International Journal of Innovative Computing, Information and Control
Volume **17**, Number **1**, February **2021**

ICIC International ©2021 ISSN 1349-4198
pp. **205–224**

MULTI-SMALL TARGET DETECTION AND TRACKING BASED ON IMPROVED YOLO AND SIFT FOR DRONES

Pancheng Lu, Yong Ding and Changjian Wang

College of Automation
Nanjing University of Aeronautics and Astronautics
No. 29, Jiangjun Avenue, Jiangning District, Nanjing 210016, P. R. China
dingyong@nuaa.edu.cn

Received April 2020; revised August 2020

Abstract. *Multi-target tracking algorithms applied to drones often face more detection algorithm, while achieving active tracking of small targets and fast-moving targets. In response to these problems, this paper proposes a multi-small target detection and tracking method based on improved YOLO and SIFT for drones. First, we prune the network of the YOLO algorithm. Through sparse training, channel pruning, and network adjustment, the drone can deploy a target detection network with a smaller model. Secondly, we propose a method of adaptive threshold and minimum distance and optimize SIFT feature extraction, achieve effective detection of small targets, and improve tracking accuracy and success rate. Finally, we consider the needs of the UAV to track fast-moving targets, we use scale and position information to complete the data association, and the UAV can effectively track fast-moving targets. The algorithm of this paper is evaluated by the VisDrone 2019 MOT benchmark data set. The leading evaluation indicators MOTA and MOTP are 38.7 and 75.7. The secondary evaluation indicators IDF1, ML, and FN, etc. all perform better. The algorithm meets the complex task requirements of multi-target tracking for UAVs.*

Keywords: Drone, YOLO algorithm, Network pruning, SIFT features, Data association, Multi-target tracking

1. **Introduction.**

Unmanned aerial vehicle (UAV) has played an important role both in local and national defense fields, and it has been applied to intelligent transportation, geological exploration, military guidance, and aviation visual navigation [1], etc. Multi-target tracking, as one of the foremost research directions in the field of drone vision, can significantly enhance the autonomous flight and monitoring capabilities of the drone, enabling the drone to complete more types of tasks and adapt to more complex and changing environments [2].

At present, the multi-target detection and tracking algorithm (Tracking-by-Detection, TBD) [3] has attracted much attention, and it has an excellent performance in the detection of new targets [4, 5, 6, 7, 8, 9]. Li et al. [10] proposed a target tracking method based on segment and multi-feature adaptive fusion. Different types of occlusions were distinguished by importing segments of color, HOG features, and angular features. Then different combinations of occlusions were used to perform pre and post information matching to reconstruct the trajectory information before and after the occlusion of the target improves the reliability of the algorithm. Zhang et al. [11] combined the Kalman filter algorithm and the mean shift algorithm to divide the tracker into different levels to track different detection templates. In this paper, the tracker is divided based on the number

DOI: 10.24507/ijicic.17.01.205

of detection templates owned by the tracker. Layer is performed through a fixed threshold to achieve the goal of online adaptive tracking of the target. Guan et al. [12] used particle filtering to track the target. It compares the position of the particles before and after, linearly estimates the moving direction of the particles, and then filters and matches these predicted positions to obtain the complete motion trajectory of the target. Gan et al. [13] proposed a hierarchical particle filter method based on multi-feature fusion. Using different feature information, it can not only handle long-term occlusion but also track the failure of a single feature in a complex environment. Ju et al. [14] adopted the idea of layering, and the detector was also layered. The whole detector is divided into four layers. The first layer is responsible for generating new templates and processing the trajectory generated in the previous frame. The second layer focuses on the trajectory drift phenomenon. The third layer processes the remaining candidate templates. The last layer deals with the problem of trajectory fragmentation caused by the occlusion of the target. Hierarchical association information achieves the ideal classification processing effect, so it can better track the target.

Although the detection-based multi-target tracking algorithm has an excellent tracking effect in general scenes [15], there are still many challenges in applying it to UAV video multi-target tracking. With the continuous development of airborne cameras and embedded systems, computer vision functions given to drones are widely used, including visual navigation, reconnaissance monitoring, and infrastructure inspection. These applications require drones to be able to sense the environment, and make corresponding responses, including identifying the types of objects in the scene, locating the positions of these objects, and determining the precise boundaries of each object. These scene analysis functions correspond to image classification, object detection, and semantic segmentation, so visual target detection and tracking is the most common scenario analysis function module in drone applications [16]. Due to the limited memory and computing power of drones, the algorithm deployment environment is diverse, and target detection and tracking methods based on traditional machine learning and manual features are prone to miss detection or insufficient accuracy [17]. In recent years, the way based on deep learning has become a research hotspot of drone target detection and tracking. However, an object detector of deep learning needs high-performance computing and large running memory to maintain good detection, and the drone platform has limited memory and computing power. The first thing to solve is how to deploy a suitable multi-target detection algorithm. In addition, in the drone video, due to the large tracking picture, the target occupies a small area in the scene, and the tracked target moves quickly. It is still difficult for the drone to effectively track small targets and fast-moving targets.

In response to the above problems, Xue et al. proposed a fusion feature correlation filter [18], which can directly convolve with multi-vector description operators to obtain a single-channel response at the target position, reduce occlusion of background interference, and improve drone target tracking robustness. Still, this way does not fundamentally improve the problem of model drift. Nguyen et al. used LightDenseYOLO to extract training features from the input image, and predicted the position of the marker through the visible light camera sensor on the drone. This way improves the tracking accuracy of drones for small targets. Still, tracking results for multiple targets are not ideal [19]. Rosario et al. proposed a multi-target detection and tracking method that can be used for binocular vision quadrotors. This algorithm effectively reduces the impact of deformation and occlusion on tracking performance [20]. Still, the method has low tracking accuracy and success rate, and it cannot meet the basic tracking needs of drones. Zhang et al. simplified the network structure of the YOLO v3 algorithm, proposed a new method suitable for multi-target detection of drones, and improved the accuracy and speed of multi-target detection [21], but the tracking algorithm based on this detection method has not been effectively verified. Given the above problems, this paper proposes a UAV multi-target tracking algorithm based on YOLO network pruning and optimized SIFT feature extraction.

In view of the above problems, this article considers the needs of UAV target detection and tracking. We make the following contributions. First, we prune the YOLO network and enable UAVs with limited memory and computing power to deploy target detection algorithm. Secondly, we optimize the SIFT feature extraction and improve the accuracy of feature point extraction. Finally, based on the target scale information and location information, we complete data association, meet the UAV tracking needs of fast-moving targets, improve the tracking success rate and accuracy. The paper is organized as follows. Section 2 briefly introduces the SIFT algorithm basic theory and KLT target tracking algorithm. Multi-small target detection and tracking based on improved YOLO and SIFT for drones are presented in Section 3. Section 4 presents the results of our experiments. Finally, Section 5 concludes this paper with a brief consideration for future studies.

2. **Problem Statement and Preliminaries.**

KLT tracking algorithm is a typical feature point tracking method. In the KLT target tracking algorithm, the KLT operator performs second-order derivative on the image and smoothes it with a Gaussian kernel, which easily leads to low accuracy of feature point detection [22]. This paper uses the optimized SIFT algorithm [23] to extract feature points, according to the feature point matching results, combined with the feature point distribution to achieve accurate target positioning.

2.1. **SIFT algorithm basic theory.**

In the SIFT algorithm, in order to obtain the features at different scales, the Gaussian difference kernel and image convolution at different scales are first used to generate a Gaussian difference scale space (DOG scale-space) [24]. The DOG is defined as follows:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) \otimes I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{1}$$

where $I(x,y)$ represents the image, k is a constant of multiples of two adjacent scale spaces, σ is the variance of the Gaussian function distribution, $\otimes$ represents convolution operations, $L(x,y,\sigma) = G(x,y,\sigma) \otimes I(x,y)$ is Gaussian scale space, $G(x,y,\sigma)$ is a scaleable Gaussian function whose expression is

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2} \tag{2}$$

To effectively construct a DOG to detect SIFT feature points, as shown in Figure 1, first preprocess the picture, and then the images are used with different sampling distances to form a layered structure. The results are shown on the left side of Figure 1, and then the adjacent images are subtracted to generate a Gaussian difference image, as shown on the right side of Figure 1. The sample pixel compares with its neighboring 8 pixels, and it is also compared with each of the 9 pixels in the upper and lower adjacent image layers in the pyramid layered image. Through the above steps, relatively stable candidate feature point position information is obtained.

By performing a second-order Taylor expansion on the Gaussian difference scale space function $D(x,y,\sigma)$ at zero, its approximate expression is written:

$$D(x,y,\sigma) \approx D(x_0,y_0,\sigma_0) + \left(x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + \sigma\frac{\partial}{\partial \sigma}\right) D(x_0,y_0,\sigma_0)$$

$$+\frac{1}{2!}\left(x\frac{\partial}{\partial x}+y\frac{\partial}{\partial y}+\sigma\frac{\partial}{\partial \sigma}\right)^2 D(x_0,y_0,\sigma_0) \tag{3}$$

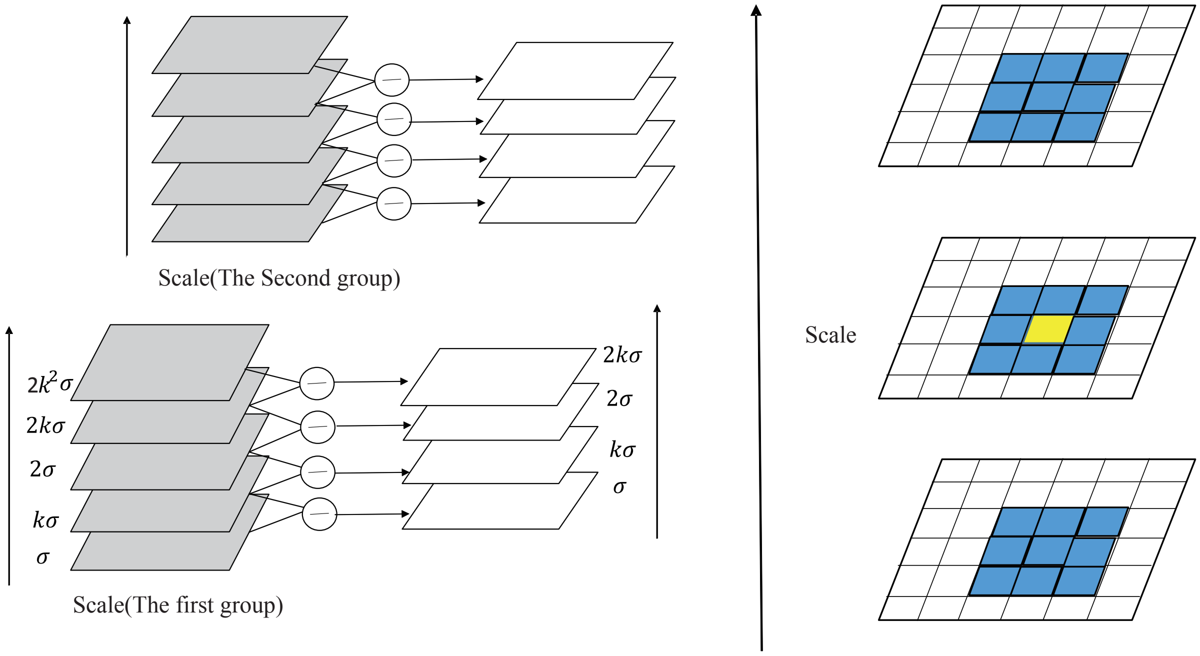


FIGURE 1. SIFT scale pyramid illustration

2.1.1. *Filtering center feature points of image.* In order to filter out the feature points in the center of the image, let $X = (x, y, \sigma)^{\mathrm{T}}$ be the sample point offset and take $X_0 = (0, 0, 0)^{\mathrm{T}}$, and then Formula (5) can be simplified as:

$$D(X) \approx D(X_0) + \left(\frac{\partial D(X_0)}{\partial X}\right)^{\mathrm{T}} X + \frac{1}{2}X^{\mathrm{T}}\left(\frac{\partial^2 D(X_0)}{\partial X^2}\right) X \tag{4}$$

where $\frac{\partial D(X_0)}{\partial X}$ and $\frac{\partial^2 D(X_0)}{\partial X^2}$ satisfy

$$\frac{\partial D(X_0)}{\partial X} = \left(\frac{\partial D(X_0)}{\partial x}, \frac{\partial D(X_0)}{\partial y}, \frac{\partial D(X_0)}{\partial \sigma}\right)^{\mathrm{T}} \tag{5}$$

By deriving Equation (6) and setting it to zero, the extreme value $|D(X)|$ of the function can be obtained. Therefore, the low-contrast feature point of $|D(X)| < D_0$ can be filtered out by selecting the central threshold D_0.

2.1.2. *Filtering edge feature points of image.* For the feature points at the edge of the image, there is a sizeable principal curvature value at the peak of the Gaussian difference function and the edge intersection. Still, the curvature value is small in the vertical direction [25]. Using this property to filter out low-contrast feature points at the edges, consider 2×2 Hessian matrix H as:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{6}$$

Suppose α is a larger eigenvalue and β is a smaller eigenvalue. Let the ratio of α to β be λ, then

$$\frac{\mathrm{Tr}^2(H)}{\mathrm{Det}(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\lambda\beta+\beta)^2}{\lambda\beta^2} = \frac{(\lambda+1)^2}{\lambda} \tag{7}$$

We set the edge threshold to λ_0, whose value can be verified by a small number of sample points. In Equation (7), when $\frac{\mathrm{Tr}^2(H)}{\mathrm{Det}(H)} > \frac{(\lambda_0+1)^2}{\lambda_0}$, the point at the edge should be filtered out. As with the selection of the threshold D_0, the feature points at the edges of the image are filtered. The larger the threshold λ_0, the more feature points will be obtained, and the smaller the threshold λ_0, the fewer feature points will be obtained.

2.2. **KLT target tracking algorithm.** KLT matching algorithm is a matching method based on optimal estimation. It is a method to obtain the best matching position by optimally estimating a certain similarity measure between two feature points. This method has high matching accuracy, and the time consumption is relatively small.

2.2.1. *Target feature point matching.* In the KLT tracking algorithm, first, supposing a feature window containing feature texture information is W, and a translation model is used to describe the change of feature points in the feature window [26]. Let the image corresponding to time t be denoted as $I(x, y, t)$, and the image corresponding to time $t+\tau$ be denoted as $I(x, y, t+\tau)$, and their positions satisfy

$$I(x, y, t+\tau) = I(x - \Delta x, y - \Delta y) \tag{8}$$

In the formula, (x, y) is the characteristic point, and the movement amount $d = (\Delta x, \Delta y)$ is the offset of the point (x, y). For achieving the optimal matching of two image feature points, it is necessary to find d that can minimize the sum of squared intensity differences (SSD) between image frames [27]. Assume that given two images I and J, let SSD be ε, and define ε as:

$$\varepsilon = \iint\limits_{W} [J(X+d) - I(X)]^2 w(X) \mathrm{d}X \tag{9}$$

where W is a given feature window and $w(X)$ is a weighting function. In the simplest case, let $w(X) = 1$, generally d is much smaller than X, so $J(X+d)$ can be Taylor-expanded, remove the high term, and only the first two terms are retained. g is the first-order Taylor coefficient of Taylor-expanded, and then d is derived according to Formula (9). Finally, Equation (9) is simplified to:

$$Zd = \varepsilon \tag{10}$$

where $Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}$, Newton iteration is performed for each point by using Equation (10), and image points can be tracked until a certain accuracy is satisfied.

2.2.2. *Target area positioning.* After obtaining the optimal matching of the image feature points according to the above method, the position information of the strong feature points can be obtained. However, it is necessary to determine the optimal position and shape of the target to achieve target tracking [28]. The position distribution of the optimal feature points is obtained according to the KLT matching, and the position of the rectangular frame containing all the feature points is determined, that is, the target area width w and height h. Assume that the height of the entire image is W, and the width is H. The specific calculation formulas of w and h are

$$\begin{aligned} w &= \max(x_1, x_2, \ldots, x_{Num}) - \min(x_1, x_2, \ldots, x_{Num}) \\ h &= \max(y_1, y_2, \ldots, y_{Num}) - \min(y_1, y_2, \ldots, y_{Num}) \end{aligned} \tag{11}$$

In the formula, $(\min(x_1, x_2, \ldots, x_{Num}), \min(y_1, y_2, \ldots, y_{Num}))$ is the coordinates of the upper left corner of the label frame, (x_i, y_i) is the coordinates of the i-th feature point, $x_i \in [0, W-1]$, $y_i \in [0, H-1]$, $i = (1, 2, \ldots, Num)$, Num is the number of feature points. Because the feature points are distributed on the target and the edges, the position of the target can be determined according to the above method, and the actual size of the target can be fully reflected, and the target area can be located.

3. Proposed Multi-Small Target Detection and Tracking for Drones.

To deploy suitable target detection algorithms for drones and achieve the active tracking of small targets and fast-moving targets at the same time, we propose an overall framework for a multi-threaded algorithm to divide the entire multi-target tracking process of the drone into three threads for processing. As shown in Figure 2, it is the whole algorithm framework. From frame t to frame $t+1$, first is the target detection thread, we prune the YOLO v3 network to UAV-YOLO, thereby reducing the size of the target detection network and deploying it to the drone. Second is the target tracking thread, and we propose an adaptive threshold and minimum distance constraint method to optimize the SIFT feature extraction process. Finally, the data association thread comprehensively considers the target's scale and position information to perform data correlation tasks, and finally completes the entire tracking task.

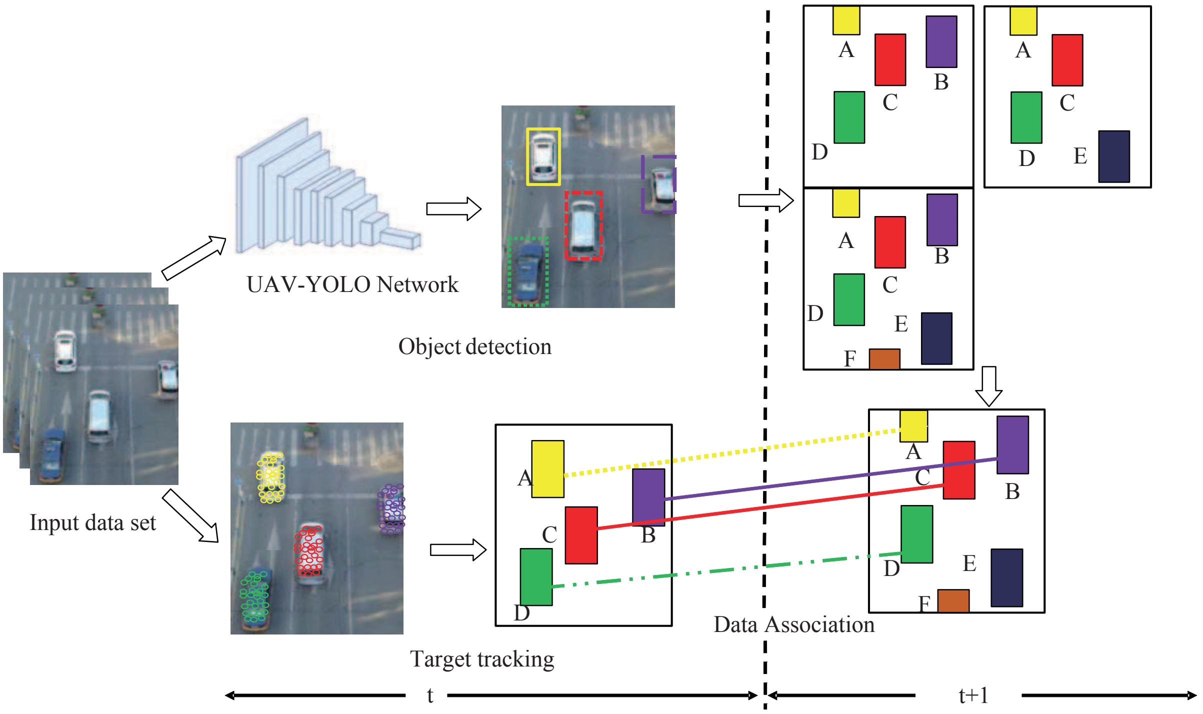


FIGURE 2. (color online) The overall framework of the algorithm in this paper

3.1. UAV-YOLO target detection for drones.

This paper uses YOLO v3 algorithm [29] for target detection. Considering the balance between accuracy and speed of drone target detection, we perform network pruning on YOLO v3. As shown in Figure 3, first deliver basic training and sparse training on the YOLO v3 network, then perform channel pruning based on the obtained network weight and cut off secondary channels with too low weights to achieve the UAV-YOLO prototype. Secondly, fine-tune the UAV-YOLO prototype, test its detection indicators and repeatedly crop according to the results, and finally obtain a UAV-YOLO network suitable for drone target detection.

During the pruning process of the YOLO v3 network, we used the initial network weight file, and VisDrone 2019 DET training set [30] for basic training for pre-training. During the training process, we refer to the detection index and the loss function, and

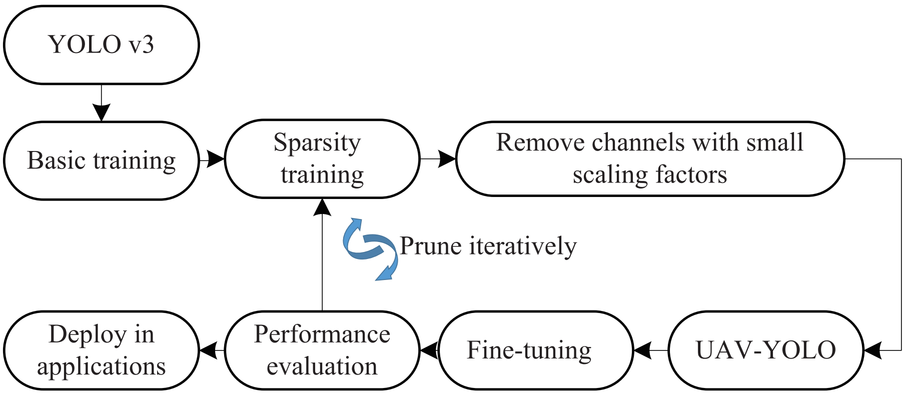


Figure 3. YOLO v3 network pruning process

judgment of the training situation based on the loss curve of the validation set, improve the generalization ability of UAV-YOLO and prevent overfitting of training.

3.1.1. *Sparse training.* In the process of target detection by YOLO v3, in addition to the previous network layer, the subsequent convolutional network layer has a batch normalization (BN) layer to accelerate convergence and improve generalization capabilities. In the process of YOLO v3 network pruning, the more sparse the channel of the depth model is, the more helpful it is for channel pruning [31]. In order to facilitate channel pruning, the BN layer uses the feature y described in each batch to describe when training with the data set, which is expressed as:

$$y = \gamma \times \frac{x - \bar{x}}{\sqrt{\sigma^2 + \varepsilon^2}} + \beta \tag{12}$$

In the formula, γ is the scaling factor for training, x is the feature describing each batch during training of the dataset, $\bar{x}$ and σ^2 are the mean and variance of describing the feature during each training of the dataset, and ε is each batch during the training of the dataset. Describing the error between features, β is the training error factor.

Based on the description of the BN layer, we apply L1 regularization to the training scaling factor γ to perform channel sparsity training. The training loss function is as follows:

$$L = loss_{YOLO} + a \sum_{\gamma \in \Gamma} f(\gamma) \tag{13}$$

where $loss_{YOLO}$ is the initial loss function of the YOLO v3 network, $f(\gamma) = |\gamma|$ is the L1 norm, Γ is the constraint of the scaling factor γ, and a is the penalty factor used to balance the initial loss function term and training scaling of the YOLO v3 network.

3.1.2. *Channel pruning and network adjustment.* The YOLO v3 network has a maximum pooling layer, an upsampling layer, a convolutional layer, a routing layer, and a direct connection layer. After sparse training, we perform different operations to achieve network channel pruning based on different characteristics of different network layers. For the convolutional layer, to prevent the network from being degraded or the model cannot be restored due to excessive pruning. We introduce a global threshold λ and local safety threshold θ to determine whether to trim the channel. We construct a pruning mask for all convolutional layers according to the global safety threshold λ and local safety threshold θ, where the global threshold λ is set to n percentages of all convolutional layers $|\gamma|$, and we sort all $|\gamma|$ according to the global size, trim the channels below n percentages to control the global trimming rate. The local security threshold θ is set to k percentages of

$|\gamma|$ in a specific convolutional layer in a hierarchical manner to prevent excessive pruning on a single convolutional layer and ensure the integrity of the network connection. As shown in Figure 4, for the i-th convolutional layer, for channels C_{i1} to C_{in}, we sort $|\gamma|$ in the i-th convolutional layer. According to the local safety threshold θ, the orange color that has little effect on the prediction result will be directly deleted, and the blue channels are retained. For the overall convolutional neural network, the channels that have little effect on the prediction result are also deleted according to the global threshold of λ.

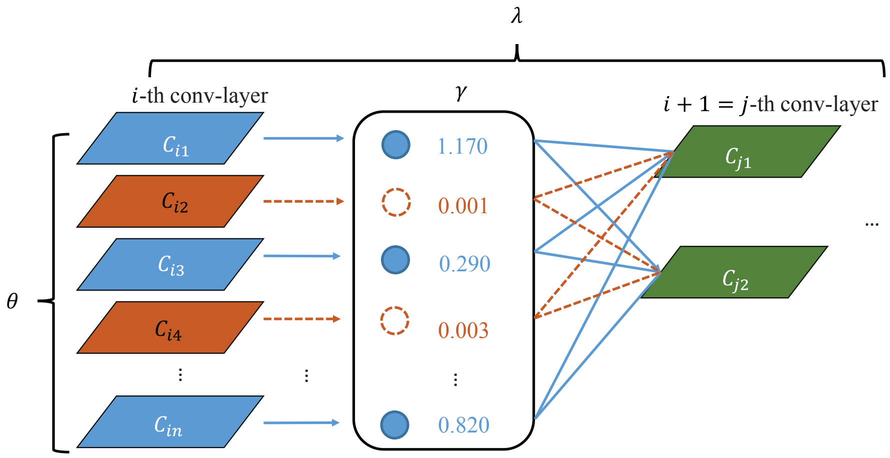


FIGURE 4. (color online) YOLO v3 convolutional layer channel pruning

For the maximum pooling layer and the upsampling layer, they have nothing to do with the channel signal, and we directly crop these two network layers. For the routing layer, we choose to keep it and use it to connect the pruning masks obtained from the convolution layer. For the directly connected layer, we traverse the pruning masks of all directly connected layers and perform or operate on these pruning masks to generate the final pruning mask.

During the tracking of multiple targets by drones, especially in the detection of small targets, the effect of channel pruning usually has a significant impact on the detection performance. After the channel pruning is completed, we perform a network fine-tuning operation on the trimmed model. During the network fine-tuning process, the UAV-YOLO is directly retrained using the same training parameters as the YOLO v3 basic training to compensate for potential temporary degradation.

3.2. **Small target SIFT feature detection.**

3.2.1. *Adaptive threshold.* By effectively constructing a Gaussian difference scale-space function $D(x, y, \sigma)$, extreme points in a Gaussian difference image can be obtained. Although the location of the feature points extracted by the SIFT algorithm is accurate, there are still many feature points that are sensitive to noise and located at the edges. At the same time, the feature point distribution is too concentrated. To this problem, this paper proposes a dynamic threshold method and a feature point distance constraint mechanism to optimize it. We define the adaptive thresholds D_0 and λ_0 as:

$$\begin{aligned} D_0 &= k_1 \Big/ N \times \sum_{i=1}^{N} \left| D\left(\hat{X}_i\right) \right| \\ \lambda_0 &= k_2 \Big/ N \times \sum_{i=1}^{N} \mathrm{Tr}^2(H) \Big/ \mathrm{Det}(H) \end{aligned} \tag{14}$$

In the formula, k_1 and k_2 are proportional coefficients, which can be selected according to the matching speed of feature points. For situations where the matching speed of feature points is not high, $k_1 < 1$ and $k_2 > 1$ can be selected. At this time, D_0 will be smaller and λ_0 will be larger. This will ensure that more feature point information is obtained. For those with high matching speed requirements, In the case, the extracted feature points cannot be too many, and $k_1 > 1$ and $k_2 < 1$ are preferable.

3.2.2. *Feature point minimum distance constraint mechanism.* The feature points obtained by the SIFT algorithm are prone to clustering in some small areas, which is not conducive to subsequent feature point matching. Therefore, we have added a minimum distance constraint mechanism. The specific ideas are:

1) Let the minimum constraint distance between two feature points be $d_{\min}$ and traverse the feature points.

2) If the distance between two feature points is less than $d_{\min}$, remove the relatively small feature points, which will reduce the matching time of the feature points to a certain extent without affecting the approximate shape and position of the target.

3) If the target image to be processed is large, you can set $d_{\min}$ to be larger to increase the feature extraction speed and meet the tracking speed requirements. If the target image to be processed is small, you can set $d_{\min}$ to be smaller to ensure that there are still reliable feature points on tiny targets.

3.3. **Data association based on scale information and position information.** In the process of multi-target tracking by UAV using KLT, the task of data association is to select a hypothetical data set H_i and divide the detection data set D into disjoint subsets $S_1, S_2, \ldots, S_J$, where each subset S_j contains all detections of points. To achieve data correlation for multi-target tracking of drones, we predict the scale information s_n and position information x_n of the tracked target to make data correlation.

3.3.1. *Scale information prediction.* In the process of multi-target tracking, we hope to obtain the scale information s_n of the tracked target in the detection of each frame. In each trajectory of the tracked target, the scale information detected for the first time cannot be predicted based on any previous prior knowledge, so we assume that the global target scale information s_n is a lognormal distribution:

$$\ln s_n \sim N\left(\mu_p, \sigma_p^2\right) \tag{15}$$

In the formula, μ_p and σ_p^2 represent the average and variance of the global target scale information s_n, respectively. We use the hypothesized global target scale information s_n as prior knowledge to more effectively predict the scale information of the next detection in the trajectory:

$$\ln \frac{s_n}{s_{n-1}} \bigg| c_j \sim N\left(0, \delta_t \sigma_j^2\right) \tag{16}$$

In the formula, δ_t is the time difference between the detected frames, σ_j^2 is the variance of different trajectory scale information, c_j is used to distinguish the type of trajectory, and c_j can be extended to represent many different types of moving objects, such as vehicles, pedestrians, and bicycles. Each class will have a separate motion model to facilitate classification.

3.3.2. *Location information prediction.* When considering the target position information x_n, we used a similar selection method, assuming that the target and the surrounding environment are evenly distributed in the image, and the probability density b of the position information x_n on the image area $p(x_n)$ can be expressed as:

$$p(x_n) = \frac{b}{s_k^2} \tag{17}$$

where s_k is the average pixel size of the tracked object. We establish a constant-speed motion model for the tracked target, expressed as:

$$\begin{cases} x_p = x_{n-1} + \delta_t v_p \\ \sum p = \delta_t \sum v \end{cases} \tag{18}$$

In the formula, v_p is the KLT tracking result in the frames before and after the detection, x_p is the position information obtained according to v_p, and $\sum v$ is the speed change of the tracked target caused by unknown acceleration, including changes in the speed and direction of motion. When the tracked target moves for a long time, the error $\sum p$ caused by $\sum v$ is still substantial. Therefore, we use a Kalman filter to estimate the possible situation at the target position. For each step of the KLT motion estimation value F, we take it into the motion model, and then obtain the target position information y at each step more accurately as:

$$\begin{cases} x_n = x_{n-1} + \sum p \left(\sum p + \delta_t \sum klt\right)^{-1} (x_{n-1} + y - x_p) \\ \sum y = \left(I - \sum p \left(\sum p + \delta_t \sum klt\right)^{-1}\right) \sum p \end{cases} \tag{19}$$

In the formula, I is a unit matrix, and $\sum klt$ represents a random error accumulated by the tracking speed of the KLT feature.

3.3.3. *Data association.* After we obtain the scale information s_n and position information x_n of the tracking target, we use the method based on the MDL principle to find the optimal representation relationship between the hypothesized data set H_i and the detection data set D. The correlation coefficient L required for them to correlate to a given accuracy depending on the corresponding likelihood function $L(D|H)$ can be expressed as:

$$L(D|H) + L(H) = -\log(p(D|H_i)p(H_i)) \tag{20}$$

First, we consider the encoding of the hypothetical set H_i, which requires that each detected object is assigned to a trajectory, and each path is assigned a type label. In multi-target tracking, if the length of the tracked target is equal to the period in the trajectory, the correlation is performed first. At this time, the probability of the data set is expressed as:

$$p(H_i) = j! \prod_{T_j \in H_i} \left(\frac{|T_j|}{|D|}\right)^{|T_j|} p(c_j) \tag{21}$$

In the formula, c_j is the type of trajectory, $p(c_j)$ is the priority of different trajectories, $|D|$ is the cardinality of the detection data D, and T_j is a subset of the hypothesized data set H_i.

In the process of data association, different trajectories that are decomposed must be associated. We decompose the likelihood function $L(D|H)$ as the component of each trajectory, let c_j be the n-th detection in trajectory T_j, where the index n only indicates the order within the trajectory, and then under the hypothetical data set, it is expressed according to the probability obtained by detection for:

$$p(D|H_i) = \prod_{T_j \in H_i} \left[p\left(d_1^j|c_j\right) \prod_{d_n^j \in T_j \backslash d_1^j} p\left(d_n^j|d_{n-1}^j, c_j\right) \right] \tag{22}$$

For each target detection result, we want to find the scale information s_n and position information x_n of the target, and we represent the likelihood function of a single detection as scale information s_n and position information x_n:

$$\begin{cases} p\left(d_1^j|c_j\right) = p(s_1)p(x_1) \\ p\left(d_n^j|d_{n-1}^j, c_j\right) = p(s_n|s_{n-1})p(x_n|x_{n-1}, c_j) \end{cases} \tag{23}$$

Based on the target's scale information s_n and position information x_n, we finally take Formula (23) into Formula (22) and assume:

$$p(D|H_i) = \prod_{T_j \in H_i} \left[p(s_1)p(x_1) \prod_{s_n \in T_j \backslash s_1} p(s_n|s_{n-1})p(x_n|x_{n-1}, c_j) \right] \tag{24}$$

According to Formula (20), Formula (21), and Formula (24), from the scale information and position information of the target, we can finally obtain the optimal representation relationship between the hypothesized data set and the detection data set.

3.4. **Algorithm flow.** The flow of the algorithm in this paper is shown in Algorithm 1.

Algorithm 1 Multi-small target detection and tracking for drones.

```
Initialization: read sequence frame number $N_v$ and current sequence frame target frame truth value $BBox_v$;
for Each $t \in [1, N_v]$ frame in video sequence $v$ do
    Obtained $D_t$ using UAV-YOLO;
    According to the detection results $D_t$, use the optimized SIFT feature to obtain the tracking target feature;
    Get scale information: $s_n \Leftarrow \ln \frac{s_n}{s_{n-1}}|c_j \sim N\left(0, \delta_t \sigma_j^2\right)$
    Get location information: $x_n \Leftarrow x_n = x_{n-1} + \sum p\left(\sum p + \delta_t \sum klt\right)^{-1}(x_{n-1} + y - x_p)$
    Assume data set $H_i$ is encoded first: $p(H_i) \Leftarrow j \prod_{T_j \in H_i} \left(\frac{|T_j|}{|D|}\right)^{|T_j|} p(c_j)$
    Detection encoding of hypothetical data set $H_i$:
    $p(H_i) \Leftarrow \prod_{T_j \in H_i} \left[ p(s_1)p(x_1) \prod_{s_n \in T_j \backslash s_1} p(s_n|s_{n-1})p(x_n|x_{n-1}, c_j) \right]$
end for
Complete data association: $L(D|H) + L(H) = -\log(p(D|H_i)p(H_i))$
Optimized SIFT features using adaptive threshold and minimum distance constraints;
Matching the best feature points: $Zd = \varepsilon$;
Determine the target position: $\begin{matrix} w = \max(x_1, x_2, \ldots, x_{Num}) - \min(x_1, x_2, \ldots, x_{Num}) \\ h = \max(y_1, y_2, \ldots, y_{Num}) - \min(y_1, y_2, \ldots, y_{Num}) \end{matrix}$;
Multi-target tracking based on target position, output $\Gamma$
```

The entire algorithm process can be summarized into the following steps.

Step 1: Initialize the video sequence, and read the frame number of the video sequence and the real value of the labeled frame.

Step 2: Prune the YOLO v3 network to get UAV-YOLO.

Step 3: Use the adaptive threshold and minimum distance to improve the SIFT algorithm for feature point detection.

Step 4: Based on target's scale and position information, compete data association.

Step 5: The feature points obtained by the improved SIFT algorithm detection are matched and targeted by the KLT algorithm to achieve multi-target tracking.

4. **Main Results.**

To effectively evaluate the performance of the algorithm in this paper, all tests were performed on Intel (R) Core (TM) i5-2450M CPU @ 2.50GHZ, 4GB memory, NVIDIA GeForce GTX 1660, and the algorithm framework was implemented based on the Pytorch deep learning framework.

This experiment is verified from four aspects: YOLO v3 network pruning, UAV-YOLO performance evaluation, optimized SIFT feature small target detection and drone multi-target tracking. The algorithm-related initialization parameter settings in the experiment are shown in Table 1.

Table 1. Algorithm related initialization parameters

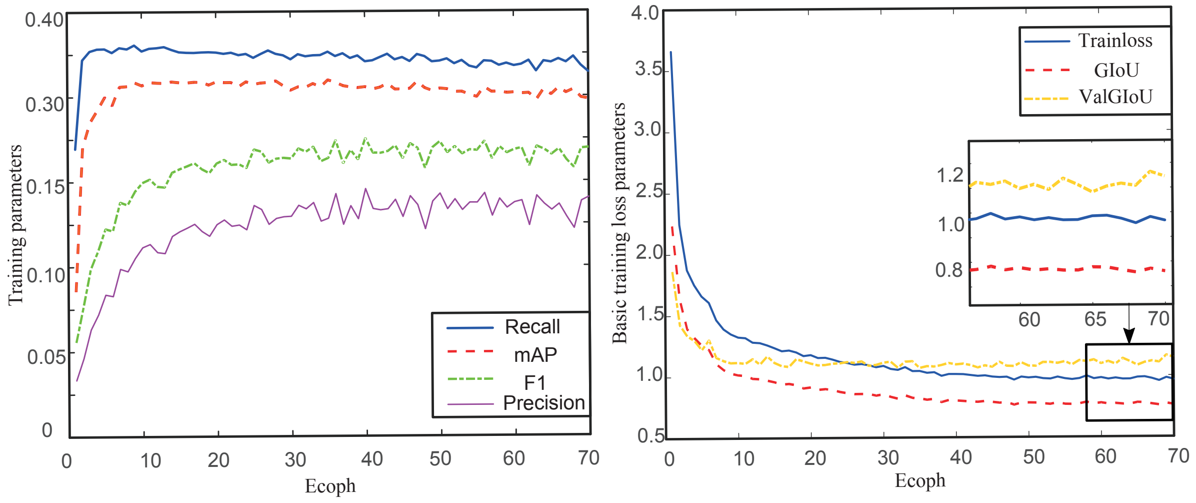
Parameter name	Parameter size
Sparse training scale factor γ	0.25
Sparse training error factor β	0.10
Penalty factor a	0.01
Global threshold λ	0.93
Local threshold θ	0.10
Training batch size	20
Network directly connected layer pruning layers	16
Learning rate	0.001
Minimum constraint distance $d_{\min}$	5
Adaptive threshold scale factor	$k_1 = 1.5$, $k_2 = 0.8$

4.1. **YOLO v3 network pruning.**

According to the pruning process, the pruning experiments of the YOLO v3 network were performed basic training, sparse training, network pruning, and adjustment, and the COCO dataset was used to evaluate the target detection of UAV-YOLO.

4.1.1. *Basic training.* The basic training of network pruning is shown in Figure 5, where 5(a) is the index parameter for basic training detection, and 5(b) is the basic training loss function. It can be seen from Figure 5(a) that during the entire training process, the detected index parameters Recall, mAP, Precision, and F1 basically reached the highest benchmark after 30 rounds of basic training, and large fluctuations occurred in the subsequent training process. From the details of the basic training loss function in Figure 5(b), it can be seen that in the 60-70 rounds of training, during the first 60 rounds of basic training, the Trainloss of the entire training continues to decline. During the 60-70 rounds of basic training, the Trainloss and GIoU of the set are still declining. The verification set valGIoU has picked up. At this time, the training of the network is stopped to prevent overfitting in training.

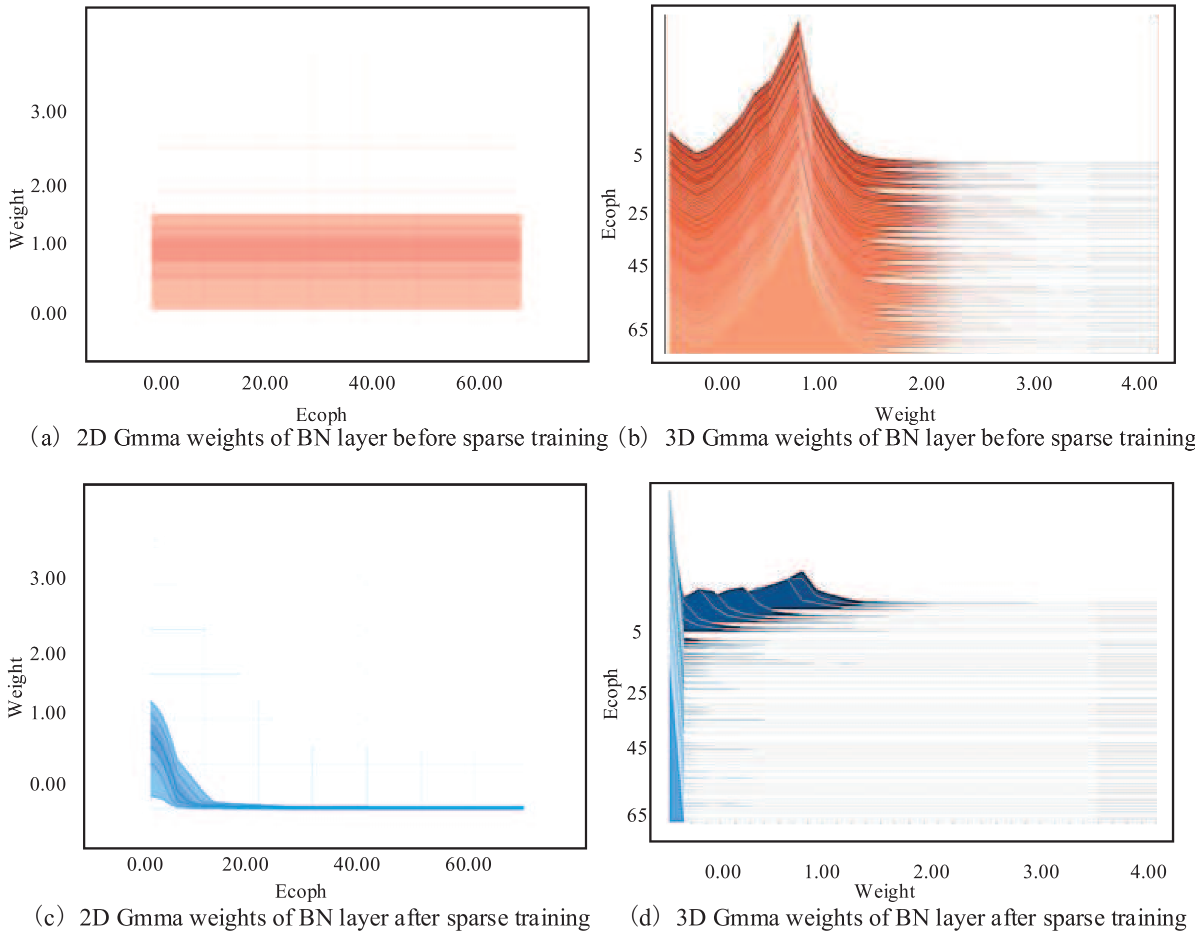
4.1.2. *Sparse training and network pruning.* After the basic training was completed, this experiment conducted 50 rounds of sparse training. Figure 6 shows the sparseness of the BN layer of the YOLO v3 network. Figures 6(a) and 6(b) show the Gmma weights of the BN layer before sparse training, and Figures 6(c) and 6(d) show the Gmma weights



(a) Detection parameters of basic training

(b) Basic training loss function

FIGURE 5. VisDrone 2019 DET basic training related parameters



（a） 2D Gmma weights of BN layer before sparse training （b） 3D Gmma weights of BN layer before sparse training

（c） 2D Gmma weights of BN layer after sparse training （d） 3D Gmma weights of BN layer after sparse training

FIGURE 6. Sparse training changes at the BN layer of YOLO v3 network

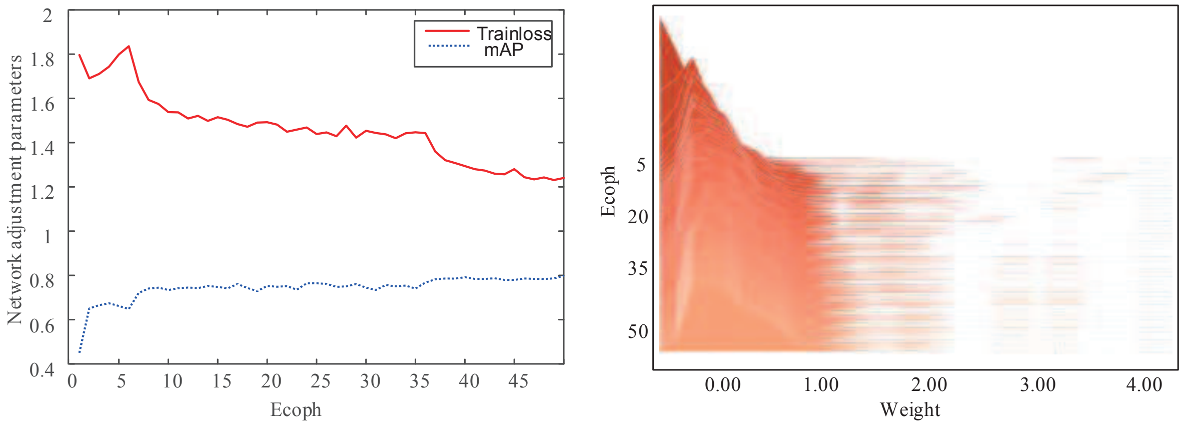
of the BN layer after sparse training. It can be seen that before the sparse training, from Figures 6(a) and 6(b) the Gmma weights of the BN layer before the sparse training, it can be seen that Gmma is generally distributed near one like a normal distribution. After 50 rounds of sparse training, Gmma is gradually compressed to close to 0, and the output value of the channel close to 0, and we cut it out.

After sparse training, the network channels are pruned. It can be seen from the change of the model parameters in Table 2 before and after pruning that the mAP of the model changed from 0.31 to 0.30 after pruning, which was a decrease of 2.95%. After pruning, the mAP dropped to 0.29, but the model parameters were reduced by 98.7%. With a slight decline in the accuracy of the entire network model, the network model was successfully cut and compressed.

Table 2. Changes in model parameters before and after pruning

Metric	Before	After prune channels	After prune layers
mAP	0.310843	0.301677	0.289565
Parameters	62573334	893513	825825
Inference	0.0944	0.0227	0.0186

4.1.3. *Network adjustment.* Given the decline in the network model after pruning, we fine-tuned the detection accuracy of the network. Figure 7 shows the fine-tuning of the YOLO v3 network. The entire network was fine-tuned for 50 rounds of training, which can be obtained from Figure 7(a). The mAP continues to recover with the increase of the number of training rounds, recovering to more than 0.31, and can be obtained from Figure 7(b). The BN layer has also begun to show a normal state.



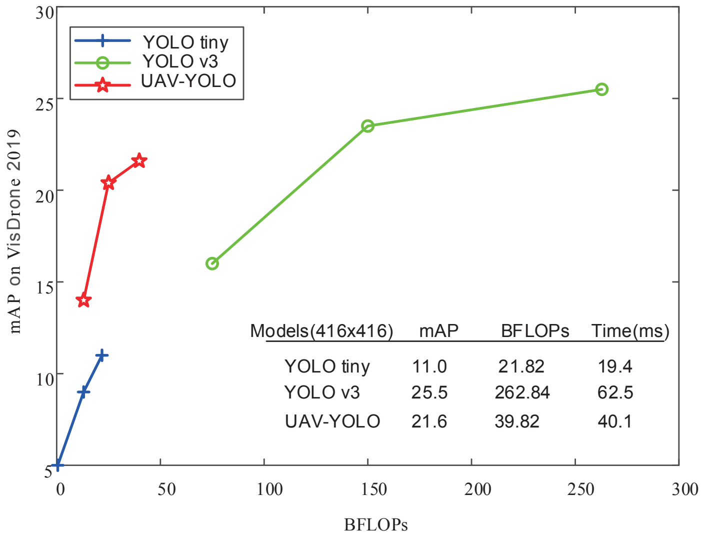
(a) Network fine-tuning related index parameters

(b) YOLO v3 network fine-tunes BN layer 3-dimensional Gmma weights

Figure 7. YOLO v3 network fine-tuning

4.2. **UAV-YOLO performance evaluation.** To meet the task requirements of real-time detection of drones, as shown in Figure 8, we compare the UAV-YOLO algorithm proposed in this paper with the original YOLO v3 and YOLO tiny algorithms on the VisDrone 2019 DET benchmark dataset.

From the evaluation results in Figure 8, it can be seen that compared with YOLO v3, the detection time of UAV-YOLO in this algorithm is 40.1 ms, which can fully meet the needs of drone target tracking, and UAV-YOLO only decreases the detection accuracy by 3.9



Models(416x416)	mAP	BFLOPs	Time(ms)
YOLO tiny	11.0	21.82	19.4
YOLO v3	25.5	262.84	62.5
UAV-YOLO	21.6	39.82	40.1

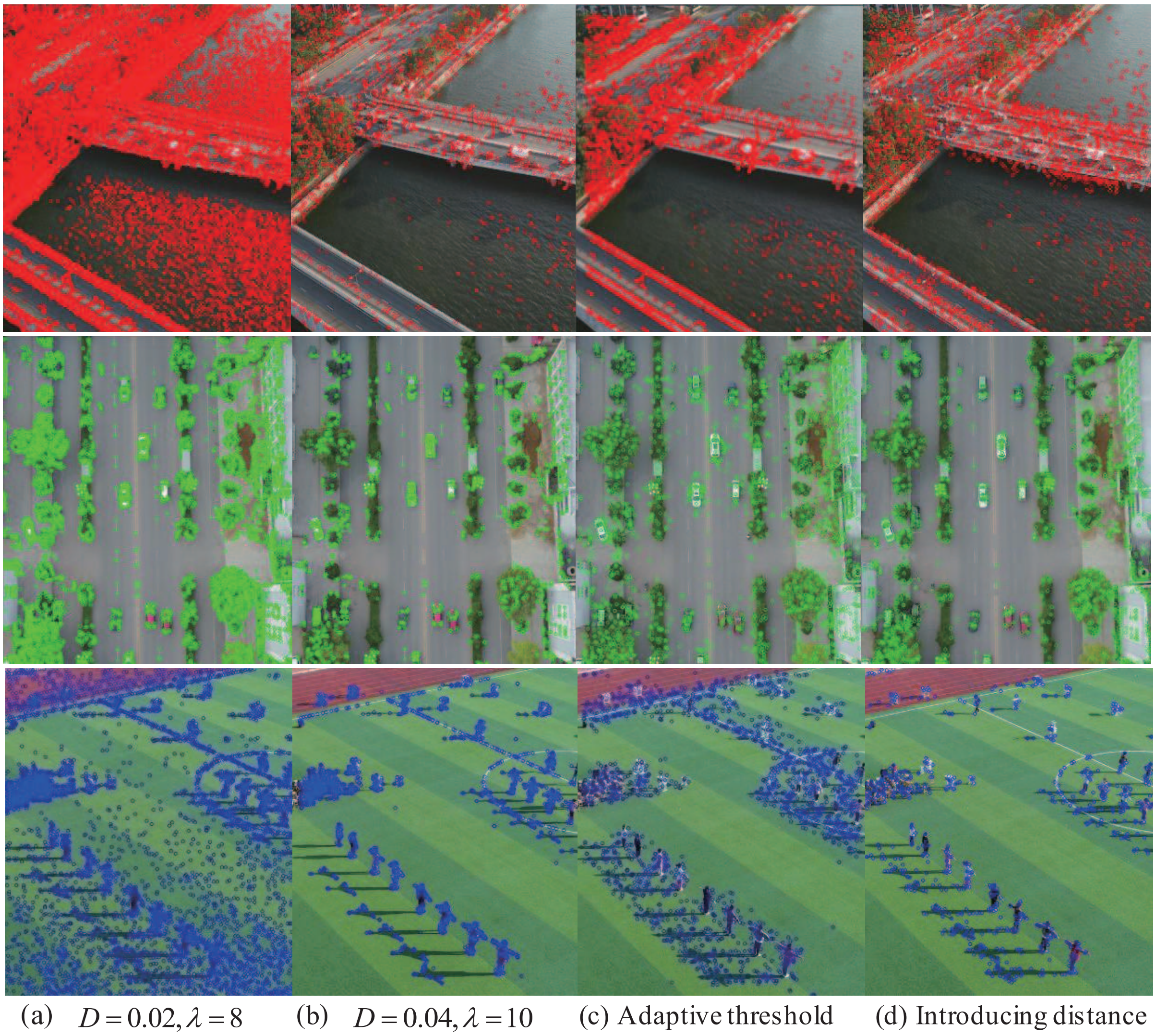
Figure 8. Performance of detection algorithm on VisDrone 2019

mAP, BFLOPs have decreased by 84.85%, and the test run time has reduced by 35.84%. UAV-YOLO is more comfortable to implement for drone deployment. Compared with YOLO tiny, UAV-YOLO has a calculation speed that meets the real-time requirements. The accuracy mAP was improved by 49.07%. This shows that UAV-YOLO after network pruning can well balance the tracking accuracy and detection speed, and can meet the application requirements of UAV multi-target detection and tracking.

4.3. **Feature detection based on optimized SIFT algorithm.** The adaptive threshold and minimum distance constraints are used to improve the SIFT feature to improve the detection of small targets. Figure 9 shows the experimental results of feature points detection based on the optimized SIFT and fixed threshold method. We performed SIFT feature point detection on the sample of the VisDrone 2019 DET data set and obtained $D = 0.03$, $\lambda = 9$. Based on this sample result, we set two sets of data with fixed thresholds to $D = 0.02$, $\lambda = 8$ and $D = 0.04$, $\lambda = 10$, which are used to perform adaptive thresholds and introduce distance compared.

It can be seen that the fixed threshold method has obvious limitations. Among them, the feature point detection in Figure 9(a) is too dense to determine the target accurately. The feature point detection in Figure 9(b) is sparse compared to Figure 9(a), the small target feature points of partial exposure are insufficient. Therefore, the selection of the threshold value has a significant influence on the result of corner detection. Once the threshold value is set unreasonably, it is easy to cause the feature points to be extracted too densely or unevenly distributed. Figure 9(c) shows the experimental result of the feature points obtained by the adaptive threshold method. It can be seen that compared with the fixed threshold, the adaptive threshold method considers the contrast of the feature points and the change of the primary curvature as a whole. The threshold value adapted to the target change, and the distribution of feature points is relatively uniform. From Figure 9(c), we can also see that the phenomenon of feature point clustering occurs in some small areas. Through the method of the minimum distance constraint proposed in this paper, the feature point detection effect shown in Figure 9(d) is obtained. It can be seen that the feature point clustering phenomenon has been significantly improved.

4.4. **Multi-target tracking for drones.** The VisDrone 2019 MOT benchmark data set was used to evaluate the multi-target tracking of the algorithm in this paper. The test



(a) $D = 0.02, \lambda = 8$ (b) $D = 0.04, \lambda = 10$ (c) Adaptive threshold (d) Introducing distance

FIGURE 9. Optimized SIFT feature detection on VisDrone 2019 DET

results of multi-target tracking of drone video sequences in three common scenarios are shown in Figure 10. The scene in the picture includes market, intersection and street. The tracking targets include pedestrians, bicycles, cars, buses, and tricycles.

From these video sequences, it can be seen that for multi-target tracking from the perspective of drones, the algorithm in this paper meets the accuracy requirements of multi-target tracking for drones. From the left side of Figure 10, it can be seen that for tracking targets near video sequences even in poor lighting conditions, the algorithm in this paper has a better tracking effect. At the same time, in middle and right scenes in Figure 10, for small distant targets and fast-moving targets at common intersections, the algorithm in this paper can still accurately locate and track the targets to meet the multi-target tracking needs of drones. In addition, the algorithm in this paper has a tracking evaluation of 75 FPS on the VisDrone20110 MOT benchmark dataset, which has a good tracking speed and meets the needs of real-time drone tracking.

The performance indicators of the algorithm in this paper and other UAV multi-target tracking algorithms are shown in Table 3. MOTA and MOTP are the main evaluation indicators of multi-target tracking performance. IDF1, FAT, MT, ML, FP, FN, IDS and FM are a secondary evaluation indicator. As can be seen from the evaluation indicators in Table 3, the main evaluation indicators of this algorithm, MOTA, and MOTP, both meet

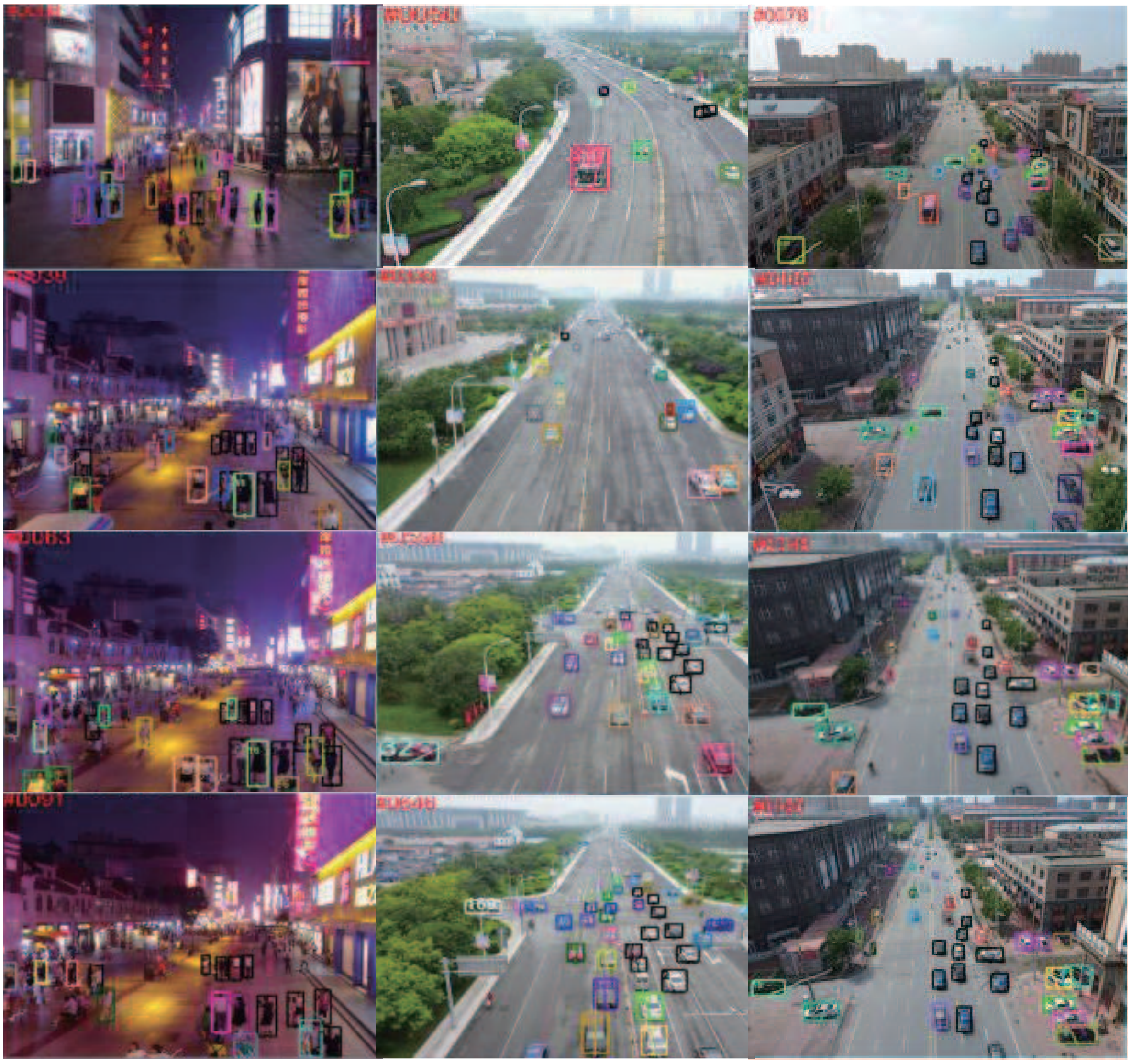


FIGURE 10. The algorithm video sequence of on the VisDrone 2019

TABLE 3. VisDrone 2019 MOT benchmark dataset evaluation

Method	MOTA↑	MOTP↑	IDF1↑	FAT↓	MT↑	ML↓	FP↓	FN↓	IDS↓	FM↓
GOC_ECO	36.9	**75.8**	46.5	**0.29**	205	589	**5445**	86399	354	**1090**
SCTrack	35.8	75.6	45.1	0.39	211	550	7298	85623	798	2042
Ctrack	30.8	73.5	51.9	1.95	**369**	375	36930	62819	1376	2190
FRMOT	33.1	73	50.8	1.15	254	463	21736	74953	1043	2534
GOC	38.4	75.1	45.1	0.54	244	496	10179	78724	1114	2012
IHTLS	36.5	74.8	43	0.94	245	446	14564	75361	1435	2662
TBD	35.6	74.1	45.9	1.17	302	419	22080	70083	1834	2307
H2T	32.2	73.3	44.4	0.95	214	494	17889	79801	1269	2035
CMOT	31.5	73.3	51.3	1.42	282	435	26851	72382	789	2257
CEM	5.1	72.3	19.2	1.12	105	752	21180	116363	1002	1858
Ours	**38.7**	75.7	**47.3**	0.57	241	**366**	10897	**61062**	**304**	1903

the requirements for the accuracy and success rate of drone tracking. The overall evaluation performance is much better than SCTrack, Ctrack, FRMOT, and other methods. The evaluation index MOTP is equivalent to the optimal way GOC_ECO, but its MOTA index is 4.7% higher than GOC_ECO, and MOTA is also superior to the optimal method GOC. In addition to the main evaluation indicators, we can also see that the secondary evaluation indicators IDF1, ML, and FN of the algorithm in this paper perform optimally. At the same time, in the drone tracking sequence, there are many small targets to track, and the ID changes detected for the targets. The problem is that the IDS of the algorithm in this paper is only 304. Compared with GOC_ECO, ID change is reduced by 14.12%.

Figure 11 shows the histogram and curve chart of the algorithm MOTA and MOTP in this paper. From this figure, it can be seen intuitively that the main evaluation indicators of the algorithm in this paper perform well, and both MOTA and MOTP meet the task of multi-target tracking of drone claim.

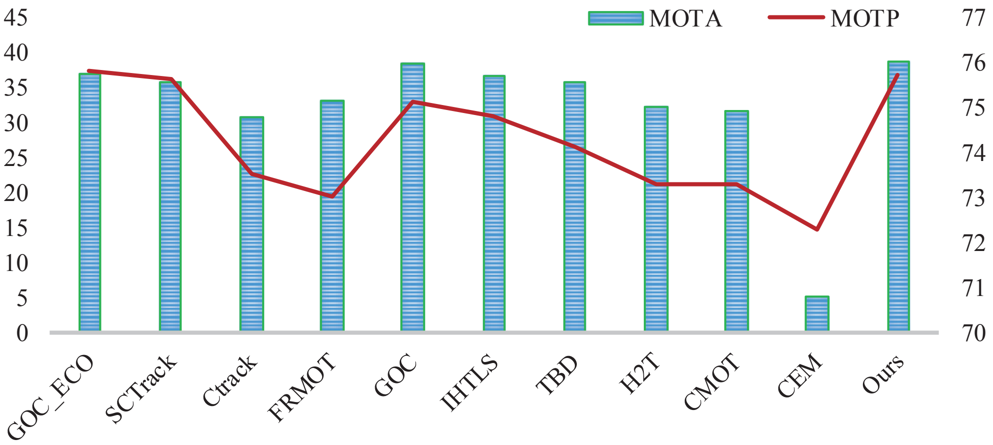


Figure 11. VisDrone 2019 MOT main evaluation indicators MOTA, MOTP

5. **Conclusions and Future Work.** This paper proposes a UAV target detection and tracking method based on improved YOLO and SIFT. First, to balance the requirements of target detection accuracy and speed, we prune the YOLO network, enabling the drone to deploy a smaller detection network. Secondly, the SIFT feature extraction is optimized, and the accuracy and success rate of tracking are improved through the adaptive threshold and minimum distance constraints, and effective detection and tracking of small targets are achieved. Finally, considering the needs of drones to track fast-moving targets, based on scale and position information, we compete for data association, which improves the real-time tracking of fast-moving targets.

However, in the complex environment, the tracked target appears in-plane rotation, deformation, lighting changes, etc., and the tracking effect of the algorithm in this paper is not good. Therefore, for drones, finding a method of multi-target detection and tracking that is robust in complex environments is the direction of multi-target tracking in the future.

REFERENCES

[1] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao and T.-K. Kim, Multiple object tracking: A literature review, *arXiv Preprint, arXiv:1409.7618*, 2014.

[2] A. B. Poore and S. Gadaleta, Some assignment problems arising from multiple target tracking, *Mathematical and Computer Modelling*, vol.43, nos.9-10, pp.1074-1091, 2006.

[3] E. Gundogdu and A. A. Alatan, Good features to correlate for visual tracking, *IEEE Trans. Image Processing*, vol.27, no.5, pp.2526-2540, 2018.

[4] S. Sharma, A. Khachane and D. Motwani, Real time multi-object tracking using TLD framework, *Proc. of 2016 International Conference on Inventive Computation Technologies (ICICT)*, vol.2, pp.1-6, 2016.

[5] P. Mukilan and A. Wahi, An efficient human object detection and tracking with the aid of morphological operation and optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.11, no.4, pp.1139-1153, 2015.

[6] L. Leal-Taixé, C. Canton-Ferrer and K. Schindler, Learning by tracking: Siamese CNN for robust target association, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp.418-425, 2016.

[7] A. Patrik, G. Utama, A. A. S. Gunawan, A. Chowanda, J. S. Suroso and W. Budiharto, Modeling and implementation of object detection and navigation system for quadcopter drone, *ICIC Express Letters*, vol.13, no.6, pp.461-468, 2019.

[8] J. A. R. Castillo, A. R. Rivera and O. Chae, Robust facial recognition based on local Gaussian structural pattern, *International Journal of Innovative Computing, Information and Control*, vol.8, no.12, pp.8399-8413, 2012.

[9] M. A. Sayeed and R. Shree, Optimizing unmanned aerial vehicle assisted data collection in cluster based wireless sensor network, *ICIC Express Letters*, vol.13, no.5, pp.367-374, 2019.

[10] W. Li, J. Yao, T. Dong and H. Li, Object tracking based on fragment template and multi-feature adaptive fusion, *Proc. of the 8th International Symposium on Computational Intelligence and Design (ISCID)*, vol.2, pp.481-484, 2015.

[11] J. Zhang, L. L. Presti and S. Sclaroff, Online multi-person tracking by tracker hierarchy, *Proc. of the 9th International Conference on Advanced Video and Signal-Based Surveillance*, pp.379-385, 2012.

[12] Y. Guan, X. Chen, D. Yang and Y. Wu, Multi-person tracking-by-detection with local particle filtering and global occlusion handling, *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, pp.1-6, 2014.

[13] M. Gan, Y. Cheng, Y. Wang and J. Chen, Hierarchical particle filter tracking algorithm based on multi-feature fusion, *Journal of Systems Engineering and Electronics*, vol.27, no.1, pp.51-62, 2016.

[14] J. Ju, D. Kim, B. Ku, D. K. Han and H. Ko, Online multi-object tracking based on hierarchical association framework, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp.1273-1281, 2016.

[15] M. Hummel, C. Rosenkranz and R. Holten, The role of communication in agile systems development, *Business & Information Systems Engineering*, vol.5, no.5, pp.343-355, 2013.

[16] Z. Zou, Z. Shi, Y. Guo and J. Ye, Object detection in 20 years: A survey, *arXiv Preprint, arXiv:1905.05055*, 2019.

[17] P. Emami, P. M. Pardalos, L. Elefteriadou and S. Ranka, Machine learning methods for solving assignment problems in multi-target tracking, *arXiv Preprint, arXiv:1802.06897*, 2018.

[18] X. Xue, Y. Li and Q. Shen, Unmanned aerial vehicle object tracking by correlation filter with adaptive appearance model, *Sensors*, vol.18, no.9, 2018.

[19] P. H. Nguyen, M. Arsalan, J. H. Koo, R. A. Naqvi, N. Q. Truong and K. R. Park, LightDenseYOLO: A fast and accurate marker tracker for autonomous UAV landing by visible light camera sensor on drone, *Sensors*, vol.18, no.6, 2018.

[20] J. R. B. Del Rosario, J. A. L. Angeles, A. J. P. Cabebe, J. J. D. Co, D. E. S. Santamaria, A. A. Bandala and E. P. Dadios, Integrative review of the development of a multi-object tracker from a dual camera system in an unmanned aerial vehicle (UAV), *Proc. of the 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp.1-4, 2017.

[21] P. Zhang, Y. Zhong and X. Li, SlimYOLOv3: Narrower, faster and better for real-time UAV applications, *Proc. of the IEEE International Conference on Computer Vision Workshops*, 2019.

[22] A. S. Mian, Realtime visual tracking of aircrafts, *2008 Digital Image Computing: Techniques and Applications*, pp.351-356, 2008.

[23] D. G. Lowe, Object recognition from local scale-invariant features, *Proc. of the 7th IEEE International Conference on Computer Vision*, vol.2, pp.1150-1157, 1999.

[24] D. J. Williams and M. Shah, A fast algorithm for active contours and curvature estimation, *CVGIP: Image Understanding*, vol.55, no.1, pp.14-26, 1992.

[25] K. He, X. Zhang, S. Ren and J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.37, no.9, pp.1904-1916, 2015.

[26] N. Wang, J. Shi, D.-Y. Yeung and J. Jia, Understanding and diagnosing visual tracking systems, *Proc. of the IEEE International Conference on Computer Vision*, pp.3101-3109, 2015.

[27] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan and S. Wang, Learning dynamic siamese network for visual object tracking, *Proc. of the IEEE International Conference on Computer Vision*, pp.1763-1771, 2017.

[28] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan and W. Hu, Distractor-aware Siamese networks for visual object tracking, *Proc. of the European Conference on Computer Vision (ECCV)*, pp.101-117, 2018.

[29] J. Redmon and A. Farhadi, YOLOv3: An incremental improvement, *arXiv Preprint, arXiv:1804.02767*, 2018.

[30] P. Zhu, L. Wen, X. Bian, H. Ling and Q. Hu, Vision meets drones: A challenge, *arXiv Preprint, arXiv:1804.07437*, 2018.

[31] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, Focal loss for dense object detection, *Proc. of the IEEE International Conference on Computer Vision*, pp.2980-2988, 2017.