEEE Softw.* **2019**, *36*, 79–82. [[CrossRef](#)]
11. Jiang, C.; Fan, T.; Gao, H.; Shi, W.; Liu, L.; C'erin, C.; Wan, J. Energy aware edge computing: A survey. *Comput. Commun.* **2020**, *151*, 556–580. [[CrossRef](#)]
12. Attzs, M.N.J.; Mahendrakar, T.; Meni, M.J.; White, R.T.; Silver, I. Comparison of Tracking-By-Detection Algorithms for Real-Time Satellite Component Tracking. In *Proceedings of the Small Satellite Conference*, Logan, UT, USA, 7–10 August 2023.
13. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 779–788.
14. Ajmera, Y.; Singh, S.P. Autonomous UAV-based Target Search, Tracking and Following using Reinforcement Learning and YOLOFlow. In *Proceedings of the 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Abu Dhabi, United Arab Emirates, 4–6 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 15–20. [[CrossRef](#)]
15. Brdjanin, A.; Dardagan, N.; Dzical, D.; Akagic, A. Single Object Trackers in OpenCV: A Benchmark. In *Proceedings of the 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, Novi Sad, Serbia, 24–26 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [[CrossRef](#)]
16. Muralidharan, M.; Tamizhnambi, Y.; Arun, A.C.; Premanand, V. Comparative Study of Object Tracking Algorithms for Urban Intersection Risk Analysis Using CSRT, FairMOT, DeepSORT, and ByteTrack. In *Proceedings of the 2025 IEEE International Conference on Computer Vision and Machine Intelligence (CVMI)*, Rourkela, India, 12–13 October 2025; IEEE: Piscataway, NJ, USA, 2025; pp. 1–7. [[CrossRef](#)]
17. Zamanakos, G.; Seewald, A.; Midtiby, H.; Schultz, U. Energy-Aware Design of Vision-Based Autonomous Tracking and Landing of a UAV. In *Proceedings of the 2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan, 9–11 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 294–297. [[CrossRef](#)]

18. Guo, H.; Wu, Q.; Wang, Y. AUHF-DETR: A Lightweight Transformer with Spatial Attention and Wavelet Convolution for Embedded UAV Small Object Detection. *Remote Sens.* **2025**, *17*, 1920. [\[CrossRef\]](#)
19. Li, F.; Zeng, A.; Liu, S.; Zhang, H.; Li, H.; Zhang, L.; Ni, L.M. Lite detr: An interleaved multi-scale encoder for efficient detr. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 18558–18567.
20. Khalili, A.; Rezaei, A.; Xu, D.; Schober, R. Energy-Aware Resource Allocation and Trajectory Design for UAV-Enabled ISAC. In *Proceedings of the GLOBECOM 2023—2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 8–12 December 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 4193–4198. [\[CrossRef\]](#)
21. Seewald, A.; de Marina, H.G.; Midtiby, H.S.; Schultz, U.P. Energy-Aware Planning-Scheduling for Autonomous Aerial Robots. In *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 2946–2953. [\[CrossRef\]](#)
22. Kreciglowa, N.; Karydis, K.; Kumar, V. Energy efficiency of trajectory generation methods for stop-and-go aerial robot navigation. In *Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), 13–16 June 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 656–662.
23. Li, M.; Jia, G.; Gong, S.; Guo, R. Energy consumption model of bldc quadrotor uavs for mobile communication trajectory planning. *TechRxiv* **2022**. [\[CrossRef\]](#)
24. Gasche, S.; Kallies, C.; Himmel, A.; Findeisen, R. Energy aware and safe path planning for unmanned aircraft systems. *arXiv* **2025**, arXiv:2504.03271.
25. Xiong, H.; Hu, J.; Diao, X. Optimize energy efficiency of quadrotors via arm rotation. *J. Dyn. Syst. Meas. Control* **2019**, *141*, 091002. [\[CrossRef\]](#)
26. Morbidi, F.; Bicego, D.; Ryll, M.; Franchi, A. Energyefficient trajectory generation for a hexarotor with dual-tilting propellers. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 6226–6232.
27. Cheng, H.; Lin, L.; Zheng, Z.; Guan, Y.; Liu, Z. An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1732–1738.
28. Kumar, M.; Mondal, S. Recent developments on target tracking problems: A review. *Ocean Eng.* **2021**, *236*, 109558. [\[CrossRef\]](#)
29. Wu, S.; Li, R.; Shi, Y.; Liu, Q. Vision-based target detection and tracking system for a quadcopter. *IEEE Access.* **2021**, *9*, 62043–62054. [\[CrossRef\]](#)
30. Gharsa, O.; Toubia, M.M.; Boumehraz, M.; Agram, N. Autonomous Vision-Based Object Detection and Tracking System for Quadrotor Unmanned Aerial Vehicles. *Sensors* **2025**, *25*, 6403. [\[CrossRef\]](#) [\[PubMed\]](#)
31. Kang, H.C.; Han, H.N.; Bae, H.C.; Kim, M.G.; Son, J.Y.; Kim, Y.K. HSV color-space-based automated object localization for robot grasping without prior knowledge. *Appl. Sci.* **2021**, *11*, 7593.
32. Xu, C.; Su, J.; Wang, T.; Zhang, Y.; Yu, A. Improved Canny Edge Detection Operator. In *Proceedings of the 2023 7th International Conference on Communication and Information Systems (ICCIS), Chongqing, China, 20–22 October 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 135–140. [\[CrossRef\]](#)
33. Wang, Q. Support Vector Machine Algorithm in Machine Learning. In *Proceedings of the 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 24–26 June 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 750–756. [\[CrossRef\]](#)
34. Salman, H.A.; Kalakech, A.; Steiti, A. Random Forest Algorithm Overview. *Babylon. J. Mach. Learn.* **2024**, *2024*, 69–79. [\[CrossRef\]](#) [\[PubMed\]](#)
35. Yang, Y.; Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10), San Jose, CA, USA, 13–16 November 2010*; Association for Computing Machinery: New York, NY, USA, 2010; pp. 270–279.
36. Chu, H. *SiamTrackers: A Repository for Siamese Network-based Tracking Algorithms*; GitHub: San Francisco, CA, USA, 2023. Available online: <https://github.com/HonglinChu/SiamTrackers> (accessed on 10 June 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.