inventions

MDPI

Article

Real-Time Object Localization Using a Fuzzy Controller for a Vision-Based Drone

Ping-Sheng Wang, Chien-Hung Lin and Cheng-Ta Chuang *

Department of Intelligent Automation Engineering, National Taipei University of Technology, Taipei 10608, Taiwan
* Correspondence: ctchuang@ntut.edu.tw

Abstract: This study proposes a drone system with visual identification and tracking capabilities to address the issue of limited communication bandwidth for drones. This system can lock onto a target during flight and transmit its simple features to the ground station, thereby reducing communication bandwidth demands. RealFlight is used as the simulation environment to validate the proposed drone algorithm. The core components of the system include DeepSORT and MobileNet lightweight models for target tracking. The designed fuzzy controller enables the system to adjust the drone's motors, gradually moving the locked target to the center of the frame and maintaining continuous tracking. Additionally, this study introduces channel and spatial reliability tracking (CSRT) switching from multi-object to single-object tracking and multithreading technology to enhance the system's execution speed. The experimental results demonstrate that the system can accurately adjust the target to the frame's center within approximately 1.5 s, maintaining precision within ±0.5 degrees. On the Jetson Xavier NX embedded platform, the average frame rate (FPS) for the multi-object tracker was only 1.37, with a standard deviation of 1.05. In contrast, the single-object tracker CSRT exhibited a significant improvement, achieving an average FPS of 9.77 with a standard deviation of 1.86. This study provides an effective solution for visual tracking in drone systems that is efficient and conserves communication bandwidth. The validation of the embedded platform highlighted its practicality and performance.

Keywords: fuzzy control; DeepSORT; MobileNet; CSRT tracker

Citation: Wang, P.-S.; Lin, C.-H.; Chuang, C.-T. Real-Time Object Localization Using a Fuzzy Controller for a Vision-Based Drone. *Inventions* **2024**, *9*, 14. https://doi.org/10.3390/inventions9010014

Academic Editor: Anastasios Doulamis

Received: 30 October 2023
Revised: 24 December 2023
Accepted: 2 January 2024
Published: 12 January 2024

1. Introduction

Recent developments in software technology, including artificial intelligence (AI) and image processing, along with hardware technology improvements like advanced flight control platforms and embedded systems, have significantly accelerated the evolution of drone applications. Recently, drones have been widely used in military applications for reconnaissance and civilian applications for bridge inspections and tower cleaning. Reducing battery consumption is critical because drones are independent carriers. Ref. [1] introduced the development and hardware implementation of an autonomous battery maintenance mechatronic system. This system can significantly extend the runtimes of small battery-powered drones. Ref. [2] developed an automatic battery replacement mechanism that allows drones to fly continuously without manual battery replacement. Ref. [3] presented a novel design of a robotic docking station for automatic battery exchange and charging. Furthermore, it lessens the computational burden on the drone's embedded system, consequently reducing power consumption.

This is a common application for drones in the outdoor tracking of targets. Ref. [4] proposed an UAVMOT network specialized in multitarget tracking from an unmanned aerial vehicle (UAV) perspective and introduced an ID feature update module to enhance the feature association of an object. Ref. [5] employed the YOLOv4-Tiny algorithm for semantic object detection, which was then combined with a 3D object pose estimation

method and a Karman filter to enhance perceptual performance. Ref. [6] proposed an online multi-object tracking (MOT) approach for UAV systems to address small object detection and class imbalance challenges, which integrates the benefits of deep high-resolution representation networks and data association methods in a unified framework. However, most studies did not mention the bottleneck in the communication between UAVs and ground stations.

This study proposes a drone with visual recognition tracking capabilities that uses deep learning to perform image recognition and real-time algorithm validation in the RealFlight simulator. Most embedded systems in drones have limited computational resources; therefore, we used a lightweight target-detection model. Consequently, this study achieves the functionality of the drone tracking a selected target and enhances the efficiency of information transmission between the drone and the ground station. When the onboard visual recognition system detects a single or several targets to be observed, the UAV immediately notifies the ground station and displays a real-time image of what it has seen.

After the ground station crew selects the primary target to be tracked, the live image is turned off, and the onboard image tracking system locks on to the selected target, simplifying the computation process by converting it to a single-object tracker. The fuzzy controller [7] outputs a corresponding motor control angle for inputs with horizontal or vertical errors, gradually moves the selected tracking target to the center of the image, and continues tracking until the ground station completes the next step of the decision-making process. Because the drone has limited communication bandwidth resources, it only returns simple information regarding the target to the ground station, such as the coordinates, color, and type of the target. This reduces the pressure on the communication bandwidth.

Therefore, our main contribution was creating a drone algorithm validation system using RealFlight as a simulation environment, which can be used as a development and validation environment before the actual flight to simplify the cost of developing algorithms and improve development efficiency. In this system, DeepSORT and MobileNet were selected as lightweight models suitable for developing the system, considering the executability of the actual flight, which is suitable for realizing an embedded platform for the drone. This study also verified the feasibility of the algorithm on Jetson Xavier NX. This study also used CSRT switching from multi-object to single-object tracking with a multithreading technique to enhance the execution speed.

The remainder of this paper is organized as follows. Section 2 explains the methodology, and Section 3 presents the experimental results and discussion. Section 4 concludes this study.

2. Methods

The architecture of the overarching system is shown in Figure 1 and involves several stages. During the initial preprocessing phase, we executed image normalization and resizing procedures on the image obtained using RealFlight. When no target was selected by the user, we employed the single-shot multibox detector (SSD) MobileNet V2 model [8] for object detection and the DeepSORT [9] model for multi-object tracking (MOT) to generate the candidate box. We chose the SSD MobileNet V2 model because it consistently outperformed other lightweight models in terms of accuracy and efficiency.

When a ground station user selects a specific target, the proposed system seamlessly transitions from SSD MobileNet V2 and DeepSORT to the single-object tracker CSRT [10], thereby ensuring real-time pinpoint-accurate tracking.

To regulate the drone movement, we integrated a fuzzy control mechanism that governs navigation based on the tracking results from the previous step. This control mechanism interfaces with ArduPilot, enabling precise and responsive drone control in a RealFlight simulation environment. This integration ensures that the drone responds accurately and promptly to the tracking results, thereby optimizing its movement in relation to the designated target. ArduPilot then controls the drone within the RealFlight

environment using FlightAxis. In this integration, RealFlight handled the physical and graphical simulations, whereas ArduPilot was responsible for converting the output of the fuzzy control into motor signals, which were then transmitted to RealFlight.

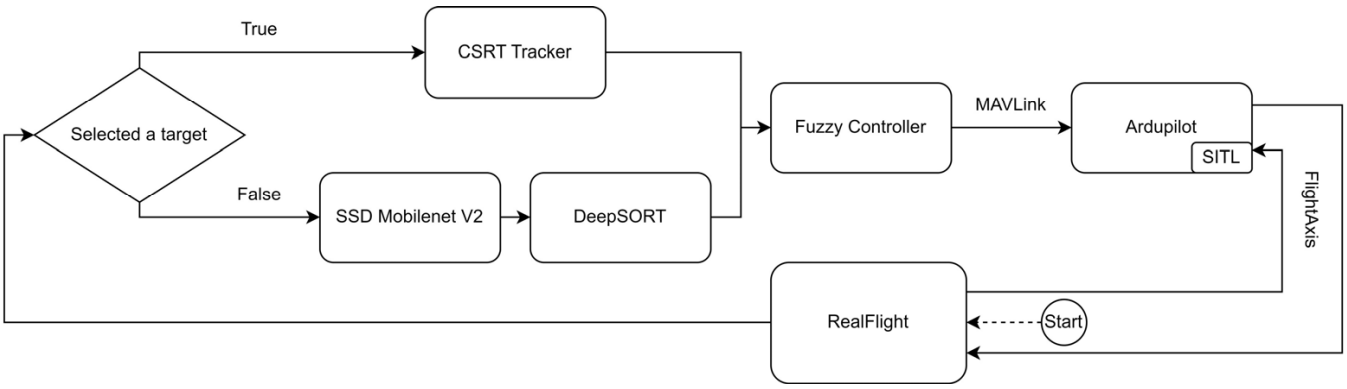


Figure 1. System block diagram.

Finally, feedback was obtained from the tracking methods using RealFlight imagery.

The following sections describe in detail the function of each of the sections in Figure 1, including design ideas and data handling.

2.1. SSD MobileNet V2

SSD MobileNet V2 is a single-stage object detection model with a streamlined network architecture and an innovative depth-wise [11] separable convolution technique. It is widely deployed in low-resource devices such as mobile devices and is highly accurate. The SSD [12] is an object detection model based on a single-stage detector. It uses convolutional neural networks (CNNs) [13] to identify and locate objects in an image directly. An SSD has a faster processing speed than a region-based CNN (R-CNN) [14] because it performs object detection in one step without generating candidate regions. By contrast, MobileNet is a specially designed network architecture that reduces the computational complexity and parameter count of the model while maintaining its high accuracy. It replaces the traditional standard convolution with a depth-wise separable convolution to lower computational costs while preserving good performance.

Therefore, SSD MobileNet V2 combines the single-stage object detection framework of SSD with the lightweight design of MobileNet, thereby making it efficient and accurate for low-resource devices. This model is suitable for object detection tasks in resource-constrained environments, such as mobile devices. Figure 2 presents a performance comparison for the models [15].

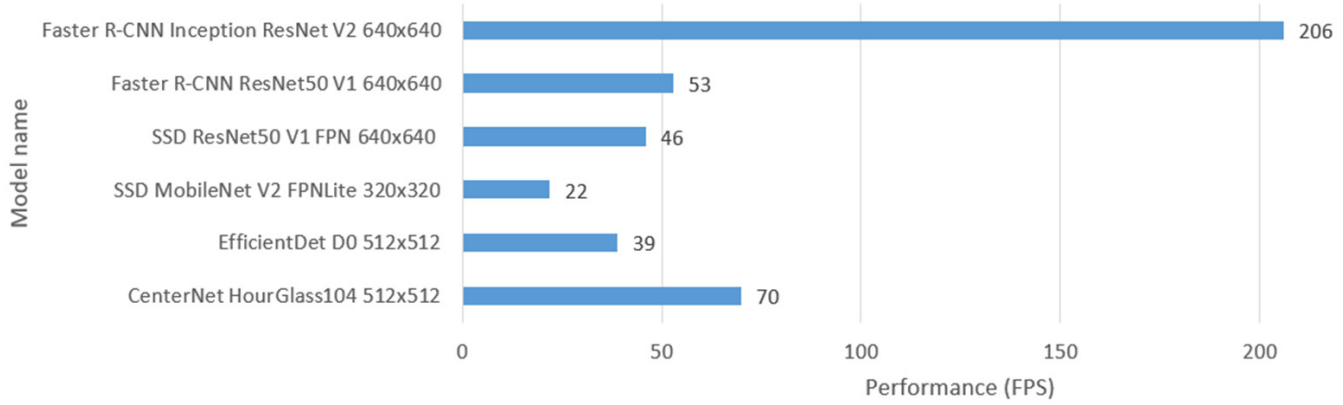


Figure 2. Models' performance comparison.

To process the dataset, we resized and applied the histogram equalization technique [16] on over 100 images. These steps ensured that the images complied with the input requirements of the model and that their contrasts were adjusted.

First, we resized each image to a consistent size of 320 × 320 × 3, meeting the input size requirements of the SSD MobileNet V2 model and ensuring that the model consistently processed different features when handling these images.

Second, we applied the histogram equalization technique, which is a popular image preprocessing technique. This adjusts the contrast of an image, meaning that the features of an image with bright and dark backgrounds and foregrounds can be extracted. This enhances the model's ability to recognize image features during both the training and testing phases. In summary, these preprocessing steps ensured that the dataset images conformed to the input requirements of the model, improving its performance and accuracy.

2.2. DeepSORT

DeepSORT is a tracking algorithm that was first introduced in 2017 and has been extensively used in addressing the MOT problem [17]. Compared with the original SORT algorithm [18], DeepSORT excels while consuming similar levels of computational resources.

In this study, we adopted the DeepSORT algorithm as the central component of the tracking system. By leveraging DeepSORT, unique identifiers can be assigned to objects detected in previous frames, allowing us to continuously track these objects and provide positional information of the target object selected by the user. Subsequently, this tracked information is relayed to the subsequent stages of the tracking system for further analysis and application. This process enables the real-time and accurate tracking of multiple targets while delivering pertinent information for subsequent tasks.

2.3. CSRT

CSRT is a popular object-tracking algorithm. This tracker trains related filters with compressed features, such as HOG [19] and color names. These filters were employed to locate regions around the object's last known position in successive frames. Spatial reliability maps were utilized during this process to modify the filter's support region and choose the tracking area. This approach ensures precise scaling and positioning of the selected region and improves tracking performance for non-rectangular regions or objects. After using MOT and determining the specific target to be tracked, we switched from DeepSORT to CSRT. This is because the continuous use of MOT burdens the entire system, which reduces the speed. Transitioning to a CSRT tracker can effectively resolve this problem. Furthermore, compared to other single-target trackers, CSRT is more accurate.

2.4. Fuzzy Controller

A fuzzy controller was employed to control the drone's direction based on the position of a desired object. We opted for the fuzzy controller because, unlike other controllers such as PID and linear controllers, it does not require knowledge of the system model. The construction of a fuzzy controller can be easily achieved by leveraging human knowledge. The motor can be easily directed toward an object using fuzzy logic [20] by defining simple rules and incorporating variables from the preceding stages of the pipeline. While fuzzy logic might show a less optimal performance compared to a neural network controller, its advantage is its minimal resource consumption in drone control. This characteristic is indispensable for embedded systems with limited computational resources. Figure 3 shows the fuzzy control system block. Initially, the system takes the error between the detected target and the center point as the input. Subsequently, the fuzzy system calculates the angle at which the motor rotates. Subsequently, the system recalculates the error based on the camera on the drone and returns it as feedback. This loop continues until the drone enters the target.

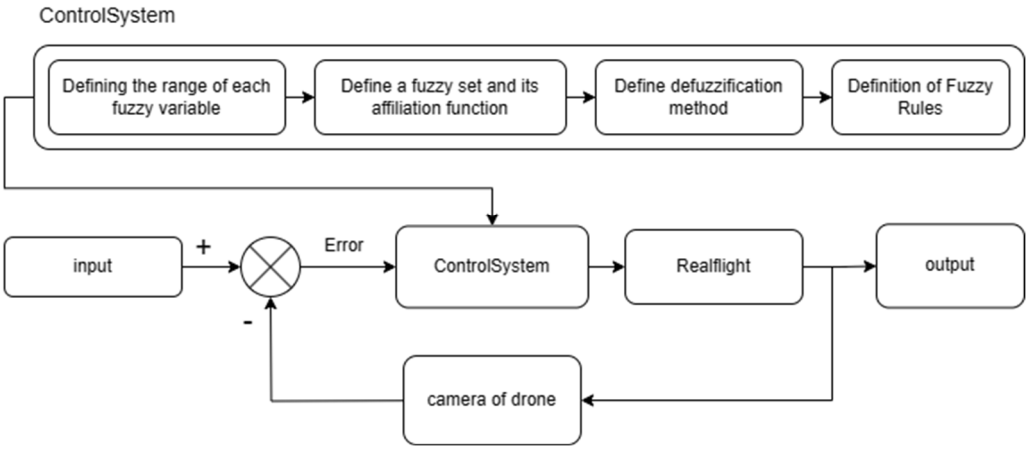


Figure 3. Fuzzy control system's block diagram.

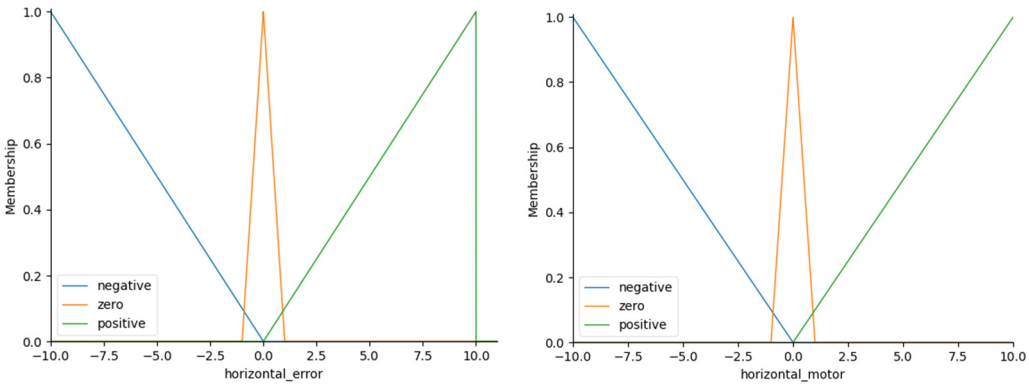
2.4.1. Fuzzy Sets and Membership Functions

Fuzzy sets and membership functions are the fundamental concepts in fuzzy logic. Fuzzy sets are used to represent the degree of affiliation of the elements, whereas membership functions define the degree of the membership function. Smoothness is not a crucial requirement for motor direction control, and our data did not follow a normal distribution. Hence, we opted for triangular membership functions [21] because of their simplicity and robustness. Additionally, the triangular shape facilitated a straightforward association with linguistic terms such as 'low', 'medium', and 'high'. Figure 4 shows the membership functions of the fuzzy variables. Equation (2) shows the fuzzy variables in Equation (1).

$$\text{Negative}: \; VerticalError_{negative}(x) = \begin{cases} 1 & if\ x < 0 \\ \frac{x-10}{0-10} & if\ 0 \le x < 10 \\ 0 & otherwise \end{cases}$$

$$\text{Zero}: \quad VerticalError_{zero}(x) = \begin{cases} 0 & if\ x < -1\ or\ x > 1 \\ \frac{x+1}{0+1} & if -1 \le x \le 0 \\ \frac{1-x}{1-0} & if\ 0 \le x \le 1 \\ 0 & otherwise \end{cases} \tag{1}$$

$$\text{Positive}: \; VerticalError_{positive}(x) = \begin{cases} 0 & if\ x < 0 \\ \frac{x}{10} & if\ 0 \le x < 10 \\ 1 & otherwise \end{cases}$$



(**a**) (**b**)

Figure 4. Fuzzy membership functions charts for (**a**) horizontal error and (**b**) horizontal motors.

2.4.2. Fuzzy Rules

Fuzzy rules are a fundamental component of fuzzy logic used to describe and control system behavior, particularly in handling uncertainty and fuzziness. Each fuzzy rule typically consists of two parts: an antecedent which describes the input conditions and a consequent which defines the actions to be performed. These rules enable systems to make appropriate decisions in complex scenarios. The fuzzy control parameters are listed in Table 1.

Table 1. Fuzzy control table.

	Horizontal Error			
		Positive	Zero	Negative
Vertical error	Positive	(Negative, Positive)	(Negative, Zero)	(Negative, Positive)
	Zero	(Zero, Negative)	(Zero, Zero)	(Zero, Positive)
	Negative	(Positive, Negative)	(Positive, Zero)	(Positive, Positive)

2.4.3. Defuzzification

Defuzzification is a crucial step in fuzzy control and is used to convert fuzzy outputs into clear and definite numerical values. In our scenario, we employed the center of gravity (CoG) method for defuzzification because of its sensitivity to the distribution of membership values across the entire range. This method is considered reasonable because it considers the full distribution of the membership values. It assigns more weight to the center of mass or the peak of the membership functions, reflecting the overall trend or concentration of values within the fuzzy set.

For example, when the vertical error is 0.5, based on the fuzzy sets and fuzzy logic defined earlier, we obtain a fuzzy decision, as illustrated in Figure 5. Subsequently, by employing the CoG method from Equation (2), we derive the following integral Equation (3). Solving this equation yields x as −2.06137703781. This implies that the fuzzy controller will adjust the drone's camera downward by −2.06137703781 degrees.

$$Center\,of Gravity = \frac{\int_a^b x \cdot A(x)\,dx}{\int_a^b A(x)\,dx} \tag{2}$$

$$\frac{\int_{-10}^{-0.95} -0.05x dx + \int_{-0.95}^{-0.5} (x+1)x dx + \int_{-0.5}^{0.5} 0.5x dx + \int_{0.5}^{1} (-x+1)x dx}{\int_{-10}^{-0.95} -0.05 dx + \int_{-0.95}^{-0.5} (x+1) dx + \int_{-0.5}^{0.5} 0.5 dx + \int_{0.5}^{1} (-x+1) dx} \tag{3}$$

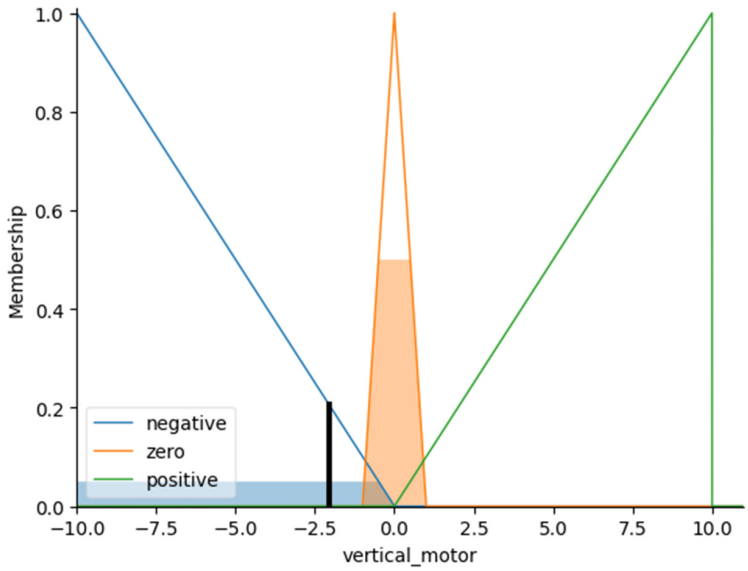


Figure 5. Result of fuzzy decision.

We used the "scikit-fuzzy" Python library [22] to implement fuzzy control effectively within the Python environment. This package provides a comprehensive toolkit for developing and applying fuzzy-logic systems. In our setting, if the drone needs to turn up or right, the fuzzy logic outputs positive values; otherwise, it outputs negative values. Based on these two variables, we designed a set of nine fuzzy rules to infer the appropriate output variables. These output variables serve as signals for the vertical and horizontal motors, calculate the error at the center point of the camera, and provide these motor signals to the loop (SITL), enabling it to adjust the drone's direction toward the designated target accurately. For instance, as illustrated in Table 1, when errors occurred in the positive direction for both axes, we provided a negative output to both the horizontal and vertical motors to rectify the direction. Similarly, when the vertical and horizontal errors were negative and positive, respectively, we outputted a positive signal to the vertical motor and a negative signal to the horizontal motor.

2.5. ArduPilot

ArduPilot:Copter-4.4 [23] is a notable open-source autopilot software suite for various autonomous crewless vehicles like rovers, copters, and boats. It includes powerful tools such as the SITL flight simulator, specifically designed for aircraft, helicopters, drones, and other flying vehicles, enabling simulation without hardware. SITL also interfaces with specific virtual environments, enabling more realistic simulations. This approach uses RealFlight Evolution as the virtual environment, and ArduPilot to receive the output from a fuzzy controller. Our control over the SITL was facilitated by Mavlink's [24] message transmission, which is a communication protocol commonly employed in crewless vehicles. All drone statuses and sensor data were accessible throughout the process using the widely used MissionPlanner-1.3.81 provided by Ardupilot. This software allowed us to monitor and analyze the drone's performance and sensor readings and launch the SITL effectively.

2.6. RealFlight

RealFlight Evolution v10.00.059 is a cutting-edge simulation software explicitly designed for planes and copters that enables users to experience their flights within a virtual environment. Notably, it extends its support to SITL users, facilitating a seamless integration between RealFlight and SITL simulations.

By utilizing the FlightAxis I/O interface, RealFlight can establish a seamless connection with SITL, enabling the visualization and precise simulation of physical models within the RealFlight environment. This integration creates a feedback loop in which RealFlight transmits essential flight parameters, including the vehicle's attitude, velocity, and position, to SITL. Subsequently, SITL processes the data and returns the corresponding control signals to RealFlight, thereby completing the closed-loop simulation.

Algorithm 1 lists the pseudocodes corresponding to the system blocks. At the outset, if the user has not chosen a specific target, the trackedObjectCoordinate will default to false. Subsequently, the primary loop commences, capturing the droneImage from the camera and subjecting it to image preprocessing. The subsequent steps depend on the trackedObjectCoordinate variable, which signifies whether a specific target has been selected for tracking.

When a target is not selected by the user, a multi-object tracker, combining SSDMobileNetV2 and DeepSORT, is activated. This persists until the user designates one of the targets from the DeepSort output. Upon selection, the trackedObjectCoordinate is updated with the coordinates of the chosen object, triggering a switch to the CSRT object tracker in line 5.

When the CSRT tracker is engaged, it pinpoints the selected target within the current preprocessed image, utilizing the tracked object identified in line 15. Following this, the verticalError and horizontalError are calculated, representing the deviation between the camera's center and the chosen target. Subsequently, employing the fuzzy controller, the control signals are computed. These signals were then transmitted to ArduPilot to direct the

movement of the drone and align it with the target. This alignment persisted until the user intervened to halt the process. Upon interruption by the user, trackedObjectCoordinate reverts to *None*, prompting the reactivation of the MOT within the pipeline.

Algorithm 1 Pseudo code of the system block

```
trackedObjectCoordinate = None;
while true do
    droneImage = CaptureDroneImage();
    preprocessedImage = PreprocessImage(droneImage);
    if trackedObjectCoordinate ! = None then
        trackedObjectCoordinate = CSRTObjectTracker(preprocessedImage,
trackedObjectCoordinate);
        verticalError, horizontalError = computeErrors(trackedObjectCoordinate)
        controlSignals = FuzzyControl(verticalError, horizontalError);
        TransmitToArdupilot(controlSignals);
        if UserInterrupt() then
          trackedObjectCoordinate = None;
  else
        detectedObjects = SSDMobilenetV2(preprocessedImage);
        trackedObjectsCoordinate = DeepSORT(detectedObjects);
        if UserSelectTarget() then
          trackedObjectCoordinate = trackedObjectsCoordinate[ SelectedId() ];
```

2.7. Multithreading

In our experiment, we encountered a challenge involving the simultaneous control of a drone while performing object detection and tracking. However, because of the inherent nature of Python's line-by-line execution, significant latency issues arise when the system attempts to execute both tasks concurrently.

This delay may stem from the code's execution time, particularly in scenarios where real-time control of the drone and implementation of object detection and tracking are required. Given Python's interpretive nature, it may struggle to deliver sufficient performance to satisfy these demands.

To overcome this issue, we adopted a multithreading approach [25]. This enables the system to execute different tasks simultaneously, thereby reducing the latency. Multithreading offers several advantages in program development. First, it allows the execution of multiple tasks simultaneously, enhancing the program performance, particularly in scenarios which require the simultaneous handling of multiple tasks. Second, multithreading enables the sharing of the same memory space, facilitating more efficient resource utilization. Third, it prevents system blockage caused by a single thread, which could impede the execution of other threads, thus enhancing the program's flexibility and responsiveness. In addition, multithreading simplifies the program design and makes it easier to handle errors in each thread. Finally, multithreading enhances performance when dealing with IO-intensive tasks, as other threads can continue execution while one thread waits for IO operations to complete, thereby reducing the overall execution time.

3. Experiment

The subsequent section describes the experimental results of the system, including the initial assumptions, database, training conditions, simulation results, and performance analysis.

3.1. Assumption

In our simulation, we assumed that an unmanned aerial vehicle (UAV) initiates the tracking of a target from 600 m away. The UAV's tracking pursuit persists until the target tracking algorithm ceases to locate the object or until the user opts to cease tracking. It is imperative that the boat maintains a moderate speed to ensure optimal tracking performance. The fuzzy input error and output angle is -10 to 10. Additionally, within the

realistic simulation, we considered daytime conditions to be essential; without adequate lighting, the camera would fail to capture images effectively.

3.2. Dataset

We employed a pretrained model trained on the PASCAL Visual Object Classes (VOC) [26] dataset to facilitate the training of SSD MobileNet V2 for ship detection within frames. The PASCAL VOC dataset is a well-known resource in computer vision and comprises images of 20 distinct object classes along with their associated bounding boxes. It is widely used for various tasks, including object detection and image segmentation.

However, given that our specific target for object tracking was ships and considering the limited representation of ship images within the PASCAL VOC dataset, we recognized the need to augment our training data. To address this issue, we created an extensive set of over 100 ship images, capturing various angles and perspectives in a virtual environment. This strategy allowed us to complement the PASCAL VOC dataset with a more diverse and relevant collection of ship images, thereby better aligning our model training with the intricacies of real-world ship detection.

By leveraging comprehensive object class information from the PASCAL VOC dataset and fine-tuning our model with our augmented ship dataset, we aimed to enhance its generalization capabilities, expedite convergence during training, and mitigate the risks associated with overfitting. This hybrid approach was designed to effectively bridge the gap between the generic object classes in the PASCAL VOC dataset and the specific demands for ship detection in our simulator scenario.

3.3. Hardware Setup

Hardware such as computers were crucial in our experiment. Our computer was responsible for running the RealFlight simulator, Ardupilot SITL, and our tracking solution simultaneously. The performance of our pipeline was significantly affected by resource-intensive processes, such as RealFlight, which also consumed graphics processing unit (GPU) resources for rendering scenes. Nevertheless, our setup was well-suited for simulating resource-constrained mobile environments such as drones. The hardware configuration used in our experiment is described in detail below.

- CPU: Intel Core i5-8400
- GPU: NVIDIA GTX 1060
- RAM: 16 GB

Although our personal computer (PC) setup may not have been exceptionally powerful, we took the measures in the previous section to optimize resource usage, and we believe that our configuration was adequate to run our pipeline effectively.

3.4. Training

The feature pyramid network (FPN) SSD MobileNet V2 model is used as the base model. This single-shot detector model can complete object detection tasks in a single forward propagation. The model uses MobileNet V2 as the base network, which has a higher running speed and lower computing power. In addition, the model contains a feature pyramid network (FPN) [27] to combine high-level semantic information with low-level detailed information to generate feature maps of different scales that can better handle objects of different scales. Thus, the FPN SSD model handles small-object detection more accurately than the traditional models. Integrating the FPN into our model significantly boosted its performance in image recognition tasks, particularly in scenarios involving targets of varying scales and complex appearances.

The eIQ toolkit [28] was used for the fine-tuning. Fine-tuning involves adjusting the model parameters based on the data of a new task using a pretrained model.

We used fine-tuning rather than transfer learning because fine-tuning can quickly use the knowledge learned in the pretrained model and fine-tune it according to the data of the new task to adapt better to the new task. Although considerable computing resources

and time are required for fine-tuning, fine-tuned systems can fully utilize the knowledge learned in the pretrained model.

Transfer learning typically requires fewer computing resources and less time because it only needs to train the output model. However, using the knowledge learned in the pretrained model may not be feasible; thus, optimal performance may not be achieved. The losses for training and evaluation are shown in Figure 6. The training parameters are as follows:

- Weight Initialization: PASCAL VOC
- Learning Rate: 0.001
- Learning Rate Decay: Disabled
- Batch Size: 10

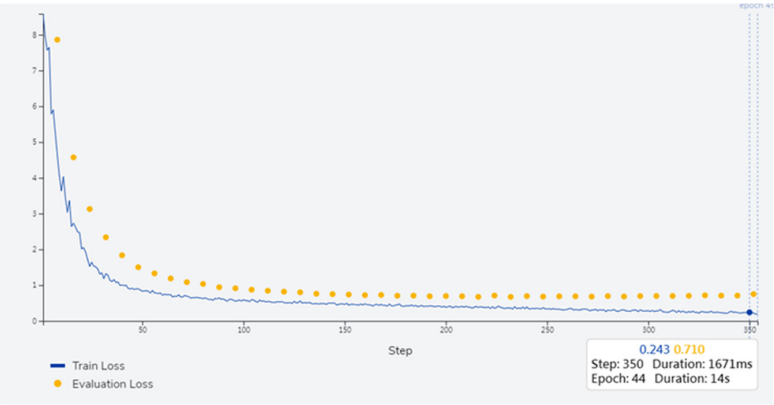


Figure 6. Loss chart for training and evaluation.

The predicted results for the images in the test set are shown in Figure 7. The blue and green rectangles represent the ground truth and decoded output of the FPN SSD MobileNet V2, respectively.

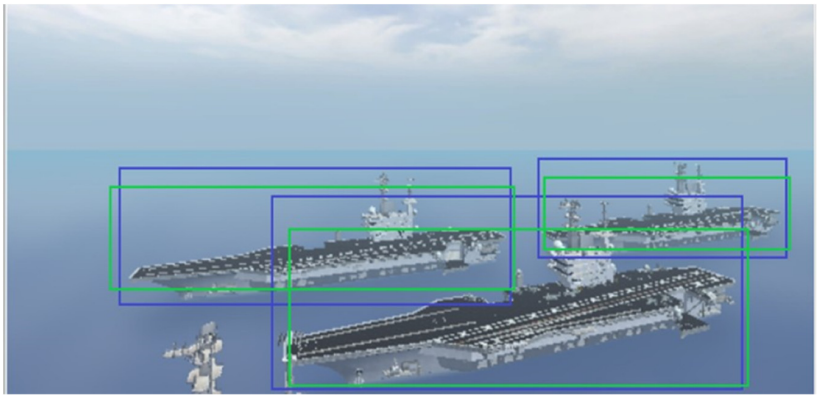


Figure 7. Testing set result of FPN SSD MobileNet V2.

3.5. Simulation

In our simulation, the drone was initially positioned on an aircraft carrier using the aforementioned methods. It was then commanded to fly toward a specific position and altitude. Next, the yaw was randomly adjusted to ensure that the aircraft carrier appeared in the field of view (FOV).

Object tracking was initiated once this setup was complete, allowing the user to select a desired target for the drone to focus on, based on the unique ID provided by DeepSORT. Upon selecting the desired target, the MOT (SSD MobileNet V2 + DeepSORT) was deactivated, and the single-object tracker CSRT seamlessly replaced it. Concurrently, the fuzzy controller was activated, and the pitch and yaw of the drone were adjusted

accordingly to orient it toward the target. When the user opted to stop the tracking process, the drone immediately halted its movement and presented windows to select a new tracking target. A drone that uses SITL with RealFlight is shown in Figure 8.

Figure 8. The drone using SITL with RealFlight.

3.6. Result

We initially used the SSD MobileNet V2 to detect three aircraft carriers, each of which was assigned a unique ID using DeepSORT, as shown in Figure 9. Upon selecting Target ID 2 from the dialog box, we seamlessly transitioned the object-tracking method from SSD MobileNet V2 and DeepSORT to CSRT, which is a single-object tracker. Simultaneously, the fuzzy controller was engaged to ensure the precise navigation of the drone toward the aircraft carrier associated with ID 2, as shown in Figure 10. As shown in Figure 11, the entire process converges after approximately 15 iterations, and each fuzzy controller output takes approximately 0.1 s. Although the drone experienced minor shaking during this adjustment, owing to the controller's sensitivity, it effectively completed the process within an impressive 1.5 s. Furthermore, once the drone aligned with the target, we promptly halted the tracking process and selected a new tracking target, demonstrating the drone's ability to seamlessly change targets and align with the new target within 1.5 s. These results demonstrated the practicality and effectiveness of the proposed method.

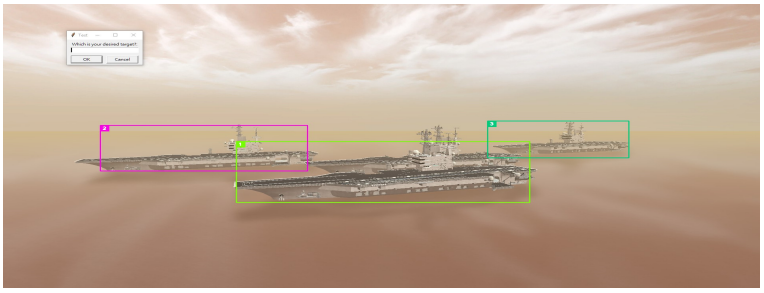


Figure 9. Drone perspective before object tracking.

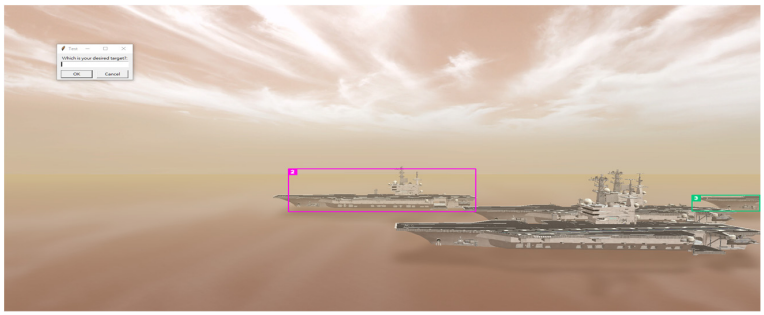


Figure 10. Drone perspective after object tracking.

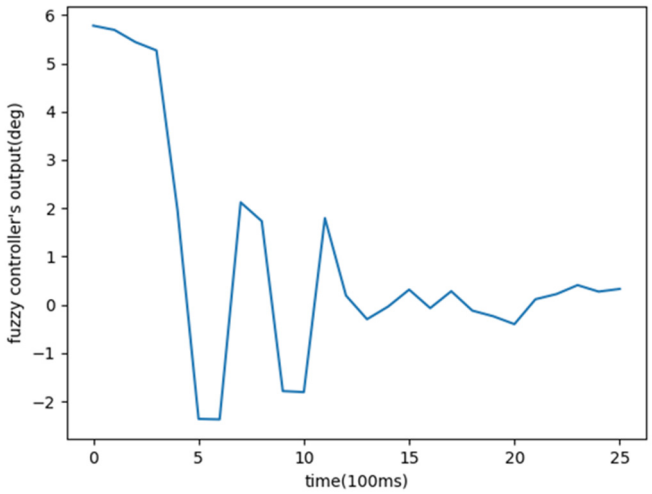


Figure 11. Output of the fuzzy controller over time.

3.7. Performance Analysis

We further evaluated the performance enhancements achieved by transitioning from an MOT (SSD MobileNet V2 + DeepSORT) to a single-object tracker, CSRT. A comparison of the number of FPS is shown in Figure 12.

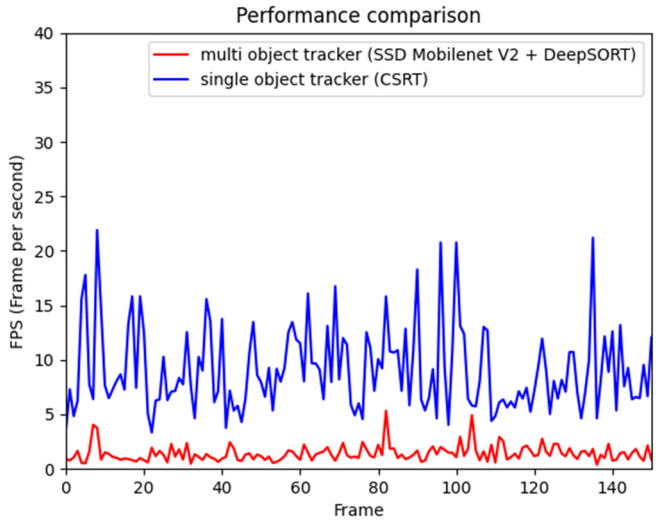


Figure 12. Performance comparison between utilizing MOT (SSD MobileNet V2 + DeepSORT) and a single-object tracker (CSRT).

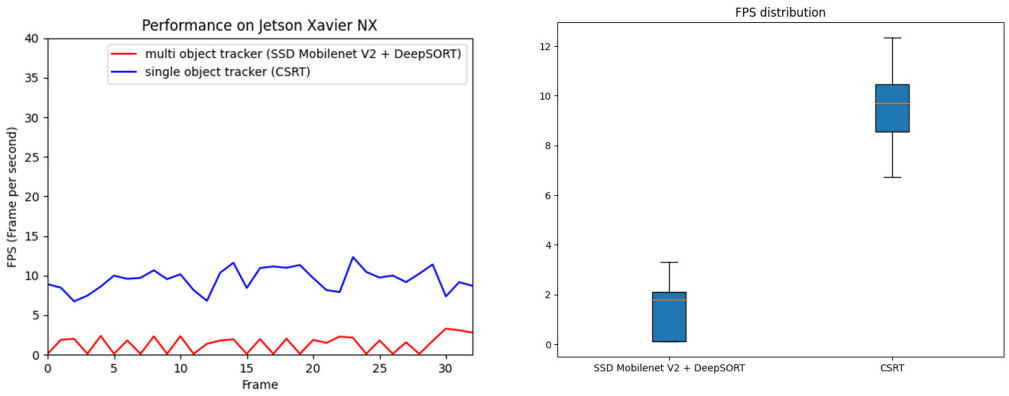
In summary, the average FPS for the MOT and the single-object tracker were 1.45 FPS and 9.28 FPS, respectively. Our switch to a single-object tracker significantly improved the performance of our pipeline, achieving a six-fold increase in FPS. Using this high-performance tracking pipeline, our drone could effectively track the desired targets in real time.

Based on our investigation, two main factors contributed to the sluggish performance of the proposed MOT. First, both SSD MobileNet V2 and DeepSORT require GPU resources for computation, and a substantial portion of our GPU capacity is allocated to the RealFlight simulators. Second, the current TensorFlow Lite op kernels are optimized for ARM processors rather than for CUDA GPUs. Consequently, there was a performance drop when running TensorFlow Lite models such as SSD MobileNet V2 on a GTX 1060 GPU.

3.8. Implementation on Embedded System

In addition, we implemented our image recognition pipeline on an embedded system called Jetson Xavier NX [29]. The lightweight algorithms and models central to our approach ensured an optimal performance on this embedded platform. With our implementation, the drone's image recognition capabilities, including the tracking of multiple objects and dynamic centering based on user selection, were seamlessly integrated into the Jetson Xavier NX environment. During the migration to this pipeline, the only issue encountered was an out-of-memory error. However, we promptly addressed this challenge by optimizing memory allocation and ensuring the correct order of package imports.

To assess the performance of Jetson Xavier NX, we analyzed the FPS for our pipeline, as shown in Figure 13a. As depicted in the results, our approach of transitioning to a single-object tracker once again significantly enhanced the performance. The average FPS for the multi-object trackers on the Jetson Xavier NX was 1.37. In contrast, adopting the single-object tracker CSRT led to a substantial improvement, achieving an average FPS of 9.77 on the Jetson Xavier NX. The standard deviation for our MOT was 1.05, whereas that for the single-object tracker was 1.86. Figure 13b shows the FPS distributions.



(**a**) (**b**)

Figure 13. Performance (FPS) of our pipeline on Jetson Xavier NX, (**a**) Performance on Jetson Xavier NX; (**b**) FPS distribution.

4. Discussion

This section discusses the experimental results, including the challenges encountered, a comparison of the proposed methodology with other studies, and directions for future improvement.

4.1. Yaw-Induced Translation in Drone Control

In the initial stages of our drone control endeavors, we encountered a noteworthy challenge associated with adjusting the yaw of an UAV. Specifically, as we attempted to align the drone with a target, necessitating yaw adjustments, we observed an unexpected lateral movement of the quadcopter. This prompted us to seek insights from domain experts, leading to the realization that both fixed-wing and quadcopter drones tend to experience movement or alterations in their trajectories when adjusting the yaw. This phenomenon is based on the principles of aerodynamics and control dynamics.

Adjusting the yaw of a drone involves changing its orientation around its vertical axis. In quadcopters, altering the yaw induces a torque that, in turn, leads to angular acceleration, causing the drone to move laterally or change its trajectory. The aerodynamic forces produced during yaw adjustments interact with the drone's inherent stability mechanisms, leading to unintended translational movements. This highlights the need to consider these dynamics in drone control systems, specifically for tasks requiring precise positioning or target tracking.

4.2. Comparative Analysis

Table 2 shows that, despite the adoption of relatively lightweight models on the Jetson Xavier NX, the computational speed did not reach the desired level. This perspective was further corroborated in another relevant study, which demonstrated that relying solely on lightweight models cannot adequately address performance issues in situations with limited hardware resources. Furthermore, it highlights the critical importance of implementing a mechanism for switching between multi-target and single-target tracking, particularly in scenarios in which both types of tracking are required simultaneously.

Table 2. Comparative analysis of papers.

	Our Paper		**Run Your 3D Object Detector on NVIDIA Jetson Platforms: A Benchmark Analysis [30]**	
Preprocessing	Method 1	Method 2	Method 1	Method 2
	Yes	Yes	No	No
Object detection	SSD	No	CIA-SSD	SE-SSD
Object tracking	DeepSORT	CSRT	No	No
FPS	1.3760	10.3923	3.12	3.17

4.3. Limitations in the Presentation of Papers

The simulation environment may not achieve convergence in response to moving objects or faster movement than a drone, which is an important issue which must be addressed. Furthermore, the inability to hover while tracking targets stems from the drone's physical mechanism limitations, representing one of the challenges faced in our study.

4.4. Future Work

The following section describes several directions for future improvements including controller design, optimization of SSD models, and data augmentation.

4.4.1. Refine the Fuzzy Controller

In addition, we intend to refine the fuzzy controller's performance to minimize the drone's shaking during adjustments. This objective can be accomplished by carefully adjusting the controller's sensitivity parameters and integrating supplementary feedback mechanisms to ensure smoother and more stable movement.

4.4.2. Optimize the SSD MobileNet V2 Model

We used SSD MobileNet V2 to fine-tune target detection accuracy. Adjusting the yaw of the drone resulted in the disappearance of the detection link to the aircraft carrier in the middle, which also led to inconsistencies in the object IDs during tracking with DeepSORT. Therefore, the optimization of our detection model can solve these problems.

4.4.3. Augmented Dataset

The diversity of the target objects for SSD MobileNet V2 detection can be increased by expanding the dataset. Our dataset includes only aircraft carriers. In the future, we can add various types of ships at sea to provide a wider selection of targets and improve the accuracy when different types of vessels overlap in the frame.

5. Conclusions

In this study, we proposed a system with visual recognition and tracking capabilities. Because the communication bandwidth of UAVs, including drones, is considerably valuable, the proposed system can transmit the simple features of a target identified during flight back to the ground station for decision-making by ground station personnel. This reduces

the requirement for communication bandwidth. We built a drone algorithm validation system using the SITL simulator and RealFlight as the simulation environments. When a candidate target is detected, blurring technology is used to control the flight of the drone to lock the target in the center of the frame. The proposed fuzzy control technology allows for the completion of the locking function within approximately 1.5 s. This study mentions enhancing the working speed of the system using lightweight modeling, multi-objective to single-objective tracking with CSRT switching, and multithreading. The proposed target-tracking architecture was validated using the Jetson Xavier NX embedded platform. On the embedded platform, the performance of the CSRT was dramatically improved, with an average frame rate of 9.77 frames per second and a standard deviation of 1.86. In the future, we aim to increase the number of targets that can be recognized by flight verification in the field.

Author Contributions: Conceptualization, C.-T.C.; funding, C.-T.C.; supervision, C.-T.C.; methodology, P.-S.W. and C.-H.L.; software, P.-S.W. and C.-H.L.; validation, P.-S.W. and C.-H.L.; writing—original draft, P.-S.W. and C.-H.L.; writing—review and editing, P.-S.W., C.-H.L. and C.-T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by HIDES, Inc.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The company has a problem that needs to be solved and entrusted our lab to solve the problem. All the experimental design, execution, data collection and analysis are done by our lab. The company agrees to publish the results of this study. There is no other potential conflict of interest between the funder and this study. The authors declare no conflicts of interest.

References

1. Ure, N.K.; Chowdhary, G.; Toksoz, T.; How, J.P.; Vavrina, M.A.; Vian, J. An Automated Battery Management System to Enable Persistent Missions with Multiple Aerial Vehicles. *IEEE/ASME Trans. Mechatron.* **2015**, *20*, 275–286. [CrossRef]
2. Fujii, K.; Higuchi, K.; Rekimoto, J. Endless Flyer: A Continuous Flying Drone with Automatic Battery Replacement. In Proceedings of the 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, Vietri sul Mare, Italy, 18–21 December 2013; pp. 216–223.
3. Wu, Y.; Teng, M.; Tsai, Y. Robot Docking Station for Automatic Battery Exchanging and Charging. In Proceedings of the IEEE International Conference on Robotics and Biomimetics, Bangkok, Thailand, 21–26 February 2009; pp. 1043–1046.
4. Liu, S.; Li, X.; Lu, H.; He, Y. Multi-Object Tracking Meets Moving UAV. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 8876–8885.
5. Lo, L.-Y.; Yiu, C.H.; Tang, Y.; Yang, A.-S.; Li, B.; Wen, C.-Y. Dynamic Object Tracking on Autonomous UAV System for Surveillance Applications. *Sensors* **2021**, *21*, 7888. [CrossRef]
6. Huang, W.; Zhou, X.; Dong, M.; Xu, H. Multiple Objects Tracking in the UAV System Based on Hierarchical Deep High-Resolution Network. *Multimed. Tools Appl.* **2021**, *80*, 13911–13929. [CrossRef]
7. Zadeh, L.A. Fuzzy Algorithms. *Inf. Control* **1968**, *12*, 94–102. [CrossRef]
8. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. Mobilenetv2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
9. Wojke, N.; Bewley, A.; Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649.
10. Lukežic, A.; Vojír, T.; Zajc, L.C.; Matas, J.; Kristan, M. Discriminative Correlation Filter with Channel and Spatial Reliability. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 4847–4856.
11. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
12. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot Multibox Detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
13. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions. *J. Big Data* **2021**, *8*, 53. [CrossRef]

14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
15. TensorFlow Models. Available online: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (accessed on 30 October 2023).
16. Han, J.-H.; Yang, S.; Lee, B.-U. A Novel 3-D Color Histogram Equalization Method with Uniform 1-D Gray Scale Histogram. *IEEE Trans. Image Process.* **2011**, *20*, 506–512. [CrossRef]
17. Luo, W.; Xing, J.; Milan, A.; Zhang, X.; Liu, W.; Kim, T.-K. Multiple Object Tracking: A Literature Review. *Artif. Intell.* **2021**, *293*, 103448. [CrossRef]
18. Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; Upcroft, B. Simple online and realtime tracking. In Proceedings of the 2017 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468.
19. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005; pp. 886–893.
20. Kontogiannis, D.; Bargiotas, D.; Daskalopulu, A. Fuzzy Control System for Smart Energy Management in Residential Buildings Based on Environmental Data. *Energies* **2021**, *14*, 752. [CrossRef]
21. Peckol, J.K. *Introduction to Fuzzy Logic*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2021.
22. Scikit-Fuzzy. Available online: https://pythonhosted.org/scikit-fuzzy/overview.html (accessed on 30 October 2023).
23. ArduPilot. Available online: https://ardupilot.org/ (accessed on 30 October 2023).
24. Koubaa, A.; Allouch, A.; Alajlan, M.; Javed, Y.; Belghith, A.; Khalgui, M. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access* **2019**, *7*, 87658–87680. [CrossRef]
25. Kwak, H.; Lee, B.; Hurson, A.R.; Yoon, S.-H.; Hahn, W.-J. Effects of Multithreading on Cache Performance. *IEEE Trans. Comput.* **1999**, *48*, 176–184. [CrossRef]
26. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.M.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
27. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
28. NXP. Available online: https://www.nxp.com/design/software/eiq-ml-development-environment:EIQ (accessed on 30 October 2023).
29. NVIDIA. Available online: https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-xavier-nx/ (accessed on 30 October 2023).
30. Choe, C.; Choe, M.; Jung, S. Run Your 3D Object Detector on NVIDIA Jetson Platforms: A Benchmark Analysis. *Sensors* **2023**, *23*, 4005. [CrossRef] [PubMed]