

Article

FastUAV-NET: A Multi-UAV Detection Algorithm for Embedded Platforms

Amir Yavariabdi ^{1,*} , Huseyin Kusetogullari ^{2,3} , Turgay Celik ^{4,5}  and Hasan Cicek ¹

¹ Mechatronics Engineering Department, Faculty of Engineering, KTO Karatay University, 42020 Konya, Turkey; hasan.cicek@karatay.edu.tr

² Department of Computer Science, Blekinge Institute of Technology, 371 41 Karlskrona, Sweden; huseyin.kusetogullari@bth.se

³ School of Informatics, Skövde University, 541 28 Skövde, Sweden

⁴ School of Electrical and Information Engineering and the Wits Institute of Data Science, University of the Witwatersrand, Johannesburg 2000, South Africa; celikturgay@gmail.com

⁵ School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China

* Correspondence: amir.yavariabdi@karatay.edu.tr

Abstract: In this paper, a real-time deep learning-based framework for detecting and tracking Unmanned Aerial Vehicles (UAVs) in video streams captured by a fixed-wing UAV is proposed. The proposed framework consists of two steps, namely intra-frame multi-UAV detection and the inter-frame multi-UAV tracking. In the detection step, a new multi-scale UAV detection Convolutional Neural Network (CNN) architecture based on a shallow version of You Only Look Once version 3 (YOLOv3-tiny) widened by Inception blocks is designed to extract local and global features from input video streams. Here, the widened multi-UAV detection network architecture is termed as FastUAV-NET and aims to improve UAV detection accuracy while preserving computing time of one-step deep detection algorithms in the context of UAV-UAV tracking. To detect UAVs, the FastUAV-NET architecture uses five inception units and adopts a feature pyramid network to detect UAVs. To obtain a high frame rate, the proposed method is applied to every n^{th} frame and then the detected UAVs are tracked in intermediate frames using scalable Kernel Correlation Filter algorithm. The results on the generated UAV-UAV dataset illustrate that the proposed framework obtains 0.7916 average precision with 29 FPS performance on Jetson-TX2. The results imply that the widening of CNN network is a much more effective way than increasing the depth of CNN and leading to a good trade-off between accurate detection and real-time performance. The FastUAV-NET model will be publicly available to the research community to further advance multi-UAV-UAV detection algorithms.



Citation: Yavariabdi, A.; Kusetogullari, H.; Celik, T.; Cicek, H. FastUAV-NET: A Multi-UAV Detection and Tracking Algorithm for Embedded Platforms. *Electronics* **2021**, *10*, 724. <https://doi.org/10.3390/electronics10060724>

Academic Editor: Byung-Gyu Kim

Received: 9 February 2021

Accepted: 6 March 2021

Published: 19 March 2021

Keywords: deep learning; CNN; detection and tracking; Unmanned Aerial Vehicle; UAVs pursuit-evasion

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 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) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

An Unmanned Aerial Vehicle (UAV) is a low-to-medium altitude, light weight, small, low cost, and slow aircraft that can be controlled either remotely by a human operator or autonomously via an onboard computer. UAVs have simple mechanical structure as well as they can fly outdoor or indoor at various speeds, hover over a point of interest, perform maneuver in close distance to obstacles, and pursue difficult tasks without putting operator's health and safety in danger [1]. Due to the aforementioned advantages, UAVs have been employed in a wide range of military and civil applications such as surveillance [2], aerial photography [3], infrastructure inspection [4], express delivery [5], pesticides spraying [6], disaster monitoring and rescue [7], and many others [8]. Generally, many of these applications have employed individual UAVs, but in recent years, there has been tremendous escalation in development of applications using multiple UAVs and UAV swarms. Therefore, currently, the main research effort in this context is directed toward developing

unmanned aerial systems for UAVs cooperation, multi-UAV autonomous navigation, and UAVs pursuit-evasion problems. This paper focuses on multi-UAV pursuit problem.

In UAVs pursuit problem, the vital task is to detect and track a target or leader UAV using a tracker or follower UAV. Therefore, the strategies in this context can be either competitive or collaborative between UAVs. For detection and tracking purposes, UAVs must sense their environment. To achieve this, optical sensors are the most suitable ones due to: (1) they are cheap and light; (2) they provide rich information about surrounding scene; and (3) they have low power consumption. To develop vision-based UAVs for pursuit purpose, a real-time detection algorithm must be developed to localize and recognize one or multiple UAVs appearing in video streams captured by tracker/follower UAV. However, this is a very challenging problem because of several issues: (1) large variations in target or leader UAV's visual appearance caused by many factors such as illumination changes, weather conditions, distance between UAVs, UAVs' orientation, occlusion, and specular light (see Figure 1a–f); (2) existence of shadows (see Figure 1g); (3) low contrast between UAV and background (see Figure 1h); (4) large variations in scene background; and (5) limited computing power and memory of onboard embedded devices of UAV platforms. In this paper, to address these challenges, a new shallow deep learning-based method for multi-UAV detection and tracking is proposed to achieve high detection and tracking accuracy while having low computational complexity.

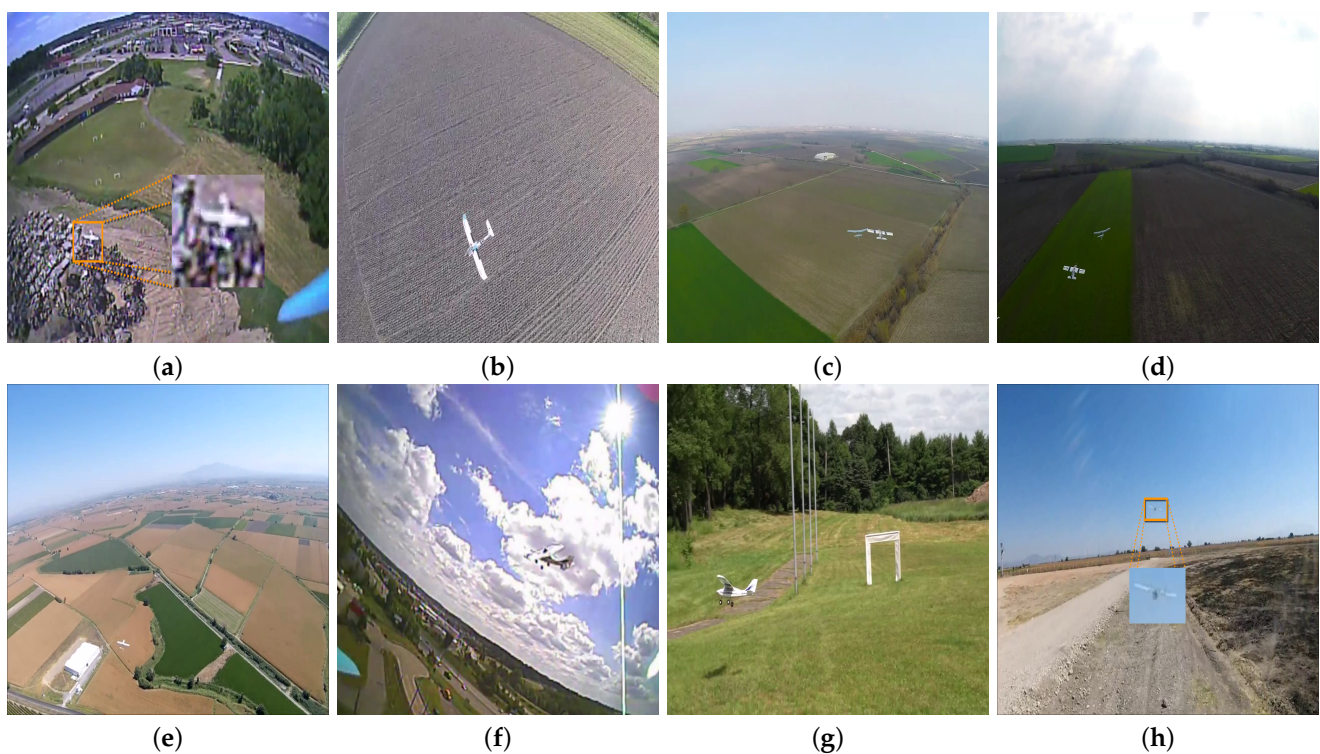


Figure 1. Detecting target or leader UAV in video streams, captured by tracker or follower UAV, deals with several issues including: (a–f) large variations in UAV's visual appearance, viewpoint, dimension of UAVs, illumination, and weather conditions; (g) existence of shadows in scene; as well as (h) low contrast between UAV and background.

In the field of computer vision, many object-detection algorithms have been developed in the last few decades [9]. Conventional object-detection methods (e.g., [10,11]) have been used in many applications and they are mainly based on sliding window search or regional proposal and handcrafted features. Generally, these methods provide low detection accuracy in complex scene scenarios as handcrafted features cannot express the characteristics of objects precisely and they are computationally expensive. Consequently, these approaches are not applicable to real-time computer vision applications. To cope with the limitation of the conventional approaches, recently many object-detection algorithms

based on deep Convolutional Neural Networks (CNNs) have been developed and employed [12,13]. Generally, deep learning-based object-detection algorithms can be divided into two classes: two-stage and one-stage object detectors. In the former approach, first, a region proposal network is used to estimate candidate object bounding boxes. Then in the second stage, the network extracts features from each candidate box and performs classification and bounding-box regression. In this manner, several methods such as R-CNN [14], SPP-Net [15], Fast R-CNN [16], Faster R-CNN [17], R-FCN [18], and FPN [19] have been proposed. Even though these methods have demonstrated high accuracy, they usually suffer from high computational cost. The latter object detector approach uses a single deep neural network with regression strategy to directly classify and detect objects. Please note that in this approach, the process of region proposal is avoided. Some of the one-step detectors are OverFeat [20], AttentioNet [21], G-CNN [22], Single Shot Detector (SSD) [23], You Only Look Once (YOLO) [24], YOLOv2 [25], YOLOv3 [26], DSSD [27], DSOD [28], RetinaNet [29], and RefineDet [30]. These methods provide both time efficiency and reasonable accuracy which are the ultimate goals of the real-time object-detection task. Among others, YOLO series algorithms have shown a good track record in solving UAV pursuit problem and achieved a higher mean average precision (mAP) than other real-time deep learning-based systems. For instance, A. Bonnet et al. [31], S. Arola et al. [32], and M. A. Akhloufi et al. [33] propose a deep learning framework to detect and track a follower or leader UAV using an optical camera mounted on another UAV. To achieve this, a search area proposal based on particle filters [34] is combined with YOLOv2 CNN model [25] to improve the performance of YOLO algorithm for tracking a UAV located far from the tracker or follower UAV. In these works, the estimated bounding boxes using YOLOv2 is given to particle filters algorithm to predict the target positions in next frames. H. Saribas et al. [35] proposes a real-time hybrid detection and tracking method to pursue an UAV in video frames. In this manner, Kernelized Correlation Filter (KCF) [36] is integrated with YOLOv3 [26] and YOLOv3-tiny detection models. More, specifically, this framework uses shallow version of YOLOv3 (YOLOv3-tiny) to detect a UAV in the first frame or in frames where the tracker fails, and it employs KCF to maintain the tracking in intermediate frames. Although the state-of-the-art methods show the applicability of YOLO series CNN models in real-time UAV-UAV detection and tracking, the methods are tested on simple scenarios as (1) YOLO and YOLOv2 CNN backbones are designed for large objects and (2) YOLOv3-tiny CNN network is a relatively shallow and small network which cannot extract UAV image features with high precision. Therefore, to meet the needs of high UAV detection accuracy, it is vital to develop a new framework to make a trade-off between speed and accuracy for multi-UAV pursuit application. The differences between the proposed method and the existing UAV detection and tracking frameworks are shown in Table 1.

Table 1. A comparative overview of UAV-UAV detection and tracking methods.

Method	Detection Strategy	Backbone	Width of CNN	Layers	Tracking Strategy	Problem Type
Bonnet et al. [31–33]	YOLOv2	Darknet-19	1	32	particle filters	single-UAV pursuit
Saribas et al. [35]	YOLOv3-tiny	Darknet-19	1	24	KCF	single-UAV pursuit
Proposed Method	FastUAV-NET	Inception module	4	28	sKCF	multi-UAV pursuit

This paper presents a novel real-time deep learning-based framework on embedded computer systems for multi-UAV detection and tracking in airborne videos captured in complex and dynamic outdoor environments. The proposed UAV detection framework is inspired by YOLOv3-tiny algorithm and modifies its CNN model to enhance its feature extraction capability while preserves its fast detection speed. Generally, in the context of UAV pursuit, YOLOv3-tiny [26] detection method provides low detection accuracy, and the main reason is three-fold. The first reason is because it uses shallow and simple Darknet19 network architecture as backbone which causes insufficient feature extraction, especially at

the deep convolutional layers where the small UAVs may not have enough information to learn from the network. The second reason is that it makes deep CNN model by simply stacking convolutional and max pooling layers, which simply can result in overfitting and degradation of neural network. The third reason is that YOLOv3-tiny's internal CNN layers use fixed filter sizes (3×3) to detect UAVs. However, UAVs and the objects in their surrounding environments appear with large variations in sizes and aspect ratios so that it is important to design a new network architecture with different filter sizes to extract features with high precision. To solve all these issues, we develop a new CNN architecture, called FastUAV-NET, with the following characteristics:

1. It is an embedded-based CNN network architecture as it needs low compute and memory demand.
2. It is a sparsely connected CNN architecture, inspired by Inception model [37], to maintain the computational cost, while increasing the depth and width of the CNN model.
3. It is a tiny wide network architecture which uses five Inception blocks, where each block is a combination of 1×1 , 3×3 , 5×5 convolutional layers and a max pooling layer, to extract features from scene objects with various sizes and aspect ratios as well as multiple orientations.

Furthermore, the proposed CNN architecture adopts a feature pyramid network to detect UAVs in two different scales. Even using the proposed tiny wide CNN network architecture, the detection frame rate is still low on computer embedded systems mounted onto UAVs. Therefore, the proposed FastUAV-NET is only used to initialize bounding boxes for UAVs in the scene and then sKCF tracking algorithm [38] which can run at a high frame rate is used to track the bounding boxes. The proposed UAV detection algorithm is applied to every 6th frame and then the detected UAVs are tracked in intermediate frames via sKCF [38]. The experiments are performed on a variety of test videos and results indicate that the proposed method is robust to issues caused by fast moving UAVs, changes in scale and aspect ratio of UAVs, illumination variation, camera viewpoint change, specular light, and shadow. Moreover, results show that the proposed framework has the least error detection rate compared to the state-of-the-art methods.

Contributions:

The main contributions of the paper are as follows:

1. proposing a new framework composed of a detector based on a novel deep learning architecture, named FastUAV-NET, and a tracker based on scalable kernel correlation filter;
2. fast and accurate localization of multiple UAVs in airborne video frames;
3. developing a framework that could run online on a GPU embedded platform mounted onto an UAV; and
4. generating the largest UAV-UAV dataset which consists of 25,000 video frames with large variations in both UAVs' backgrounds and foregrounds.

This paper is organized as follows. Section 2 briefly describes the YOLOv3-tiny algorithm. Section 3 describes the proposed framework for multi-UAVs pursuit. Section 4 provides experimental results of the proposed method and compared with state-of-the-art methods. The paper is concluded in Section 5.

2. Brief Review of YOLOv3-Tiny

Currently, one-step object-detection algorithms (e.g., [20–30]) have been widely used in many object-detection problems. Among others, YOLOv3 [26] is one of the most popular object detectors since a single convolutional neural network simultaneously predicts multiple bounding boxes and class probabilities at high accuracy and a fast inference speed. To extract features and detect objects, YOLOv3 [26] stacks two fully convolutional underlying architectures with 1×1 and 3×3 convolution kernels. More specifically, Darknet53 [26]—it is inspired by Resnet [39]—with 53 convolutional layers is used as a backbone to extract

features and then 53 more layers are stacked onto it to achieve object detection. However, using 106 convolutional layers causes heavy computational cost and large run-time memory footprint so that the model is not able to achieve real-time speed in embedded systems. In other words, Darknet53 network deals with a problem of diminishing feature reuse, which makes this network slow. To take advantage of this widely used object-detection method in real-time applications, a lightweight version, called YOLOv3-tiny, has been developed. The YOLOv3-tiny uses Darknet19 network architecture which includes 13-layer for feature extraction and 11 more layers for object detection. The structure of YOLOv3-tiny is shown in Table 2. Generally, this method achieves up to 220 Frame Per Second (FPS), whereas YOLOv3 achieves up to 35 FPS on a computer with a Titan X GPU. In YOLOv3-tiny, to achieve this high computational performance, first, the Darknet53 architecture is simplified by reducing the size of the backbone model and then a two-scale prediction strategy is used to detect objects on two-scale feature maps.

Table 2. YOLOv3-tiny network structure which uses Darknet19 as the backbone network and 2-scale prediction.

Layer (L)	Type	Filter (F)	Kernel	Stride	Input	Output
0	Conv	16	3×3	1	$416 \times 416 \times 3$	$416 \times 416 \times F_{L_0}$
1	Maxpool		2×2	2	$416 \times 416 \times F_{L_0}$	$208 \times 208 \times F_{L_0}$
2	Conv	32	3×3	1	$208 \times 208 \times F_{L_0}$	$208 \times 208 \times F_{L_2}$
3	Maxpool		2×2	2	$208 \times 208 \times F_{L_2}$	$104 \times 104 \times F_{L_2}$
4	Conv	64	3×3	1	$104 \times 104 \times F_{L_2}$	$104 \times 104 \times F_{L_4}$
5	Maxpool		2×2	2	$104 \times 104 \times F_{L_4}$	$52 \times 52 \times F_{L_4}$
6	Conv	128	3×3	1	$52 \times 52 \times F_{L_4}$	$52 \times 52 \times F_{L_6}$
7	Maxpool		2×2	2	$52 \times 52 \times F_{L_6}$	$26 \times 26 \times F_{L_6}$
8	Conv	256	3×3	1	$26 \times 26 \times F_{L_6}$	$26 \times 26 \times F_{L_8}$
9	Maxpool		2×2	2	$26 \times 26 \times F_{L_8}$	$13 \times 13 \times F_{L_8}$
10	Conv	512	3×3	1	$13 \times 13 \times F_{L_8}$	$13 \times 13 \times F_{L_{10}}$
11	Maxpool		2×2	1	$13 \times 13 \times F_{L_{10}}$	$13 \times 13 \times F_{L_{10}}$
12	Conv	1024	3×3	1	$13 \times 13 \times F_{L_{10}}$	$13 \times 13 \times F_{L_{12}}$
13	Conv	256	3×3	1	$13 \times 13 \times F_{L_{12}}$	$13 \times 13 \times F_{L_{13}}$
14	Conv	512	3×3	1	$13 \times 13 \times F_{L_{13}}$	$13 \times 13 \times F_{L_{14}}$
15	Conv	$3 \times (4 + 1 + \text{classes})$	1×1	1	$13 \times 13 \times F_{L_{14}}$	$13 \times 13 \times F_{L_{15}}$
16	Detection					
17	Route 13					
18	Conv	128	1×1	1	$13 \times 13 \times F_{L_{13}}$	$13 \times 13 \times F_{L_{18}}$
19	Up-sampling		2	1	$13 \times 13 \times F_{L_{18}}$	$26 \times 26 \times F_{L_{18}}$
20	Route 19 & 8					
21	Conv	256	3×3	1	$26 \times 26 \times (F_{L_{18}} + F_{L_8})$	$26 \times 26 \times F_{L_{21}}$
22	Conv	$3 \times (4 + 1 + \text{classes})$	1×1	1	$26 \times 26 \times F_{L_{21}}$	$26 \times 26 \times F_{L_{22}}$
23	Detection					

2.1. Detection Procedure in YOLOv3-Tiny

YOLOv3-tiny is an end-to-end object-detection method. In this method, the input image splits into $S \times S$ grid cells (i.e., 13×13 and 26×26) which are used in a CNN architecture to predict 3 bounding boxes and c class probability for each grid cell. In this manner, the network predicts bounding-box center coordinates (x, y) relative to the bounds of the grid cell as well as width and height (w, h) relative to the whole image. Moreover, a confidence score (C) for each bounding box is estimated based on product of the probability that the bounding box contains the object of interest ($p(\text{object})$) and Intersection over Union (IoU) of predicted box and ground-truth box. This can be formulated as follows:

$$C = p(\text{object}) \times IoU_{pred}^{GT} \quad (1)$$

Please note that IoU represents a value between 0 and 1. To calculate IoU , first the overlapping area between the predicted bounding box and ground-truth must be calculated to manipulate intersection and then the intersection must be divided with union which

is the total area between both predicted and ground-truth. Ideally, when the *IoU* value closes to 1, it means the predicted bounding box is close to the ground-truth. Moreover, in YOLOv3-tiny parallel with the bounding-box prediction, each grid cell also predicts *c* conditional class probability.

$$p(c|object) \times p(object) \times IoU_{pred}^{GT} = p(c_i) \times IoU_{pred}^{GT} \quad (2)$$

where the class-specific confidence score, which reflects how possible the object belonging to the class exists in individual box confidence, is estimated via the product of the individual box confidence and conditional class probability.

In contrast to the other YOLO versions, YOLOv3 and YOLOv3-tiny predict the objectness score ($p(object)$) for each bounding box using a multilabel classification approach based on logistic regression rather than SoftMax to better model the data. The model in this framework returns a tensor of shape $S \times S \times \text{number of anchors} \times (\text{bounding-box offsets} + \text{objectness prediction} + c)$, where number of anchors, bounding-box offsets, and objectness prediction are set to 3, 4, and 1, respectively. In the final step of YOLOv3-tiny, the fine tune bounding box(es) is generated using class-specific confidence score thresholding and non-maximum suppression. Finally, to train YOLOv3-tiny, sum of the squared error loss is used for bounding box and binary cross-entropy loss is used for the objectness score and class probabilities. More specifically, YOLOv3-tiny loss function is broken into four main parts: (1) error in bounding-box centers; (2) error in bounding-box dimensions; (3) loss related to confidence score; and (4) object classification loss. In this manner, YOLOv3-tiny employs the following loss function (L):

$$\begin{aligned} L = & \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^B 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^B 1_{ij}^{obj} (2 - w_i \times h_i) [(w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2] \\ & - \sum_{i=0}^{s^2} \sum_{j=0}^B 1_{ij}^{obj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] \\ & - \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^B 1_{ij}^{noobj} [\hat{C}_i \log(C_i) + (1 - \hat{C}_i) \log(1 - C_i)] \\ & - \sum_{i=0}^{s^2} 1_i^{obj} \sum_{c \in \text{classes}} [\hat{p}_i(c) \log(p_i(c)) + (1 - \hat{p}_i(c)) \log(1 - p_i(c))] \end{aligned} \quad (3)$$

where λ_{coord} is a weight used to increase emphasis on boxes with objects whereas λ_{noobj} is used to lower the emphasis on boxes with no objects. In Equation (3), s^2 is the number of cells, B is the number of bounding boxes predicted by each grid, c is the number of classes, C refers to the confidence score, and $p(c)$ refers to the class prediction. Moreover, 1_i^{obj} describes whether the object is in grid i , and 1_{ij}^{obj} denotes that the j^{th} bounding-box predictor in grid i is responsible for that prediction.

2.2. YOLOv3-Tiny Network Architecture

The Darknet19 structure of the YOLOv3-tiny network consists of 24 layers where layer 0 to 12 are convolutional and pooling layers for extracting features of the target objects from the input images. In the backbone architecture, each convolutional layer with filter size 3 and stride 1 is followed by a batch normalization, a Leaky ReLU activation function, and a pooling layer with kernel size 2. In this algorithm, max pooling strategy is used to achieve dimensionality reduction. After feature extraction, YOLOv3-tiny tends to detect objects at two different scales using feature pyramid network. To achieve this, the first detection is made by the 16th layer. For the first 15 layers, the input image is down sampled to generate

feature map of size 13×13 . Then, the detection is made using the 1×1 detection kernel, resulting a feature map of $13 \times 13 \times 3 \times (5 + c)$. Then, the feature map from layer 13 is subjected to a 1×1 convolutional layer before up-sampling by factor of 2 to dimensions of 26×26 . To improve the performance of feature pyramid, the up-sampled feature map is concatenated with the feature map from layer 8, which is the corresponding resolution feature map generated by the Darknet19. Then, the combined feature maps are passed through a 3×3 convolutional layer and followed by 1×1 convolutional layer to fuse the features from the earlier layer. Next, the second detection is made by the 23rd layer, yielding a detection feature map of $26 \times 26 \times 3 \times (5 + c)$. Then to detect objects, the position of the objects is estimated using the semantic information, and a bounding box is dedicated to all possible targets based on 6 anchors at two scales. Finally, the non-maximum value suppression is used to remove redundant bounding boxes to determine the optimal final bounding box.

2.3. Limitations of YOLOv3-Tiny in UAV Detection

YOLOv3-tiny can achieve real-time detection on GPU. However, as shown in Table 2, the Darknet-19 has very limited number of convolutional layers (depth) of CNN, resulting in limited feature extraction capability. This simply leads to low detection accuracy when the scene complexity is high. Consequently, in the UAV pursuit application where the complexity in scene and demand for detection precision are high, YOLOv3-tiny detection algorithm cannot be used without improving its CNN Darknet-19 network architecture.

3. Proposed UAV-UAV Detection and Tracking Framework

This paper presents a new real-time framework to detect and track target or leader UAVs in airborne video streams captured by a tracker or follower UAV. The proposed framework consists of two steps. In the first step, the proposed FastUAV-NET architecture is used to detect UAVs. In the second step, the detected UAV(s) is tracked using scalable Kernel Correlation Filter (sKCF) [38].

3.1. FastUAV-NET Architecture

To cope with the aforementioned limitations of YOLOv3-tiny multi-UAV detection, there can be two different solutions: (1) increasing depth of the Darknet-19 network by adding more layers and (2) increasing width of the Darknet-19 network by adding more filter sizes. By increasing depth using different strategies (e.g., Resnet [39], VGG [40], and Darknet-53 [26]) the network can extract features and approximate the target model with higher accuracy as it increases network nonlinearity. However, this strategy also increases the computational cost and complexity of the network, which makes the network be more difficult to optimize and limits YOLOv3-tiny application in embedded systems. In contrast with the first strategy where more convolution layers are naively stacked for higher performance, inevitably increasing computational cost, in the second strategy, the aim is to increase the width of the Darknet-19 network. To achieve this, the Darknet-19 architecture must be redesigned based on Inception module [37]. In the Inception-based networks the goal is to increase both speed and accuracy by integrating different filter sizes and factorization convolutions to simultaneously extract both local and global features in an image. Therefore, in this paper, the Darknet-19 architecture of YOLOv3-tiny is widened based on Inception module [37] to extract features more precisely from airborne videos without increasing network's complexity. More precisely, the proposed CNN architecture aims at increasing the detection accuracy of YOLOv3-tiny without incurring too much additional computational cost.

The proposed network architecture not only uses the high detection performance of dense thin CNN networks, but also keeps the sparsity of the network, which makes it applicable in GPU embedded systems. The proposed FastUAV-NET architecture, which is shown in Figure 2, includes 43 convolutional, 11 pooling, 6 concatenation, 1 up-sampling, and 2 detection layers. Even though the proposed architecture has high number of elements,

it still has low computational cost. This is due to the fact that the proposed CNN architecture uses multiple convolutional layers on the same level, rather than naively stacking them sequentially. This makes the proposed CNN architecture suitable for real-time applications. The proposed method is divided into two parts: (1) backbone where different inception blocks are used to extract features and (2) head subnet where YOLOv3-tiny detection network is used to detect UAVs.

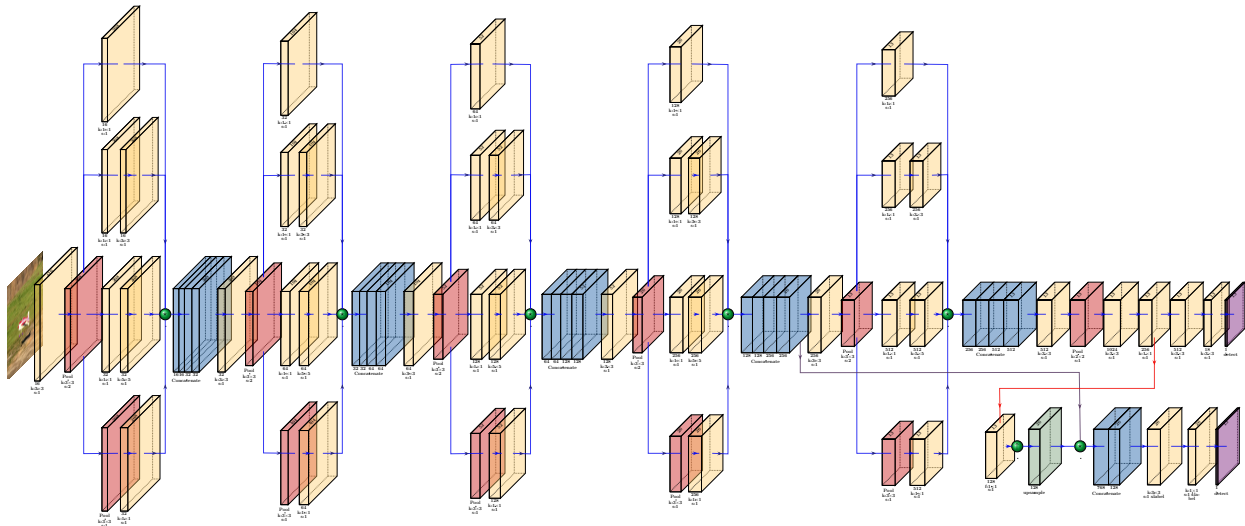


Figure 2. Proposed FastUAV-NET detection architecture. In layers, f and s denote filter size and stride, respectively.

Backbone:

In the proposed backbone architecture, multiple filters with different filter kernel sizes are applied on an input feature map to generate a wider network rather than deeper. In this manner, the backbone of the proposed FastUAV-NET contains five Inception blocks. Each Inception building block consists of 3 units: input unit, feature map generator unit, and output unit. The input unit consists of a 3×3 convolutional layer and a max pooling layer with kernel size of 3×3 and stride of 2, which reduces the size of the feature map by a factor of 2. In the feature map generator unit, 3 convolutional layers, consisting of 5×5 , 3×3 and 1×1 , and a max pooling layer with kernel size of 3×3 and stride of 1 are used. However, one main issue with the feature map generator unit is that the 3×3 and 5×5 convolutions are computationally expensive on the top of the convolutional layer with many filters, which can limit the applicability of the proposed network on the GPU embedded systems. Thus, to make the proposed network lighter, 1×1 convolution is used to reduce dimensionality of the input feature map before applying the 3×3 and 5×5 convolutions as shown in Figure 2. Furthermore, 1×1 convolution is also used after the max pooling layer. Finally, in the output unit, the generated feature maps are concatenated. Batch Normalization is used after each convolutional layer followed by the Leaky ReLU activation function. Based on the aforementioned properties of the backbone architecture, the proposed backbone structure consists of 23 convolutional layers, 6 pooling layers, and 5 concatenation layers.

Head subnet:

The head subnet of the FastUAV-NET architecture employs a multi-scale feature pyramid network to strengthen the features of the backbone network and construct a more efficient multi-scale feature pyramid. More specifically, the FastUAV-NET builds feature pyramid network on the top of the backbone architecture and constructs a pyramid with down-sampling strides 32 and 16. Thus, two levels of the pyramid, which are based on 13×13 and 26×26 resolution feature maps, are used to detect UAVs with various sizes. In this manner, the FastUAV-NET performs large-to-medium scale UAVs detection in 13×13 resolution feature maps and detects the small-scale UAVs in 26×26 .

Moreover, to construct 26×26 resolution feature map, the proposed FastUAV-NET uses concatenation to perform the merging step in lateral connections. In other words, the concatenation layer is used to take a feature map from earlier in the network and merge it with up-sampled features to get more precise semantic information. In the proposed architecture, instead of using a fully connected layer, a 1×1 convolutional layer with the tensor shapes of $13 \times 13 \times (3 \times (4 + 1 + 1))$ for the lower resolution feature map and $26 \times 26 \times (3 \times (4 + 1 + 1))$ for the higher resolution feature map are used. As a result, at each scale, the number of filters in the last convolutional layer is set to 18. In the final step of FastUAV-NET, the fine-tuned bounding box(es) is generated using thresholding UAV confidence score and non-maximum suppression.

In each detection layer of the proposed FastUAV-NET, 3 anchor boxes must be provided to achieve high detection rate. The anchor boxes are one of the vital and critical parameters and they should be provided to the proposed network based on the scale and size of UAVs in the training data. In the proposed method, to automatically find the 6 optimum anchor boxes, the k-means++ algorithm, where $k = 6$, is applied to the training annotations. The estimated 6 anchor parameters are sorted based on area and then distributed evenly across scales. Please note that only the prior box which has the highest intersection over union with the ground-truth label will be considered for detecting a UAV.

3.2. Tracking Strategy

The proposed FastUAV-NET provides a great accuracy in finding location of UAV(s) in airborne video frames; however, it cannot be considered to be a tracking strategy as it has still high computational time on onboard embedded systems. To solve this issue, a swift and efficient tracking algorithm must be integrated to the FastUAV-NET detection algorithm. Currently, various tracking methods have been proposed and developed [36,38,41–45]. Among others, scalable Kernel Correlation Filter (sKCF) tracker [38] provides an efficient and fast-tracking algorithm to track objects [41]. Therefore, in this paper, sKCF tracker [38] algorithm is combined with the proposed FastUAV-NET to track UAV(s) within a certain frames.

Recently, the KCF tracking algorithm [36] and its variants [41] have been widely used in real-time visual tracking as they have high accuracy and low computational time. The KCF tracker is based on the idea of conventional correlational filter and uses a kernel strategy and circulant matrices to significantly improve the computation speed. However, the KCF uses a fixed size kernel so that the tracker is not able to handle scale changes occurring during motion, which is a very common issue in the corresponding problem. To solve the fixed size limitation in the KCF, the improved KCF tracking algorithm, which is sKCF [38], is used. In this algorithm, a keypoint-based model and adjustable Gaussian kernel function are employed for scale estimation. Furthermore, this tracking algorithm improves the computational time as it integrates the HOG descriptors and complex conjugate symmetric packed format. Therefore, in this proposed framework, the sKCF algorithm is combined with FastUAV-NET to track UAVs.

The sKCF tracker needs initialization in the first frame and in case tracking fails due to motion of camera, illumination change, occlusion, and motion change. Hence, it is important to develop a strategy to take advantage of both FastUAV-NET and sKCF to solve their limitations. In this paper, the FastUAV-NET is applied to detect UAV(s) in every 6th frame and then the detected UAV(s) is used as an initial solution for sKCF tracker [38] to track UAVs in intermediate frames. In this manner, the proposed framework has the following advantages: (1) low memory footprint, (2) applicable to GPU embedded systems, and (3) ability to lock onto each UAV target/track during the pursuit.

4. Results and Discussion

In this section, the performance of the proposed FastUAV-NET multi-UAV detection architecture with and without sKCF tracking [38] is quantitatively and qualitatively evalu-

ated on outdoor videos captured under different range of variability in real scenes. The proposed method is compared with state-of-the-art detection methods.

4.1. Dataset

The dataset is generated from 11 different videos which are used for training and testing. The video frames contain one or two UAVs and are captured in RGB color space under: (1) large background combinations, (2) various weather conditions, (3) different illumination changes, (4) different viewpoints, scales, and aspect ratios (see Figure 3), and (5) existence of shadows. In this dataset, size of UAVs varies from 11×8 to 300×150 pixels. The height and width of the video frames are 416×416 . To generate training dataset, 6 different videos are used and the remaining 5 videos are used to construct testing dataset. In this manner, the training and testing datasets consist of 15,000 and 10,000 video frames, respectively. Moreover, they are manually annotated by bounding boxes. The basic characteristics of the constructed dataset are summarized in Table 3.

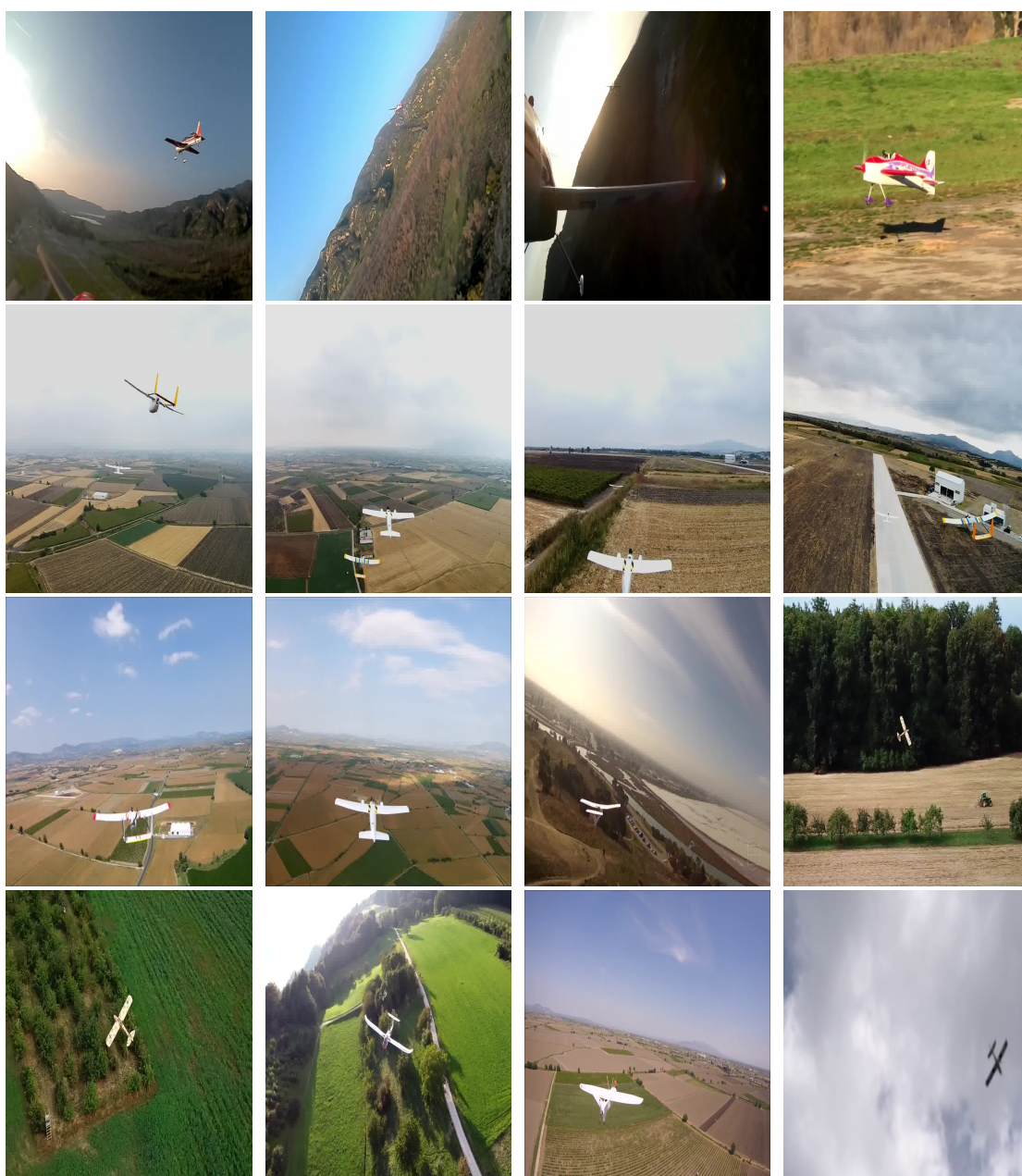


Figure 3. Example UAV images from different videos captured by airborne optic camera. These images show the characteristics and complexity of training and testing dataset.

Table 3. Attributes of UAV-UAV dataset.

	Videos	Video Frames	Number of UAVs	Resolution	Color Space	Annotation
Training	6	15,000	17,843	416 × 416	RGB	Bounding Box
Test	5	10,000	11,128	416 × 416	RGB	Bounding Box

4.2. Quantitative Measurement Metrics

To validate the detection results, three different quantitative error measures such as recall, precision, and Average Precision (AP) are used. Recall refers to the ratio of true positives (T_P) to the number of true positives plus the number of false negatives (F_N). Recall score is written as $Recall = \frac{T_P}{T_P + F_N}$. The precision refers to ratio of true positives (T_P) to the number of true positives plus the number of false positives (F_P). This metric can be formulated as $Precision = \frac{T_P}{T_P + F_P}$. To determine whether a predicted bounding box by a model is true positive, false positive, or false negative, the overlap between UAV detection result (R_P) and ground-truth bounding box (R_{GT}) is considered.

$$IoU = \frac{|R_{GT} \cap R_P|}{|R_{GT} \cup R_P|} \quad (4)$$

When the IoU score is greater than 0.5, it is considered a true positive, else it is considered a false positive. Moreover, when the model could not detect an UAV is considered a false negative. The third metric is average precision which is the area under precision-recall curve.

4.3. UAV Detection Methods

To understand and analyze the performance of the detection methods, the proposed model is compared with YOLOv3 [26] and YOLOv3-tiny models. Moreover, to improve the computational cost, all the UAV detection methods are applied to every 6th frame and then the detected UAVs are tracked in intermediate frames via the sKCF algorithm [38]. In this manner, the tracking performance of UAVs is also examined using different UAV detection models.

In the proposed and the compared UAV detection methods, anchor boxes are used to predict bounding boxes. Please note that YOLOv3 algorithm uses nine anchors whereas the proposed and YOLOv3-tiny methods employ six anchors. In YOLOv3, nine anchors are selected using k-means++ clustering and they are set to (11, 15), (25, 18), (47, 30), (39, 53), (88, 54), (71, 106), (138, 85), (221, 130), and (289, 188). In the proposed method and YOLOv3-tiny methods, the anchors are set to (11, 15), (39, 53), (71, 106), (138, 85), (221, 130), and (289, 188). In all the methods, the parameters such as learning rate, batch size, and subdivision are selected as 0.0001, 64, and 16, respectively. Moreover, threshold score for all the methods is set to 0.5. In the proposed and YOLOv3-tiny methods, epoch is set to 20,000 whereas in YOLOv3, epoch is 40,000. The convolutional neural network frameworks are implemented in CUDA C++ on the Windows platform. They train and test on a computer with a single NVIDIA GTX 1050TI GPU, an Intel i7-7700HQ CPU, and 16 GB RAM. Moreover, the algorithms also are tested on a Jetson TX2 module embedded on a UAV.

4.4. Performance Comparison of UAV Detection Methods

In this section, the performance of the proposed FastUAV-NET multi-UAV detection model is compared with the state-of-art detection methods which are YOLOv3 and YOLOv3-tiny. The proposed FastUAV-NET and compared detection methods are trained with the 15,000 collected training dataset and are tested on 10,000 video frames from testing dataset. To evaluate the accuracy of obtained bounding boxes using proposed FastUAV-NET detection method as well as YOLOv3 and YOLOv3-tiny, different quantitative measurements such as precision, recall, IoU, and AP quantitative measurements are

used. According to the recall and precision rates of the test results obtained using different methods, the precision-recall curves of UAV detection are plotted and shown in Figure 4. Figure 4 shows that FastUAV-NET performs favorably against YOLOv3 and YOLOv3-tiny. Moreover, the IoU and AP of these 3 models on the generated test dataset are shown in rows 2 to 4 of Table 4. The proposed FastUAV-NET provides 0.7576 AP and 0.6175 average IoU, YOLOv3 has 0.4579 AP and 0.5225 average IoU, and YOLOv3-tiny gives 0.4210 AP and 0.4683 average IoU. The results in Figure 4 and Table 4 show that the proposed multi-UAV detection method provides the highest AP and IoU scores. The IOU results simply show that the proposed FastUAV-NET achieves a higher overlap between the predicted bounding box and the ground-truth. On the other hand, the YOLOv3-tiny detection method provides the lowest accuracy. The main reasons that the proposed method is more efficient than the other two methods are two-fold. First, the proposed model is based on different kernel sizes for convolution operation which causes multi-level feature extraction. Please note that extracting multiple features using multiple filters with different kernel sizes simply improves the performance of the network as the network can learn UAVs from small details, middle sized features or almost whole images. Second, the proposed network is specifically designed for UAV detection in wild airborne scenes; however, this is not the case in YOLOv3 and YOLOv3-tiny where their network architectures are designed to detect various objects in natural scenes. Consequently, the FastUAV-NET is more robust to large variations in both UAVs' backgrounds and foregrounds. Moreover, the results show that the widening of CNN network is a much more effective way of improving performance of UAV detection compared to increasing depth CNN network.

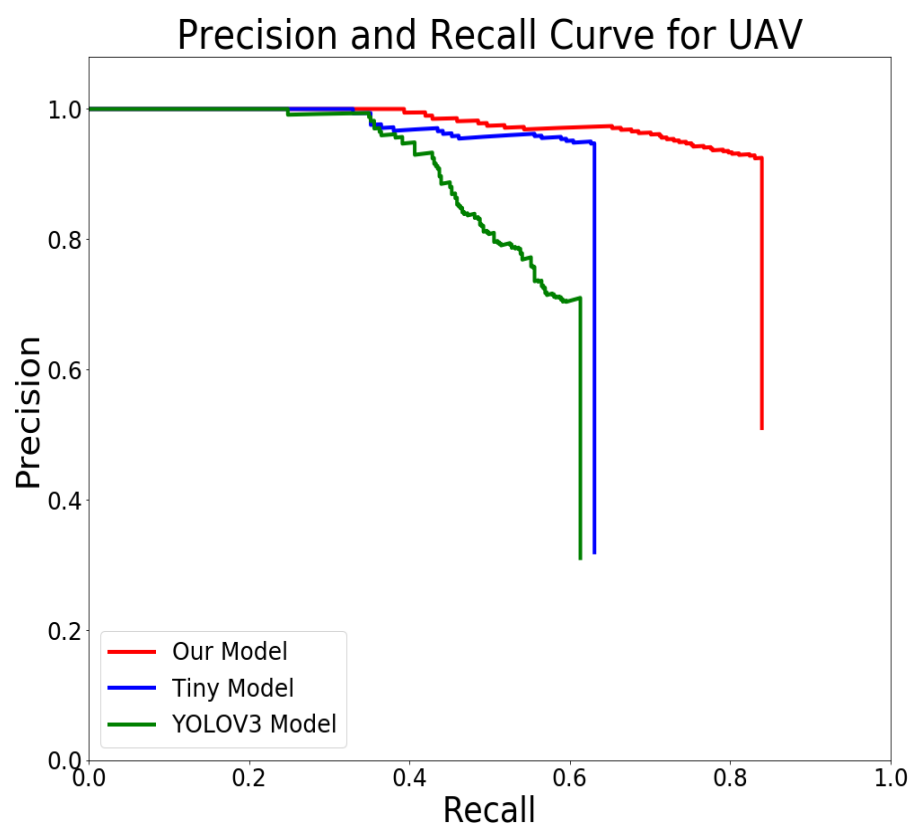


Figure 4. Precision and recall scores of FastUAV-NET and the state-of-the-art detection methods on the UAV testing dataset.

Table 4. Quantitative results for 10,000 testing video frames. The algorithms are tested on both PC and a Jetson TX2 GPU embedded system mounted onto UAVs.

Method	FPS			
	PC	Jetson-Tx2	IOU	AP
Proposed Method	21.35	13	0.6175	0.7576
YOLOv3	13.53	7	0.5225	0.4579
YOLOv3-Tiny	50.36	25	0.4683	0.4210
Proposed Method + sKCF	52.83	29	0.6583	0.7916
YOLOv3 + sKCF	34.04	19	0.5470	0.4788
YOLOv3-Tiny + sKCF	75.34	36	0.5046	0.4568

The task of UAV pursuit is considered to be a very challenging problem due to the existence of shadows, image instability, platform motion, instant change in size of the captured UAVs, existence of many interfering objects in the environment, and finally variation of camera orientation and illumination. To visualize this, 3 detection methods are applied to 12 different frames of 5 test videos under different scenarios and the results are depicted in Figure 5. The results shows that the proposed method is robust to existence of shadows and sudden changes in illumination, weather condition, background, scale, aspect ratio, and viewpoints. Figure 5 illustrates that the proposed detection method effectively detects all UAVs. Furthermore, it can be seen that the proposed detection method can drastically improve accuracy of the predicted bounding boxes in terms of scale and aspect ratio. However, in multi-UAV scenario (see Video 2), the results show that the proposed method can fail to detect multiple UAVs when UAVs lie very close to each other. Moreover, Figure 5 also demonstrates that the YOLOv3 and YOLOv3-tiny networks have high miss detection rates under most of the scenarios. Consequently, the overall experiment results show that the widened network architecture can even outperforms thin 106-layer deep YOLOv3 network.

The experimental results that are obtained on the computational time are tabulated in Table 4. The computational time on the desktop computer shows that all methods can be used for real-time applications. On the other hand, the computational results on the embedded Nvidia Jetson TX2 show that only the FastUAV-NET and YOLOv3-tiny can be considered for embedded applications as YOLOv3 can run at 7 FPS, which is very slow. Table 4 illustrates that the proposed model is approximately two times quicker than YOLOv3. After training, the proposed network runs at about 13 FPS on Nvidia Jetson TX2. However, the YOLOv3-tiny network is quicker than the proposed network as it runs at 25 FPS. The main reason that YOLOv3-tiny is quicker than the proposed network architecture is simply because the YOLOv3-tiny has only 13 convolutional layers which decreases computational time, but with the cost of decreasing accuracy and quality of solutions. Even though the proposed CNN architecture is slower than YOLOv3-tiny, it is significantly provides higher detection accuracy.

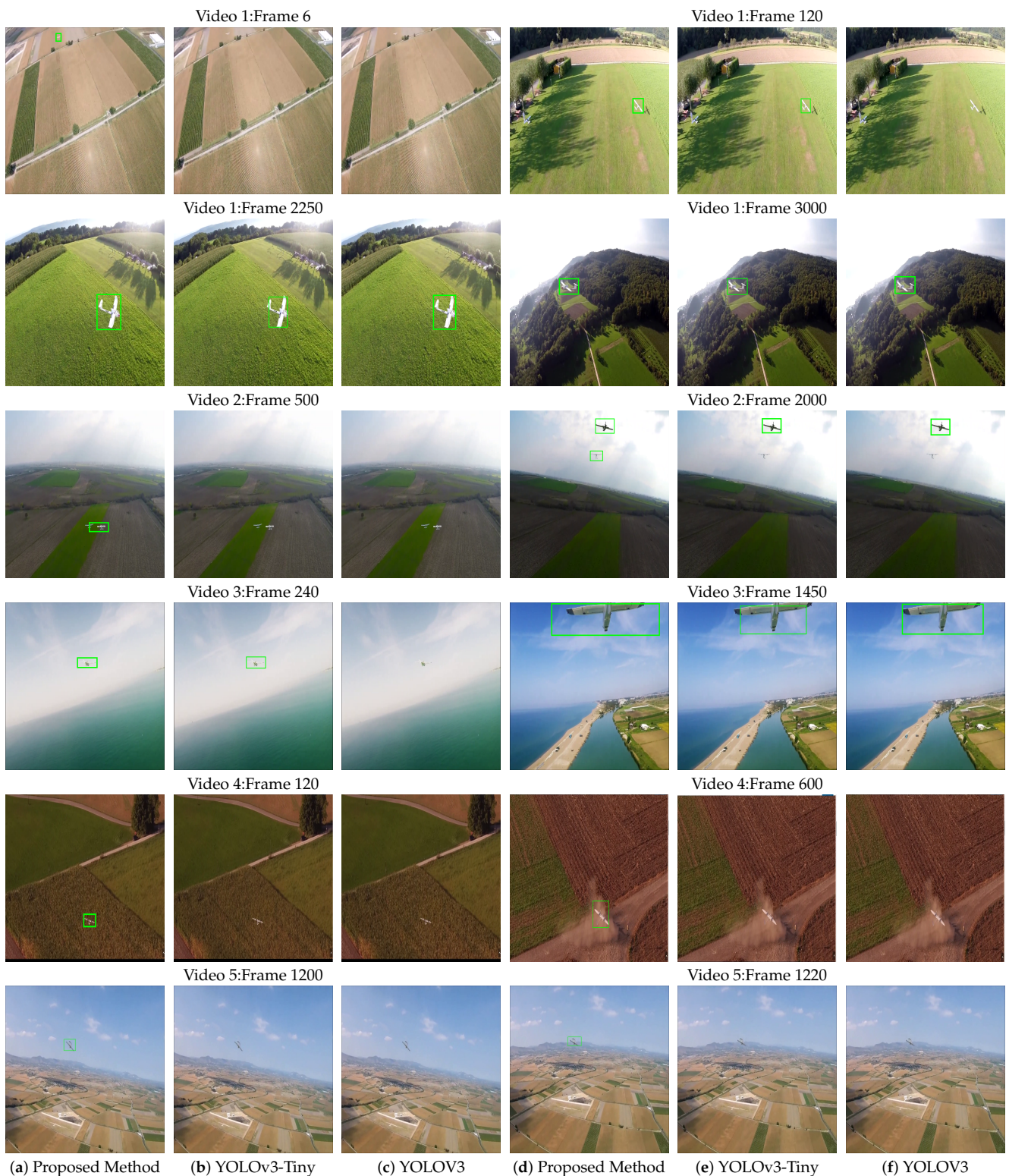


Figure 5. Illustration of UAV detection results in different video frames using proposed FastUAV-NET method, YOLOv3-tiny and YOLOv3.

4.5. Performance Comparison of UAV Detection Methods with Tracking

Here, the intention is to test the performance of detection algorithms with the sKCF tracking method [38] on the 5 test videos. To this end, the detection algorithms are applied to every 6th frame and then the predicted bounding box(es) is tracked in intermediate frames using the sKCF algorithm. To conduct experiments, the estimated bounding boxes

using sKCF algorithm are used to calculate IoU and AP scores and the results are tabulated in the last 3 rows of Table 4. The results show that the highest and the lowest accuracies belongs to the sKCF tracking method initialized with the FastUAV-NET and YOLOv3-tiny, respectively. Table 4 shows that the proposed method with the sKCF tracker provides 0.7916 AP and 0.6583 average IoU, YOLOv3 with the sKCF tracker gives 0.4788 AP and 0.5470 average IoU, and YOLOv3-tiny with the sKCF tracking method provides 0.4568 AP and 0.5046 average IoU. More specifically, the results indicate that the proposed detection method with the sKCF tracker outperforms approximately by 65% and 73%. Furthermore, Table 4 shows that integration of UAV detection methods with the sKCF tracking algorithm reduces the computational cost, and thus to speed up the multi-UAV detection and tracking methods. From the results it is obvious that both proposed framework and YOLOv3-tiny can only be considered to run in real time on the embedded Nvidia Jetson TX2. Finally, we present a visual comparison of the proposed method along with YOLOv3-tiny on the test videos dataset and the qualitative detection results are shown in Figure 6. The tracking results in video 1 illustrate that initializing the sKCF tracker with the proposed network successfully tracks the UAV in all the inter-frames, but initializing the sKCF with YOLOv3-tiny fails to detect the UAV at inter-frame 2750. Video 2 demonstrates multi-UAV scenario, and the results show that the proposed framework tracks UAVs in most of the frames, except where the UAVs are very close to each other. On the other hand, the sKCF tracker with YOLOv3-tiny provides very low tracking performance. For instance, the proposed method can detect both UAVs at video frames 640 and 850, but the sKCF tracker with YOLOv3-tiny could only track one of the UAVs. Video 4 belongs to complex background scenario and the results depict that our method can track the UAV in all the video frames; however, the compared framework fails to track the UAV at video frames 380 and 410. Consequently, the qualitative results verify that the proposed method can initialize the sKCF tracker with higher accuracy than the YOLOv3-tiny.

4.6. Discussion

The ultimate goals of the multi-UAV detection and tracking are two-fold. First, UAVs must be localized with high precision. Second, the processing speed should also be high so that the localization system could be run online on a GPU embedded system mounted onto an UAV. To achieve these goals, the current state-of-the-art methods tries to integrate the most popular real-time object-detection methods such as YOLOv3-tiny and YOLOv2 with one of the fast-tracking algorithms including KCF and particle filters. The YOLOv3-tiny and YOLOv2 methods use the Darknet-19 backbone structure to extract features. However, in this paper, it is shown that the detection methods based on the Darknet-19 architecture are fast, but cannot perform well with the constructed UAV dataset, which comprises RGB video frames with large variations in both UAVs' backgrounds and foregrounds. This is because the Darknet-19 is a shallow and simple CNN network and cannot extract features and approximate the UAV model with high accuracy. Moreover, the experiments show that when focusing on UAV's pursuit problem, a customized CNN architecture must be used to improve the detection performance. In this manner, we propose to widen the backbone structure of the YOLOv3-tiny detection method to balance the localization performance by learning more robust features and the speed without massively increasing the depth of convolutional layers. The results clarify that the proposed method can detect UAVs with high precision even at harsh environmental conditions. However, the experimental results indicate the proposed method can fail to detect multiple UAVs when UAVs fly very close to each other. To further improve the computational time of the FastUAV-NET detection algorithm, scalable Kernel Correlation Filter (sKCF) tracker which is an efficient and fast-tracking algorithm is used. More specifically, the FastUAV-NET detection algorithm is applied to every 6th frame and then the detected UAV(s) is tracked in intermediate frames using the sKCF tracker. The experimental results show that FastUAV-NET combined with the sKCF tracker outperforms sKCF tracker initialized with YOLOv3-tiny approximately by 73%.

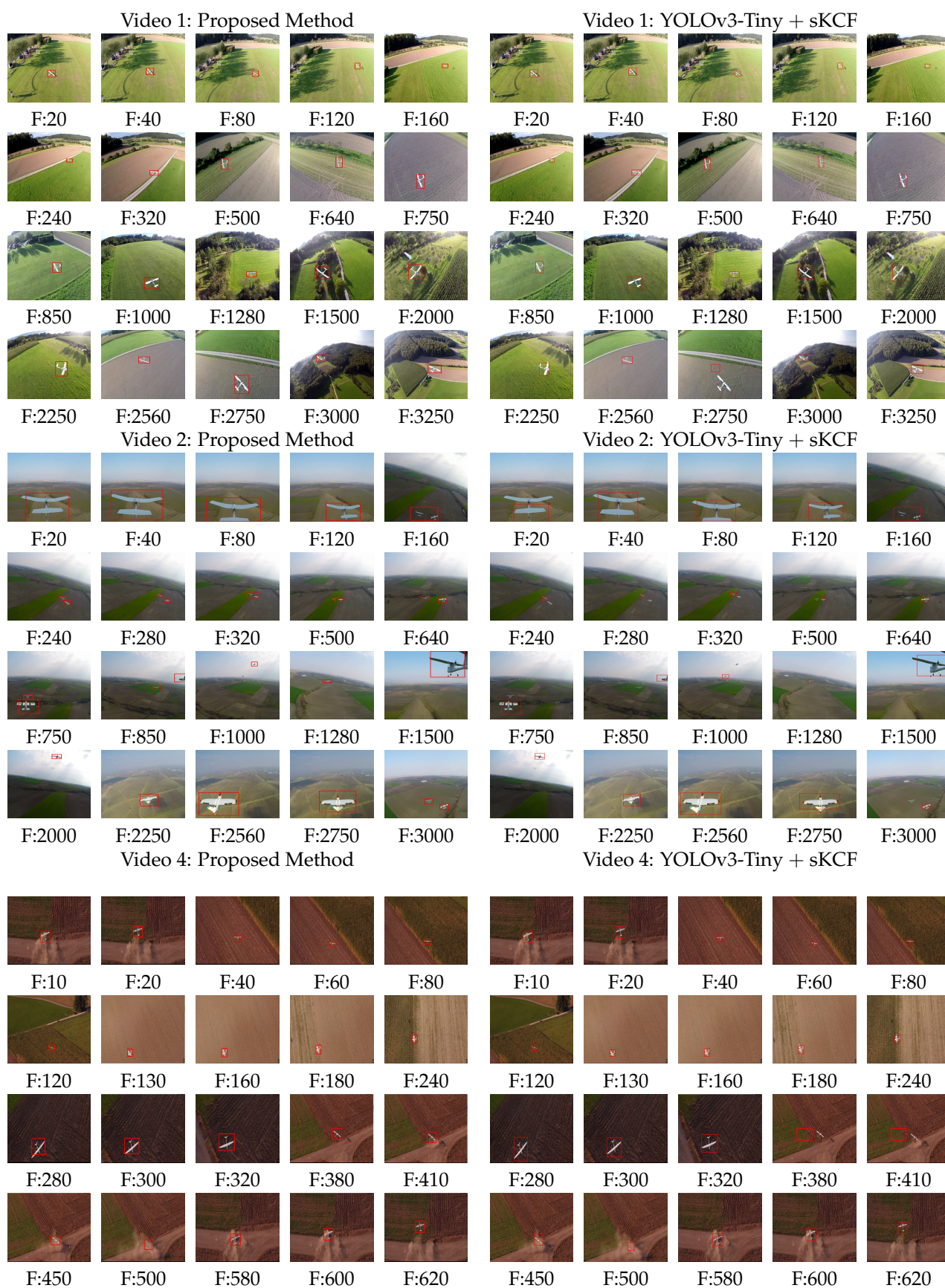


Figure 6. Illustration of UAV tracking results in different video frames using proposed framework and YOLOv3-tiny with the sKCF tracker.

5. Conclusions

In this paper, a new real-time deep neural network framework to detect and track multi-UAV in airborne videos is proposed. To achieve this, a new tiny wide deep Convolutional Neural Network (CNN) based on YOLOv3-tiny detection method is designed and combined with scalable Kernelized Correlation Filter (sKCF) tracking algorithm. The major improvement of the FastUAV-NET is owing to the Inception model which allows the network gets wider and extract feature with higher accuracy without significantly increasing the computational cost. In addition, the sKCF tracking algorithm is used to track the detected UAVs within a certain video frames. This simply improves the performance of the detection and tracking as well as the computational cost. Therefore, the proposed framework is a light, fast and efficient algorithm which can also be used on GPU embedded systems mounted on UAVs. The proposed framework is compared with the state-of-the-art detection methods and they are tested on different videos captured under various complexities. Moreover, experimental results demonstrate that the FastUAV-NET takes full advantage of image features in the framework and improves the performance of multi-UAV detection accuracy with low memory footprint. More specifically, the proposed framework achieves 0.7916 AP with 29 FPS performance on an embedded Jetson TX2 platform. Considering a trade-off between accuracy and speed, the proposed framework exhibits the best performance in many complex scenarios. As a future work, to further develop the proposed framework, we will focus on autonomous UAV target tracking technique that sequentially builds upon its prior knowledge of the environment and the current location of the target UAV. To achieve autonomous target UAV tracking, a reinforcement learning-based approach can be developed to predict the follower UAV motion actions based on detection and tracking results.

Author Contributions: Data acquisition, A.Y.; conceptualization, H.K.; software, A.Y.; experiments, H.C.; supervision, A.Y.; methodology, A.Y.; formal analysis, H.K. and T.C.; resources, A.Y. and H.C.; validation, T.C.; review and editing, A.Y. and T.C.; writing original draft, A.Y. and H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Lu, Y.; Xue, Z.; Xia, G.S.; Zhang, L. A Survey on Vision-Based UAV Navigation. *J. Geo-Spat. Inf. Sci.* **2018**, *21*, 21–32. [\[CrossRef\]](#)
2. Semsch, Z.E.; Jakob, M.; Pavlicek, D.; Pechoucek, M. Autonomous UAV Surveillance in Complex Urban Environments. In Proceedings of the IEEE International Conference on Web Intelligence and Intelligent Agent Technology, Milan, Italy, 15–19 September 2009; pp. 82–85.
3. Ahmad, A.; Tahar, K.N.; Udin, W.S.; Hashim, K.A.; Darwin, N.; Hafis, M.; Room, M.; Hamid, N.F.A.; Azhar, N.A.M.; Azmi, S.M. Digital Aerial Imagery of Unmanned Aerial Vehicle for Various Application. In Proceedings of the IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, 29 November–1 December 2013; pp. 535–540.
4. Jordan, S.; Moore, J.; Hovet, S.; Box, J.; Perry, J.; Kirsche, K.; Lewis, D.; Tse, Z.T.H. State-of-the-Art Technologies for UAV Inspections. *J. IET Radar Sonar Navig.* **2018**, *2*, 151–164. [\[CrossRef\]](#)
5. Troudi, A.; Addouche, S.A.; Dellagi, S.; el Mhamedi, A. Post-Production Analysis Approach for Drone Delivery Fleet. In Proceedings of the IEEE International Conference on Service Operations and Logistics, and Informatics, Bari, Italy, 18–20 September 2017; pp. 150–155.
6. Faical, B.S.; Pessin, G.; Filho, G.P.R.; Carvalho, A.C.P.L.F.; Furquim, G.; Ueyama, J. Fine-Tuning of UAV Control Rules for Spraying Pesticides on Crop Fields. In Proceedings of the International Conference on Tools with Artificial Intelligence, Limassol, Cyprus, 10–12 November 2014; pp. 527–533.
7. Erdelj, M.; Natalizio, E. UAV-Assisted Disaster Management: Applications and Open Issues. In Proceedings of the International Conference on Computing, Networking and Communications, Kauai, HI, USA, 15–18 February 2016; pp. 1–5.
8. Shakhathreh, H.; Sawalmeh, A.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634. [\[CrossRef\]](#)
9. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A Survey of Deep Learning-based Object Detection. In Proceedings of the Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 1–30.

10. Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2005; pp. 1–8.
11. Moranduzzo, T.; Melgani, F. A SIFT-SVM Method for Detecting Cars in UAV Images. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, Seoul, Korea, 22–27 July 2012; pp. 6868–6871.
12. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikainen, M. Deep Learning for Generic Object Detection: A Survey. In Proceedings of the Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 1–35.
13. Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [[CrossRef](#)] [[PubMed](#)]
14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
15. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [[CrossRef](#)] [[PubMed](#)]
16. Girshick, R. Fast R-CNN. In Proceedings of the International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
17. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-cnn: Towards Real-Time Object Detection with Region Proposal Networks. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
18. Li, Y.; He, K.; Sun, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In Proceedings of the Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 379–387.
19. Lin, T.Y.; Dollar, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature Pyramid Networks for Object Detection. In Proceedings of the Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
20. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
21. Yoo, D.; Park, S.; Lee, J.Y.; Paek, A.S.; Kweon, I.S. AttentionNet: Aggregating Weak Directions for Accurate Object Detection. In Proceedings of the International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
22. Najibi, M.; Rastegari, M.; Davis, L.S. G-CNN: An Iterative Grid Based Object Detector. In Proceedings of the International Conference on Computer Vision, Las Vegas, Nevada, 25–28 July 2016.
23. 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, 8–16 October 2016.
24. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
25. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
26. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
27. Fu, C.Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional Single Shot Detector. *arXiv* **2017**, arXiv:1701.06659.
28. Shen, Z.; Liu, Z.; Li, J.; Jiang, Y.G.; Chen, Y.; Xue, X. DSOD: Learning Deeply Supervised Object Detectors from Scratch. In Proceedings of the International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
29. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
30. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-Shot Refinement Neural Network for Object Detection. In Proceedings of the Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
31. Bonnet, A.; Akhloufi, M.A. UAV Pursuit using Reinforcement Learning. In Proceedings of the SPIE Unmanned Systems Technology, Baltimore, MD, USA, 16–18 April 2019.
32. Arola, S.; Akhloufi, M.A. UAV Pursuit-Evasion using Deep Learning and Search Area Proposal. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 1–6.
33. Akhloufi, M.A.; Arola, S.; Bonnet, A. Drone Chasing Drones: Reinforcement Learning and Deep Search Area Proposal. *J. Drones* **2019**, *3*, 58. [[CrossRef](#)]
34. Akhloufi, M.A.; Regent, A.; Ssosse, R. 3D Target Tracking using a Pan and Tilt Stereo Vision System. In Proceedings of the SPIE, Defense, Security, and Sensing, Airborne Intelligence, Surveillance, Reconnaissance Systems and Applications, Baltimore, MD, USA, 1–2 May 2013; pp. 8713–8738.
35. Saribas, H.; Uzun, B.; Benligiray, B.; Eker, O.; Cevikalp, H. A Hybrid Method for Tracking of Objects by UAVs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–20 June 2019.
36. Henriques, J.F.; Caseiro, R.; Martins, P.; Batista, J. High speed tracking with kernelized Correlation Filters. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 583–596. [[CrossRef](#)] [[PubMed](#)]
37. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
38. Montero, A.S.; Lang, J.; Laganieri, A.R. Scalable Kernel Correlation Filter with Sparse Feature Integration. In Proceedings of the IEEE International Conference on Computer Vision Workshop, Santiago, Chile, 7–3 December 2015; pp. 587–594.

39. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
40. Liu, S.; Deng, W. Very Deep Convolutional Neural Network based Image Classification using Small Training Sample Size. In Proceedings of the Asian Conference on Pattern Recognition, Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 730–734.
41. Georgea, M.; Josea, B.R.; Mathew, J. Performance Evaluation of KCF based Trackers using VOT Dataset. In Proceedings of the International Conference on Smart Computing and Communication, Tokyo, Japan, 10–12 December 2018; pp. 560–567.
42. Chen, Z.; Zhong, B.; Li, G.; Zhang, S.; Ji, R. Siamese Box Adaptive Network for Visual Tracking. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 16–18 June 2020.
43. Zhou, Q.; Zhong, B.; Lan, X.; Sun, G.; Zhang, Y.; Zhang, B.; Ji, R. Fine-Grained Spatial Alignment Model for Person Re-Identification with Focal Triplet Loss. *IEEE Trans. Image Process.* **2020**, *29*, 7578–7589. [[CrossRef](#)]
44. Zhong, B.; Bai, B.; Li, J.; Zhang, Y.; Fu, Y. Hierarchical Tracking by Reinforcement Learning based Searching and Coarse-to-fine Verifying. *IEEE Trans. Image Process.* **2019**, *28*, 2331–2341. [[CrossRef](#)] [[PubMed](#)]
45. Zhou, Q.; Zhong, B.; Zhang, Y.; Li, J.; Fu, Y. Deep Alignment Network Based Multi-person Tracking with Occlusion and Motion Reasoning. *IEEE Trans. Multimed.* **2019**, *21*, 1183–1194. [[CrossRef](#)]