VDTNet: A High-Performance Visual Network for Detecting and Tracking of Intruding Drones

Xunkuai Zhou, *Graduate Student Member, IEEE*, Guidong Yang, *Graduate Student Member, IEEE*, Yizhou Chen, Li Li, *Member, IEEE*, and Ben M. Chen, *Fellow, IEEE*

***Abstract*— The misuse of drones can jeopardize public safety and privacy. The detection and catching of intruding drones are crucial and urgent issues to be investigated. This work proposes VDTNet, an accurate, lightweight, and fast network for visually detecting and tracking intruding drones. We first incorporate an SPP module into the first head of YOLOv4 to enhance detection accuracy. Model compression is utilized to shrink the model size and concurrently speed up inference. We then propose and insert an SPPS module and a ResNeck module into the neck, and introduce an effective attention module for the backbone to compensate for the accuracy drop brought on by compression. With the above strategies, we present the accurate and compact VDTNet with a model size of merely 3.9 MB, ensuring low computational cost and fast detection and tracking performance in real time. Extensive experiments on four challenging public datasets show that our proposed network outperforms state-of-the-art approaches. In real-world scenarios, the comparative ground-to-air detection testing proves the generalization ability of the VDTNet, and we further demonstrate the portability and practicability of the network by deploying it on drone onboard edge-computing devices for air-to-air real-time detection of the intruding drones.**

***Index Terms*— Drone detection, tracking, lightweight, deep learning.**

I. Introduction

DRONES, or unmanned aerial vehicles (UAVs), have been widely adopted in various applications due to their maneuverability and portability [1], [2] and hence have garnered significant commercial interest and prominent application attention [3]. Critical infrastructures such as airports, seaports, military sites, and prisons face grave security threats due to the misuse of drone technologies. Consequently, it is a pressing issue to create effective countering drone systems [4], [5], [6], including detecting, tracking, and capturing intruding drones. Among them, detection and tracking are the primary problems to solve. Attributed to the great success of deep learning-based pioneers for object detection, such as CJMODT [7], SSD [8], Faster R-CNN [9], CONVLSTM [10] and YOLOv4 [11], detecting and tracking system comprising a host drone with an onboard camera to autonomously detect intruding drones based on visual cues have been proven to be a feasible solution [12].

Based on the significant contributions of their predecessors, the successors have made substantial efforts to further improve detection accuracy. Zhai et al. [13] add multi-scale prediction to enhance the detection ability of small objects. Zhu et al. [14] propose an enhanced YOLOv5 network by incorporating the self-attention module and convolutional block attention module (CBAM) [15] into its prediction head to improve the predicting ability. The networks of the YOLO family have been improved for model size reduction by redesigning the backbone and neck of the detector [16], [17], [18]. Chen et al. [19] present the DenseLightNet by adapting DenseNet to reduce the model size and speed up the inference. Although these networks possess powerful representation capability and high inference speed, they are computationally expensive and memory-consuming, thus hard to be deployed on drones equipped with limited computational resources. For instance, YOLOv4 contains more than 60 million parameters, necessitating 60 billion floating-point operations (BFLOPs) when the input image resolution is just 416×416, and the memory footprint reaches up to 256 MB. Despite the great efforts to reduce model size, DenseLightNet still takes up a 50 MB memory footprint.

To deploy the drone detection networks on edge-computing devices, Sun et al. [20] present a drone detection network with a tiny iterative backbone (TIBNet) that can reduce computational burden while compressing the network. However, TIBNet can not perform real-time drone detecting and tracking. Jiang et al. [21] propose a lightweight drone detection network (DTD-YOLOv4-Tiny) by first optimizing the anchor box generation with the K-means++ clustering and then recasting the backbone and head of the network to increase the inference speed and reduce the model size. However, the detection accuracy is not satisfactory. In conclusion, existing drone detection networks cannot satisfactorily balance detection accuracy, inference speed, and model size. In the spirit of tackling the challenges of [20] and [21] and other prior work for real-time applications while keeping high accuracy

Manuscript received 10 January 2023; revised 10 November 2023; accepted 3 January 2024. Date of publication 22 January 2024; date of current version 1 August 2024. The work of Xunkuai Zhou and Ben M. Chen was supported in part by the Research Grants Council of Hong Kong, SAR, under Grant 14217922. The Associate Editor for this article was S. S. Nedevschi. *(Corresponding author: Li Li.)*

Xunkuai Zhou and Li Li are with the School of Electronics and Information Engineering, Tongji University, Jiading, Shanghai 201804, China (e-mail: 2010474@tongji.edu.cn; lili@tongji.edu.cn).

Guidong Yang, Yizhou Chen, and Ben M. Chen are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, China (e-mail: gdyang@mae.cuhk.edu.hk; josephchen@link.cuhk.edu.hk; bmchen@cuhk.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TITS.2024.3350920, provided by the authors.

Digital Object Identifier 10.1109/TITS.2024.3350920

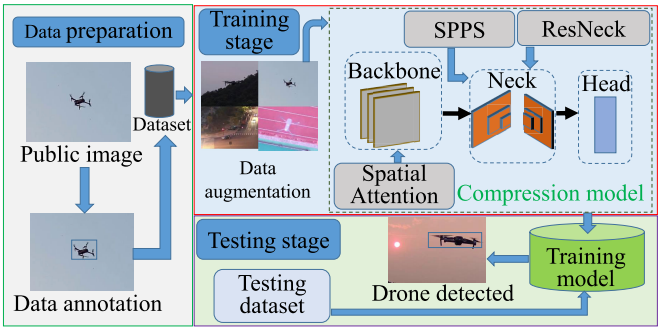


Fig. 1. The development pipeline of the proposed VDTNet. The proposed VDTNet comprises the backbone, neck, and head, where the compression technology, a new SPP module, and a new neck module are adopted and proposed to improve the model performance. We train and test the VDTNet on public datasets, and data augmentation is used to prevent overfitting.

and small size, we design a novel network for visually detecting and tracking intruding drones, namely VDTNet. The development workflow of VDTNet is detaily depicted in Fig. 1. To boost detection accuracy, we first integrate a spatial pyramid pooling (SPP) module [22] into the first head of YOLOv4 (termed as YOLOv4-SPP) and then compress YOLOv4-SPP to decrease its model size (named as YOLOv4-SPP-s). While the SPP module improves model accuracy, it tends to degenerate the inference speed. Model compression can also lead to a loss of accuracy.

Through structural analysis of the SPPF [23], we observe that utilizing the same pooling size and the concatenation operation can enhance the inference speed of the network. Inspired by SPPF, we adopt a unified pooling size of 7×7 in SPP to increase the perception field. However, unlike SPPF [23], we simplify the module complexity by excluding the concatenation operation. YOLOv3-tiny-prn [24] has demonstrated that using a residual architecture for feature fusion can improve inference speed while maintaining accuracy. However, we have found that most feature fusion structures employ convolutional operations for downsampling, resulting in feature loss. Preserving complete feature propagation would benefit accuracy enhancement. Inspired by this, we adopt residual connections in the feature fusion module. However, unlike YOLOv3-tiny-prn [24], our approach does not involve connecting with lower-level semantics; instead, we focus on fusing features between higher-level layers to save computational resources and employ a greater number of residual connections. Additionally, we utilize feature pixel recombination instead of convolutional operations for downsampling to maintain complete feature transmission, thereby improving model accuracy. Incorporating attention mechanisms is a strategy to enhance model accuracy while minimizing the increase in model size. However, we have noticed that the extensive utilization of attention modules can lead to a reduction in model inference speed. To strike a balance between accuracy and speed, we opt to incorporate only a single attention module.

Based on the analysis above, we then recast the neck and backbone of YOLOv4-SPP-s by proposing an SPP-Seven (SPPS) module, a ResNeck with an SPPS, and introducing an effective attention module to address the accuracy degradation brought on by the compression. We employ the mosaic technique to prevent overfitting in the data training stage. With the above strategies, we present an accurate, lightweight, and fast VDTNet. Our network successfully achieves an excellent trade-off between the accuracy and inference speed, between accuracy and computational consumption respectively. Extensive experiments on four public datasets [2], [12], [20], [25] for drone detection and tracking show that our VDTNet outperforms state-of-the-art (SOTA) approaches in terms of accuracy, inference speed, and model size. In real-world scenarios, the comparative ground-to-air detection testing proves the generalization ability of the VDTNet, and we further deploy the network on drone onboard edge-computing devices for air-to-air real-time detecting and tracking of the intruding drones to demonstrate its portability. To summarize, the contributions of our work are:

- We present a lightweight, effective, and efficient network for visual detection and tracking of intruding drones. Our approach utilizes computational resources efficiently, avoiding the need for expensive computational costs.
- We design the ResNeck and SPPS modules to improve both accuracy and inference latency. Furthermore, we incorporate a spatial attention module to emphasize key feature information, reduce redundancy, and suppress noise. This results in an enhancement in detection accuracy with minimal additional memory overhead.
- We conduct systematic ablation studies to validate the efficacy of the designed modules. Through extensive comparative experiments on public datasets and on-site testing, our method has demonstrated superior performance compared to state-of-the-art approaches. The successful deployment of our method on an edge computing device showcases its portability and feasibility for real-world applications.

The remainder of this paper is organized as follows. Related works are discussed in Section II. Section III presents the designed network, with detailed modular structures. The benchmarking experimental results of improvements and ablation studies are presented and discussed in Section IV. The comparative testing results in real-world scenarios are shown in Section V. Finally, we draw concluding remarks in Section VI.

II. Related Works

A. Drone Detection

Numerous studies have been conducted regarding drone detection and tracking, considering the perspectives involving non-visual and visual perception. Support vector machines are employed for the detection and classification of drones by anwar et al. [26] that can recognize drone acoustic fingerprints. However, when challenged with similar acoustic frequencies, the sound recognition accuracy droplets. Radiofrequency sensors and ubiquitous frequency-modulated continuous wave (FMCW) radar are also used to identify intruding drones for security purposes [27], [28], [29]. However, these techniques are useless for locating tiny drones that lack signal transmitters or have weak reflected signals. Conversely, visual cues may provide solutions to overcome these restrictions. Ashraf et al. [30] propose a visually-based drone detection

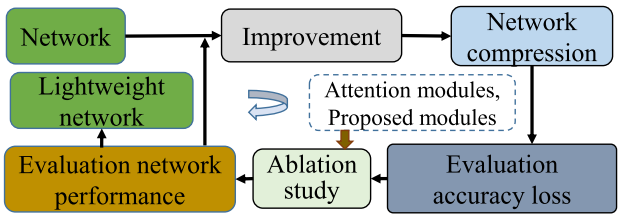


Fig. 2. The framework of the developing lightweight network. The curly and gray arrow denotes circulation of these steps.

system with high accuracy, albeit with limited inference speed (i.e., 1fps on NVIDIA RTX A6000). In their subsequent work, they introduce Transvisdrone [31] further to enhance the accuracy and speed of drone detection. However, Transvisdrone's model size of 917M presents a storage burden for devices with low memory capacity. Additionally, Singha et al. [32] present an automated drone detection approach based on YOLOv4 that increases detection speed (20.5fps). Similarly, Zhai et al. [13] enhance the detection capability of small drones using YOLOv3 with a frame rate of 21fps. Nevertheless, YOLOv3 still has a large model size of 246 MB.

B. Lightweight Model for Drone Detection

The YOLO family has been enhanced by recasting the backbone and neck to reduce the model size. Chen et al. propose YOLOv4-MCA [33] by using MobileViT [34] as the backbone and improving PANet to obtain multi-scale attention for enhancing the detection accuracy. DTD-YOLOv4-Tiny [21] is proposed by recasting the backbone and head of the network to improve the inference speed and reduce the model size. Lv et al. [35] use YOLOV5s as the baseline, and the spatial attention mechanism is introduced to reduce the number of network parameters while improving feature extraction ability. However, the accuracy is unsatisfactory, although these model sizes have been reduced [21], [33], [35].

In conclusion, existing drone detection and monitoring methods can not satisfactorily balance accuracy, latency, model size, and computational cost. Such as SAG-YOLOv5s [35], although the model size is reduced to 15 MB, when the pixels of 96 $\times$ 96 as input, the speed is only 15 fps on Nvidia GeForce RTX 2070 SUPER. This approach would not be practical for deploying on edge-computing devices.

III. Methodology

Our general framework for developing an accurate and lightweight network is shown in Fig. 2. The framework mainly includes five modules. We adopt network compression to shrink the original network model size, and before this, the improvement stage aims to enhance the accuracy of the original network, dwindling the accuracy gap between the network before and after the compression. The specific accuracy gap is evaluated in the stage of evaluating accuracy loss. The ablation study then compensates for the accuracy degradation by integrating several effective modules into the compressed network. The benefits of the introduced modules are evaluated in the stage for assessing network performance. Ultimately, an accurate and lightweight network is obtained by iteratively executing the above steps.

With YOLOv4 [11] as a baseline network, we integrate an SPP module into the first head of YOLOv4 (termed as YOLOv4-SPP) to improve the accuracy and then compress the YOLOv4-SPP to reduce model size while enhancing the inference speed. Following the compression workflow, the VDTNet is presented in Fig. 3, where we propose and integrate an SPPS module (marked with a light grey block) and a ResNeck module (marked with a light blue block) into the neck. In the backbone, we introduce the spatial attention module (SAM) in the first ResBlock to form SAM-RES. These blocks annotated with the same number (such as 1:87) are interconnected. These same numbers can be found in Fig. 3 and Fig. 6. The details of the adopted compression technique, and the proposed modular structures in neck and backbone are illustrated in Subsection III-A, Subsection III-B, and Subsection III-C, respectively.

A. Network Compression

As illustrated in Fig. 4, the representative workflow of the model compression mainly comprises three iterative steps: (i) Sparsity training, (ii) Removing channels with small-scale factors, and (iii) Evaluating network performance. Sparsity training is crucial to prevent dramatic degradation of accuracy [36] in each iteration. Removing channels with small-scale factors is a pivot procedure for reducing the number of network parameters. Evaluating network performance assists in selecting a desired lightweight network that is suitable for deployment.

In this study, we mainly remove the channels with small-scale factors in the Batch Normalization (BN) layer [37], which is formulated as follows:

$$\hat{y} = \frac{x_{in}-\mu}{\sqrt{\sigma^2+\epsilon}}; \quad y_{out} = \alpha\hat{y} + \gamma, \tag{1}$$

where μ and σ^2 are the mean and variance of input activation, respectively. α and γ represent the learnable scale and shift parameters, respectively, x_{in} denotes input features in a mini-batch, and the α decides the importance of the channel. $\hat{y}$ is the normalized activation, and y_{out} is the output of the BN layer. ϵ is a neglecting constant value added to the variance σ^2 for numeric stability.

We adopt the following objective function to discriminate the importance of the channel.

$$g(\alpha) = Loss(\alpha) + \eta \sum_{\alpha\in\Gamma} ||\alpha||_1, \tag{2}$$

where α is the learnable scale factor in the BN layer, determining the importance of the channel. Γ denotes the set of all α, $||\cdot||_1$ represents the L_1-$norm$, $Loss(\alpha)$ is the loss function of the network to be compressed, and η denotes penalty coefficient balancing the two terms in equation (2), we set $\eta = 0.001$ in this study.

After sparsity training, all α values in the BN layers of the network are stable. We then set the $\hat{\alpha}$ as the removal ratio of the channels in the BN layers. Specifically, $\alpha \times 100\%$ number of channels in all BN layers of the network will be removed. To prevent the removing process from destroying

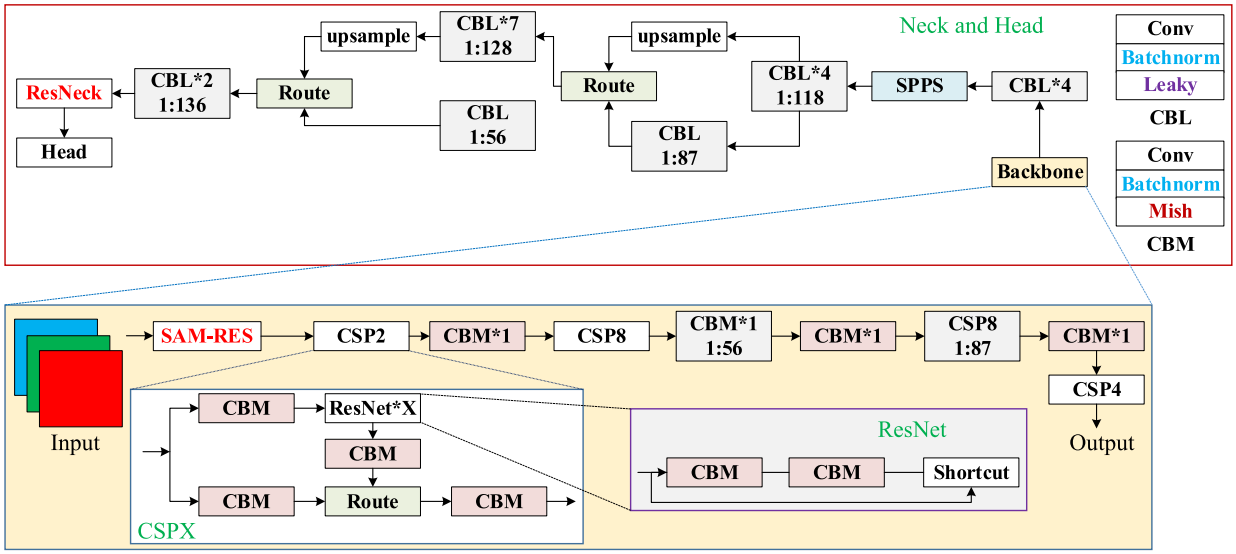


Fig. 3. The overall architecture of our VDTNet. The SPPS and ResNeck are proposed and integrated into the neck, while SAM is introduced in the ResNet of the backbone to form SAM-RES. The yellow block denotes a cross-stage partial network (CSPNet), and CSPX denotes that X CSPNet is concatenated serially. The gray block represents ResNet, and ResNetX indicates that X ResNet is concatenated serially. The convolutional layer, batch normalization layer, and mish activation function layer are collectively referred to as CBM. The '1:56' denotes the block is on the 56th layer of VDTNet. Please zoom in for the details.

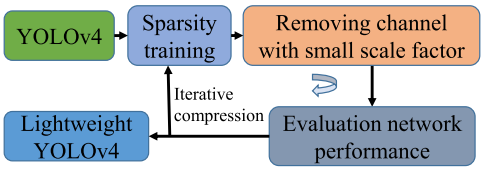


Fig. 4. The workflow of network compression, including three iterative steps. The curly and gray arrow denotes the iteration of these steps, and the compression process terminates when a desired lightweight model is obtained.

the consistency of the unabridged network, we set β as the retaining proportion of all the α values in a specific convolutional layer. In this study, β is 0.01. The procedure of model compression is performed iteratively. At each iteration, the model without dramatic accuracy degradation will be compressed continuously in the next iteration, until a desired lightweight model is acquired.

B. SPPS and ResNeck Modules in Neck

In this section, we first describe the detailed structure of SPPS and ResNeck, and then comprehensively introduce the feature map-extracting procedure in SPPS and ResNeck.

1) SPPS: Through a comprehensive structural analysis of the SPPF proposed in YOLOv5 [23], we have made an intriguing observation regarding the potential benefits of employing a consistent pooling size and concatenation operation for enhancing the network's inference speed. Drawing inspiration from the SPPF (i.e. Fig. 5(c)), we have incorporated a unified pooling size of 7×7 within the SPP module, thereby effectively expanding the perception field. It is worth noting, however, that in contrast to the SPPF [23], we opt to simplify the module structure complexity by excluding the concatenation operation.

The SPP can enhance accuracy but decrease inference speed (i.e. Fig. 5(a)), whereas the SPPF can accelerate the inference speed [23]. By leveraging the distinctive characteristics of both SPP and SPPF, we propose a novel module called SPPS. The SPPS model aims to strike a balance between accuracy and inference speed by combining the advantages of SPP and SPPF. The detailed structure of our SPPS module is shown in Fig. 5(b), where the semantic feature map from the previous layer is processed by four parallel branches, including three 7×7 MaxPool operations along a spatial dimension and one identity shortcut. After the above operations, the feature maps from four parallel branches are concatenated along the channel dimension to get the output feature map. In this study, the proposed SPPS module is integrated into the neck of our VDTNet to accelerate the inference speed and improve accuracy.

Specifically, our SPPS module defines the mapping between the input feature map $x_{in} \in \mathbb{R}^{C \times H \times W}$ and output feature map $y_{out} \in \mathbb{R}^{4C \times H \times W}$ as follows:

$$y_{out} = Concate[MaxPool(x_{in}), x_{in}, MaxPool(x_{in}), MaxPool(x_{in})], \quad (3)$$

where $Concate$ represents the concatenation operation along the channel dimension

2) ResNeck: YOLOv3-tiny-prn [24] has demonstrated the effectiveness of using a residual architecture for feature fusion, which improves inference speed while maintaining accuracy. However, most existing feature fusion structures rely on convolutional operations for downsampling, leading to feature loss. To address this issue and further enhance accuracy, we propose a different approach in our work. Inspired by the idea of residual connections, we incorporate them into the feature fusion module. Unlike YOLOv3-tiny-prn [24], our approach focuses on fusing features between higher-level layers to save computational resources and employs a larger number of residual connections. Moreover, instead of using

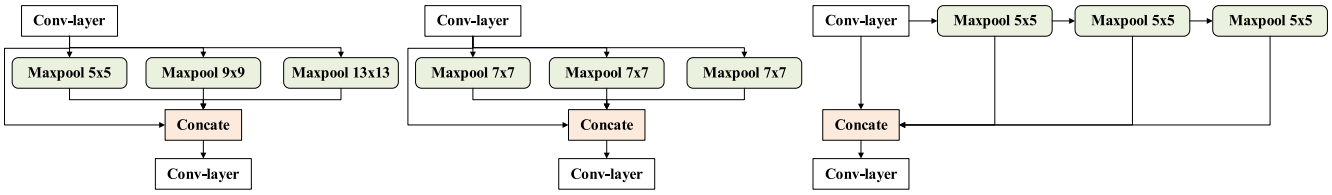


Fig. 5. The structure of the proposed SPPS module, where the semantic feature map from the previous layer is processed by four parallel branches, including three 7×7 MaxPool operations and one identity shortcut. Please zoom in for the best view.

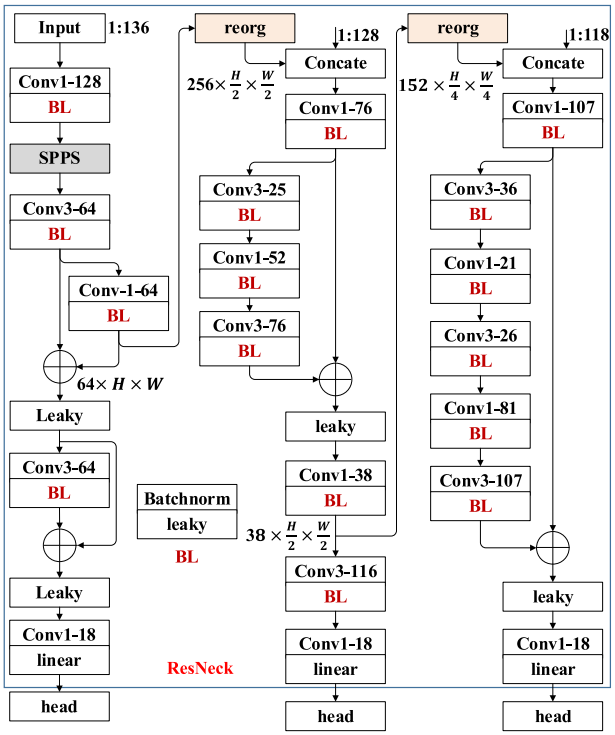


Fig. 6. The structure of the proposed ResNeck, which includes three necks, and we use the reorg layer to downsample, BL is a combination for batch normalization layer, followed by a leaky activation layer. linear also denotes the activation layers. Please zoom in for the best view.

convolutional operations for downsampling, we utilize feature pixel recombination, ensuring complete feature transmission and improving model accuracy.

As shown in Fig. 6, based on the aforementioned analysis, we design a ResNeck module to improve the accuracy and speed. ResNeck includes three neck structures where we adopt the reorg layer [38] (marked with a yellow block) instead of the convolution layer to downsample and enlarge the receptive field, retaining the semantic information of the previous convolutional layers. To transfer information to each succeeding layer, we utilize the residual structure on each neck. From left to right, the three necks are termed as neck-1, neck-2, and neck-3, respectively. For simplicity, in the following, we denote that CBL is a combination of a convolutional layer, batch normalization layer, and leak activation layer. CL is the combination of the convolutional layer and the leak activation layer.

In the neck-1 on the left, the input feature map $F_1 \in \mathbb{R}^{C_1 \times H \times W}$ is obtained from the 136th convolutional layer of the VDTNet (see Fig. 3). Afterwards, the enhanced features $F_{1a} \in \mathbb{R}^{C_2 \times H \times W}$ pay attention to additional feature information through the CBL layer, SPPS layer, and CBL layer successively. Next, the weight $\omega_1 \in \mathbb{R}^{C_2 \times H \times W}$ can be learned from F_{1a} through a residual structure, and the output feature is marked by $F_{1b} \in \mathbb{R}^{C_2 \times H \times W}$. Subsequently, by concatenating a residual structure again, the weight $\omega_2 \in \mathbb{R}^{C_2 \times H \times W}$ can be learned from F_{1b}, we label the output as $F_{1c} \in \mathbb{R}^{C_2 \times H \times W}$. The final output of the neck-1 $F_{1d} \in \mathbb{R}^{C_3 \times H \times W}$ can be obtained from a CL layer, which can be expressed by:

$$\begin{aligned} F_{1a} &= CBL(SPPS(CBL(F_1))), \\ F_{1b} &= leak(F_{1a} + \omega_1 * F_{1a}), \\ F_{1c} &= leak(F_{1b} + \omega_2 * F_{1b}), \\ F_{1d} &= CL(F_{1c}), \end{aligned} \tag{4}$$

where $*$ represents the channel-wise multiplication, leak denotes the leak activation layer.

After the downsampling operation with the reorg layer, the second neck in the middle obtains the input of $F_{12} \in \mathbb{R}^{4C_2 \times \frac{H}{2} \times \frac{W}{2}}$, we concatenate this features and the feature from 128th convolutional layer of the VDTNet (see Fig. 3) to be a fusion feature of $F_{2f} \in \mathbb{R}^{C_4 \times \frac{H}{2} \times \frac{W}{2}}$. Afterward, the enhanced feature $F_{2a} \in \mathbb{R}^{C_5 \times \frac{H}{2} \times \frac{W}{2}}$ can be obtained by the CBL layer. Subsequently, the weight $\omega_3 \in \mathbb{R}^{C_5 \times \frac{H}{2} \times \frac{W}{2}}$ can be learned from F_{2a}, and the output $F_{2b} \in \mathbb{R}^{C_6 \times \frac{H}{2} \times \frac{W}{2}}$ is obtained through the residual structure. The final output F_{2c} of the neck-2 is obtained with the three convolutional operations, which can be expressed by:

$$\begin{aligned} F_{2a} &= CBL(F_{2f}), \\ F_{2b} &= leak(F_{2a} + \omega_3 * F_{2a}), \\ F_{2c} &= CL(CBL(CBL(F_{2b}))). \end{aligned} \tag{5}$$

Similarly, following the steps from neck-2, we concatenate the features obtained by the operation of the reorg layer and the 118th convolutional layer, referred to as $F_{3f} \in \mathbb{R}^{C_7 \times \frac{H}{4} \times \frac{W}{4}}$. and then the enhanced feature $F_{3a} \in \mathbb{R}^{C_8 \times \frac{H}{4} \times \frac{W}{4}}$ is obtained by the CBL layer. Subsequently, the weight $\omega_4 \in \mathbb{R}^{C_8 \times \frac{H}{4} \times \frac{W}{4}}$ can be learned from F_{3a}, and the final output of the neck-3 $F_{3b} \in \mathbb{R}^{C_9 \times \frac{H}{4} \times \frac{W}{4}}$ is obtained through the CL layer, which can be expressed by:

$$\begin{aligned} F_{3a} &= CBL(F_{3f}), \\ F_{3b} &= CL(leak(F_{3a} + \omega_4 * F_{3a})). \end{aligned} \tag{6}$$

With the multiple necks fusing feature collaboratively, VDTNet can detect and track drones with different sizes, and inference time can also be accelerated by using the simpler downsampling operation than convolution.

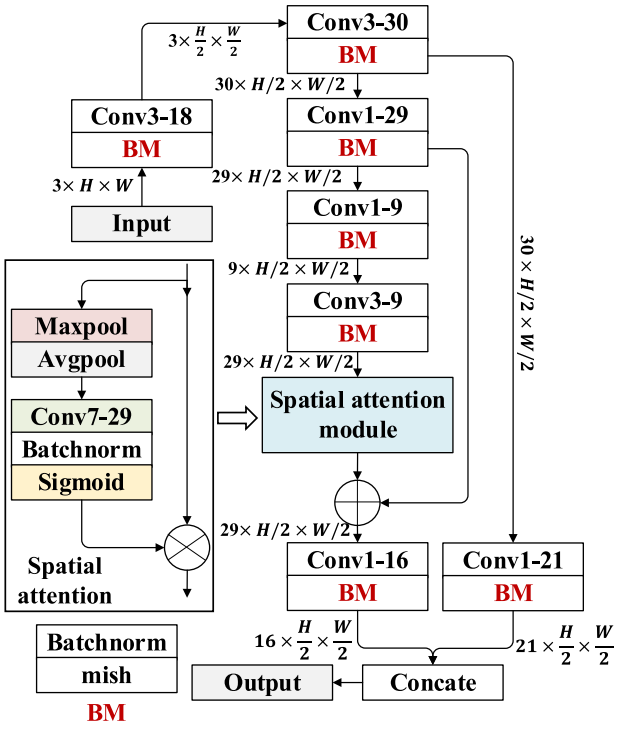


Fig. 7. The spatial attention integrated into the first ResBlock of VDTNet. BM is an combination for batch normalization layer, followed by a mish activation layer. Please zoom in for the best view.

C. Spatial Attention Module in Backbone

In most cases, the accuracy of small-drone detection is inferior to medium-drone or large-drone detection, partly due to the limited feature information extracted by the network. The attention mechanism can focus on the crucial information of the feature map, only with small extra parameters. Inspired by the feature aggregation in the SAM [39], we experimentally compare the detection accuracy when integrating the SAM into different locations of the ResBlock in VDTNet. The best detection accuracy can be achieved when integrating the SAM into the first ResBlock (SAM-RES), as shown in Fig. 7, where 'Conv$n$-k' represents the k convolutional kernels with the size of $n \times n$, 'Batchnorm' is the batch normalization layer, 'Sigmoid' and 'Mish' are activation functions, and 'Concate' means the concatenation operation concatenating the feature maps along the channel dimension.

To illustrate the detailed operations of the SAM, we denote the input feature map as $x_{in} \in \mathbb{R}^{C\times H\times W}$, where C, H, W represents the channel, height, and width number, respectively. The SAM will output the enhanced feature map $y_{out} \in \mathbb{R}^{C\times H\times W}$. The relationship between y_{out} and x_{in} can be formulated as follows:

$$\begin{aligned} y_{out} &= SAM(x_{in}) \\ &= \sigma(Conv^{7\times7}([AvgPool(x_{in}); MaxPool(x_{in})]) \otimes x_{in}, \\ &= \sigma(Conv^{7\times7}([Map_{avgp}; Map_{maxp}]) \otimes x_{in}, \end{aligned} \tag{7}$$

where $\otimes$ denotes element-wise multiplication, σ denotes the sigmoid activation function, and $Conv^{7\times7}$ represents a convolutional operation with the filter size of 7×7.

We use two pooling operations to obtain channel information of a feature map, and then generate two 2D maps: M_{avgp} and M_{maxp}, each denotes the average-pooled features and max-pooled features across the channel. The information features of two 2D maps are combined to be an entirety feature (EF), convolved EF by using a standard convolutional layer to get convolutional information (CF), and multiply CF with the previous convolutional layer to produce a 2D spatial attention map.

IV. Experiments

A. Datasets and Evaluation Metrics

1) Dataset: Due to the lack of publicly available large-scale datasets for drone detection, this study utilizes the Det-fly [12], TIB-Net [20], FL-Drone [25], and DUT [2] for training, validation, and testing, respectively.

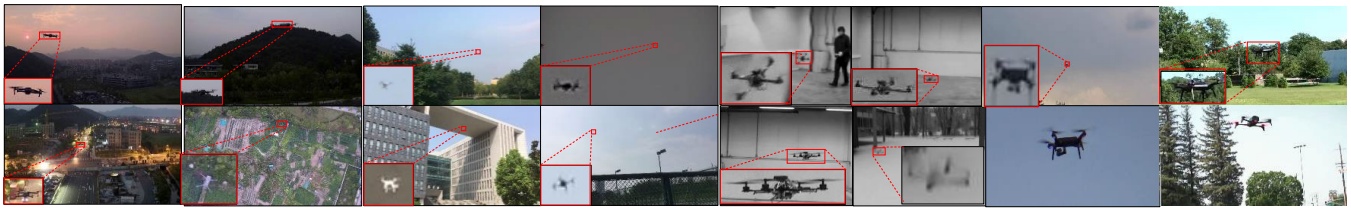
Det-Fly [12] and FL-Drone [25] (air-to-air drone detection) are captured by a flying drone. TIB-Net [20] is gathered by a fixed camera on the ground (ground-to-air drone detection), as shown in Fig. 8. Compared with Det-Fly, TIB-Net has a more complex environment and contrast of foreground-background, and most of the object size is smaller than Det-Fly, while its totality is smaller than Det-Fly. FL-Drone contains a mix of indoor and outdoor scenes. In our study, we extract the frames annotated in Dogfight [30] as images for training and testing. DUT [2] dataset comprises images with varying resolutions, this substantial difference in image sizes introduces challenges during training, as the image data undergoes compression and stretching, resulting in varying degrees of deformation for the target objects. Consequently, the training data may exhibit deviations from the actual shapes of drones encountered in real-life scenarios, posing a significant challenge for achieving high-accuracy detection in practical applications.

In this study, we train our network and benchmark the testing results on Det-Fly, TIB-Net, FL-Drone, and DUT, respectively, by following the same data partition rule in [2], [12], [20], and [30]. The annotation labels are unified into the YOLO format for easy training in advance.

2) Evaluation Metrics: To evaluate and compare the performance of the proposed network with SOTA approaches, we set the IoU threshold between predictions and ground truth to be 0.5. Therefore, detections matching with ground truth with IoU$\geq$0.5 are counted as the true positives; this study focuses on four key aspects: (i) drone detection accuracy, (ii) inference speed (measured in ms), (iii) model size (measured in MB). The mean Average Precision (mAP) is adopted for accuracy evaluation [40] on Det-Fly. For the TIB-Net and FL-Drone, we follow their evaluation rule in [20] and [30]. The inference speed, model size, and BFLOPs are critical criteria to evaluate if deploying the network on edge-computing devices is feasible for real-time drone detection and tracking.

B. Implementation Details

1) Training: We train and evaluate our VDTNet on the Det-Fly, TIB-Net, and FL-Drone, respectively. During training, we set the number of overall training iterations to 40k, 200k, 40k, and 60k on the Det-Fly [12], TIB-Net [20], FL-Drone [25], and DUT [2], respectively, and choose the



(a) Det-Fly (b) TIB-Net (c) FL-Drone (d) DUT

Fig. 8. Partial images of Dataset. TIB-Net is more complicated than Det-Fly. FL-Drone is RGB images collected in indoor and outdoor scenarios. DUT data includes 35 types of drone images. Please zoom in for the best view.

stochastic gradient descent optimizer (SGD) with an initial learning rate $\eta_{initial} = 1.3 \times 10^{-3}$. At 80% and 90% of the setting iterations, the learning rate drops to ten times the previous learning rate. The weight decay is 0.0005, and the momentum is 0.949, respectively. In Det-Fly and TIB-Net, the batch size and the mini-batch are 64 and 32, respectively; in FL-Drone, the batch size and the mini-batch are 64 and 8, respectively. In DUT, the batch size and the mini-batch are 64 and 16, respectively. All experiments are trained with an RTX 3090 GPU.

2) Evaluation: In Det-Fly [12], we test the inference time and BFLOPS on RTX 3090 GPU. Following the original paper [20], [30], in TIB-Net [20] and FL-Drone [30], we utilize NVIDIA TITAN Xp and 1080Ti to test inference time, respectively. For the DUT dataset, the inference time is tested with RTX 2080super GPU devices; since we are a lake of RTX 2080super GPU, we just use a low-performance device of RTX 1080Ti to test the inference time of VDTNet. In each testing experiment, for testing inference time and BFLOPS. We input the same original resolution image from the testing set in three datasets [12], [20], [30], respectively, and the accuracy (mAP) is gauged on the testing set.

C. Improving and Compressing Network

As shown in Fig. 9, we improve the YOLOv4 by integrating SPP into its first head (termed as YOLOv4-SPP) and then compress the YOLOv4-SPP iteratively by following steps in Section III-A. We compress YOLOv4-SPP in three iterations, and set the compression rate from 0.1-0.9 in each iteration, and select the model with little loss of accuracy to compress in the next iteration. Finally, the experimental results from the chosen model in each iteration are shown in Table I.

With 608×608 input image resolution to train. the mAP of YOLOv4-SPP is improved by 0.3% w.r.t. original YOLOv4. After three iterations of compression are complete, model C with the model size of 3.5 MB is obtained, and the model size has been reduced by about 98.6%, while its accuracy barely drops about 2.5%, and the inference accelerates about 37%.

D. Ablation Studies on Proposed Modules

To compensate for the accuracy loss of model C, we propose ResNeck and SPPS, and we introduce SAM. The effectiveness of each optimization method is verified by adding modules step by step, with input images at a resolution of 1024×1024 from Det-Fly [12] to train. Eventually, we present these results in Table II.

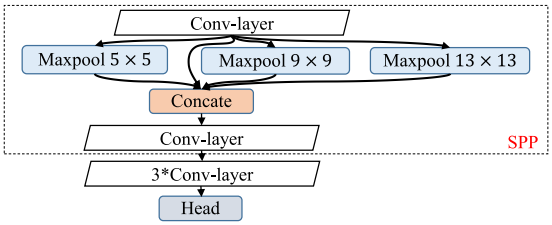


Fig. 9. SPP is positioned between the penultimate forth and fifth convolution layers of the first head. Please zoom in for the best view.

TABLE I
COMPRESSION RESULTS BASED ON YOLOV4 ON DET-FLY

Model	Method description	mAP ↑	Latency ↓	model size ↓
YOLOv4	baseline	94.65%	12.1ms	256M
V4-SPP	integrate SPP	94.95%	13.0ms	259.6M
A	compress V4-SPP (0.4)	94.20%	11.4ms	108.5M
B	compress A (0.6)	93.91%	9.56ms	28.1M
C	compress B (0.8)	92.15%	7.6ms	3.5M

[1] ↑ (↓) indicates that larger (smaller) values lead to better (worse) performance. V4-SPP is an abbreviation of YOLOv4-SPP.
[2] '0.4', '0.6', and '0.8' in the second column denote the compression rate of models YOLOv4-SPP, A, and B, respectively.

TABLE II
EFFECTS OF VARIOUS COMPONENTS ON PERFORMANCE ON DET-FLY

Modules	From Model C to VDTNet			
ResNeck		✓	✓	✓
SAM			✓	✓
SPPS				✓
model size	3.5M	3.9M	3.9M	3.9M
Improvement	-	+0.4M	0M	0M
Latency/per image	12.7ms	11.3ms	13.5ms	13.2ms
Improvement	-	-1.4ms	+2.2ms	-0.3ms
mAP	95.10%	95.65%	95.95%	96.12%
Improvement	-	+0.55	+0.30	+0.17

[1] The evaluation metric marked with red (green) color denotes the performance degradation (improvement).
[2] The ✓denotes that the module is integrated into this ablation study.

1) Benefits of ResNeck: In Table II, since the residual structure provides more rich information, the reorg layer can transfer previous rich information to the next neck, it improves the mAP by 0.55% and inference time by 1.4ms, which is in line with our original philosophy for designing this module.

2) Benefits of SAM: The mAP increases by 0.3% when integrating SAM into the first ResBlock with 29 filters, as shown in Fig. 7. In our study, the mAP decreases when integrating SAM into the last ResBlock with 199 filters. We consider

TABLE III
QUANTITIVE BENCHMARKING RESULTS ON DET-FLY

Model	Image size	mAP ↑	Recall ↑	Precision ↑	F1-score ↑	Latency ↓	Model size ↓
Cascade R-CNN [12]	[640,640]	79.4%	-	-	-	-	552.6M
RetinaNet [12]	[600,600]	77.9%	-	-	-	-	80.0M
RefineDet [12]	[320,320]	69.5%	-	-	-	-	78.1M
FPN [12]	[600,600]	78.7%	-	-	-	-	97.7M
Faster R-CNN [12]	[1000,600]	70.5%	-	-	-	-	333.5M
Grid R-CNN [12]	[600,600]	82.4%	-	-	-	-	493.0M
SSD512 [12]	[416,416]	78.7%	-	-	-	-	96.3M
YOLO-Fastest [41]		78.5%	81.1%	73.5%	77.1%	6.0ms	**1.2M**
YOLOv3-Tiny-Prn [24]		25.1%	20.6%	81.0%	32.9%	1.8ms	19.4M
YOLO-Lite [42]		25.5%	37.0%	33.8%	35.3%	1.4ms	2.2M
YOLOv2 [38]		67.1%	69.9%	75.7%	72.7%	3.1ms	202.3M
YOLOv3 [43]	[416,416]	87.8%	89.7%	88.9%	89.3%	6.3s	246.3M
YOLOv4-Tiny [11]		26.7%	20.5%	85.8%	33.1%	2.2ms	23.5M
SlimYOLOv3-SPP3-50 [37]		86.4%	89.1%	88.2%	88.7%	6.6ms	133.5
SlimYOLOv3-SPP3-90 [37]		83.5%	85.4%	86.3%	85.9%	4.6ms	32.2M
SlimYOLOv3-SPP3-95 [37]		82.1%	84.2%	86.7%	85.4%	4.0ms	20.4M
YOLOv3-SPP [43]	[320,320]	83.4%	82.7%	85.4%	84.0%	5.1ms	250.5M
	[416,416]	87.5%	89.4%	88.5%	89.0%	6.5ms	250.5M
	[1024,1024]	89.2%	89.6%	87.2%	88.4%	24.4ms	250.5M
YOLOv3-SPP3 [37]	[416,416]	78.1%	80.2%	83.3%	81.7%	7.1ms	255.6M
	[608,608]	84.2%	85.1%	85.1%	85.1%	11.8ms	255.6M
	[1024,1024]	91.7%	90.4%	91.9%	91.1%	26.7ms	255.6M
YOLOv4 [11]	[416,416]	89.4%	90.0%	88.1%	89.0%	7.5ms	256.0M
	[640,640]	94.9%	95.5%	90.5%	92.9%	12.6ms	256.0M
	[1024,1024]	95.3%	96.1%	89.4%	92.6%	27.2ms	256.0M
YOLOv5n [23]	[416,416]	86.3%	84.3%	89.5%	86.8%	**0.3ms**	3.6M
	[640,640]	94.0%	93.0%	92.1%	92.6%	0.4ms	3.6M
	[1024,1024]	94.2%	93.5%	92.1%	92.8%	0.6ms	3.6M
YOLOv6n [44]	[416,416]	87.5%	*	*	*	0.4ms	9.3M
	[640,640]	93.0%	*	*	*	0.6ms	9.3M
	[1024,1024]	94.7%	*	*	*	1.3ms	9.3M
YOLOv7-Tiny [45]	[416,416]	85.8%	78.7%	94.5%	85.9%	4.0ms	23.0M
	[640,640]	88.8%	86.5%	88.1%	87.3%	4.6ms	23.0M
	[1024,1024]	94.1%	93.3%	94.7%	94.0%	8.0ms	23.0M
VDTNet (Ours)	[416,416]	90.3%	91.3%	89.8%	90.5%	4.1ms	3.9M
	[640,640]	94.8%	94.9%	92.1%	93.5%	6.4ms	3.9M
	[1024,1024]	96.1%	96.6%	92.4%	94.5%	13.2ms	3.9M
	[2048,1440]	**96.2%**	**96.6%**	**92.7%**	**94.6%**	32.8ms	3.9M

[1] '-' denotes no official metric report in the original paper, and '*' denotes no evaluation metrics computed by the open source codes.
[2] The best results in each evaluation metric column are in bold.

that the integration of SAM provides more critical information compensation for the low-level feature map while adding SAM to the last ResBlock with 199 filters results in noisy feature extraction. Notably, the SAM can improve accuracy with negligible extra parameters, beneficial to deploying the network on memory-limited edge-computing devices.

3) Benefits of SPSS: The SPPS module expands the semantic information with three simple MaxPool layers. the SPPS improves the mAP by 0.17%, and the inference time accelerates by 0.3ms, the result is in line with our expectations. where we only replace the two SPP with SPPS (see Fig. 3 and Fig. 6).

V. BENCHMARKING AND REAL-WORLD TESTING

In this section, we first present and analyze the benchmarking results of our VDTNet on three public challenging datasets stated in subsection V-A. We then demonstrate the portability of our network through real-world testing in subsection V-B.

A. Benchmarking and Analysis

We conduct extensive experiments to compare and benchmark the performance of our VDTNet with SOTA approaches for drone detection. The performance is evaluated in four aspects, including (i) drone detection accuracy (measured by mAP), (ii) inference speed (measured in ms), (iii) model size (measured in MB), and (iv) computational consumption (measured in BFLOPs). For fair comparison with our lightweight VDTNet, we choose YOLOv5n [23], YOLOv6n [41], and YOLOv7-Tiny [42] as the lightweight representatives of YOLOv5 [23], YOLOv6 [41], and YOLOv7 [42], respectively. We then train and test different networks on Det-Fly, TIB-Net, FL-Drone, and DUT, respectively. For experiments on Det-Fly [12], TIB-Net [20], FL-Drone [30], and DUT [2], all the models are trained on NVIDIA RTX 3090, and the first three datasets are tested on NVIDIA RTX 3090, 2080Ti, and TATIN xp for a fair comparison, the DUT dataset is tested on RTX 1080Ti. The benchmarking results are analyzed in detail as follows:

1) Benchmarking on Det-Fly Dataset: The quantitative benchmarking results on Det-Fly [12] are reported in Table III. Our VDTNet achieves SOTA performance in terms of mAP, concurrently maintaining the most compact model size (3.9 MB only). Additionally, the improvement in recall means that VDTNet has more room to enhance and the increase in F1 score means that VDTNet is more robust, and the higher precision means the correctness of detection. Notably, our VDTNet outperforms YOLOv7-Tiny by 2.0% points in terms of mAP, showing that our network has the best performance and strongest generalization ability. Besides, VDTNet infers

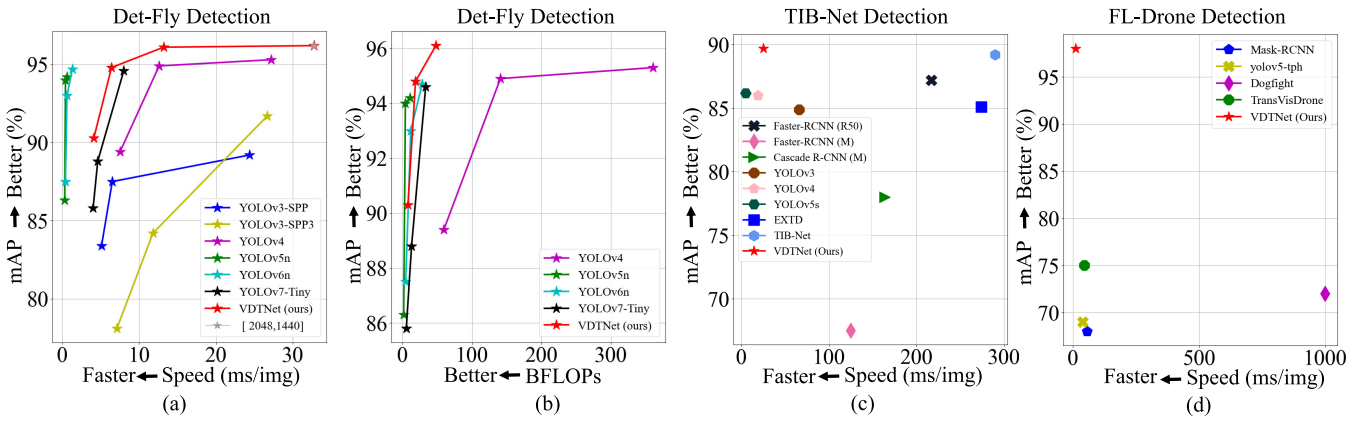


Fig. 10. (a) Inference speed versus accuracy on Det-Fly. (b) BFLOPs versus accuracy on Det-Fly. (c) Inference speed versus accuracy on TIB-Net. (d) Inference speed versus accuracy on FL-Drone. Please zoom in for the best view.

TABLE IV
QUANTITIVE BENCHMARKING RESULTS ON TIB-NET

Model	Image size	mAP ↑	Latency ↓	Model size ↓
Faster RCNN (R50) [9]		87.2%	217ms	333.5M
Faster RCNN (M) [9]		67.5%	125ms	162.5M
Cascade R-CNN (M) [43]		78.0%	164ms	384.9M
YOLOv3 [44]		84.9%	66ms	234.1M
YOLOv4 [11]	[1333,800]	86.0%	19ms	256.0M
YOLOv5s [23]		86.2 %	**5ms**	14.9M
EXTD [45]		85.1%	274ms	**696.9KB**
TIBNet [20]		89.2%	290ms	697.0KB
YOLOv4-Tiny [11]		78.5%	-	23M
DTD-YOLOv4-Tiny [21]	[960,540]	80.3%	-	1.4M
	[1344,756]	83.3%	-	1.4M
	[1920,1080]	85.1%	-	1.4M
VDTNet (Ours)	[1024,1024]	**89.7%**	25ms	3.9M

[1] '-' denotes no official metric report in the original paper. 'M' and 'R50' represent the backbone of MobileNet and ResNet50, respectively.
[2] The best results in each evaluation metric column are in bold.

twice as fast as YOLOv4 [11] can. We get a series of VDTNet results by changing the input size of the image. Fig. 10(a) shows that VDTNet results have advantages in the balance of speed and accuracy, VDTNet maintains high accuracy and speed when the inputting size is 2048×1440 at the highest point, additionally. We also demonstrate that the VDTNet has a better balancing between the requirement of computation resource and accuracy for identical inputting images in Fig. 10(b), the low point denotes the inputting size is 416×416, the middle point denotes the inputting size is 640×640, and the high point denotes the inputting size is 1024×1024. Compared with YOLOv4, even if the inputting size of 1024×1024, the BFLOPs (48.1) of VDTNet is lower than the BFLOPs (59.6) of YOLOv4 at the inputting size of 416×416, VDTNet delivers the highest detection accuracy, and less computing power required. Compared with the rest of the methods in Fig. 10(b), our VDTNet requires only cheap computing resources to achieve better accuracy, such performance is very competitive in drone applications.

2) Benchmarking on TIB-Net Dataset: The quantitative benchmarking results on TIB-Net are shown in Table IV. Our VDTNet outperforms all the approaches in terms of mAP and model size by a significant margin. For instance, when compared with TIBNet, the best-performing model on the TIB-Net, although VDTNet is larger than TIBNet [20] in model size, our VDTNet gets higher accuracy and runs faster inference speed than TIB-Net, which is because multiple spatial attention module used in TIBNet [20] that can degenerate the inference speed. We visualize the results of inference speed versus accuracy, as shown in Fig. 10(c), which demonstrates our VDTNet has already achieved a satisfactory trade-off between the accuracy and inference speed.

3) Benchmarking on FL-Drone Dataset: We train VDTNet on FL-Drone [30] with the input image resolution of 640×480, as shown in Table V, it shows the VDTNet gets a better accuracy and speed on FL-Drone, and VDTNet is 30% both more accurate and faster than TransVisDrone that is a new model for air-to-air drone detection, and the speed is almost 90 times faster than doglight. We visualize the results of inference speed versus accuracy, as shown in Fig. 10(d), which demonstrates our VDTNet has already achieved a satisfactory trade-off between the detection accuracy and inference speed.

4) Benchmarking on DUT Dataset: We train VDTNet on DUT [2] with the input image resolution of 640×640, as shown in Table VI, although we use an RTX 1080Ti that inferior to the original paper of RTX 2080super to test the latency, VDTNet still gets the better accuracy and faster speed than other SOTA methods on DUT than Cascade(ResNet50), it is faster than YOLOX that is the fastest in original paper [2] by 7ms.

5) Lightness Analysis: We arrange the values of these model sizes from smallest to largest, as shown in Fig. 11, the VDTNet (3.5 MB) is just 17% of YOLOv7-Tiny (23.0 MB) [42], and take up memory is only 1.5% of YOLOv4 (256.0 MB) [11]. While six models are lighter than VDTNet they struggle to precisely localize drones. The rest of the models presented take more memory than VDTNet, highlighting the practicality of our method.

B. Real-World Testing

1) Comparative Testing for Ground-to-Air Drone Detection: To show the superiority and generalization ability of our VDTNet, we comparatively test the VDTNet (trained on Det-Fly) through on-site experiments. The on-site environment and qualitative testing results are shown in Fig. 12, where

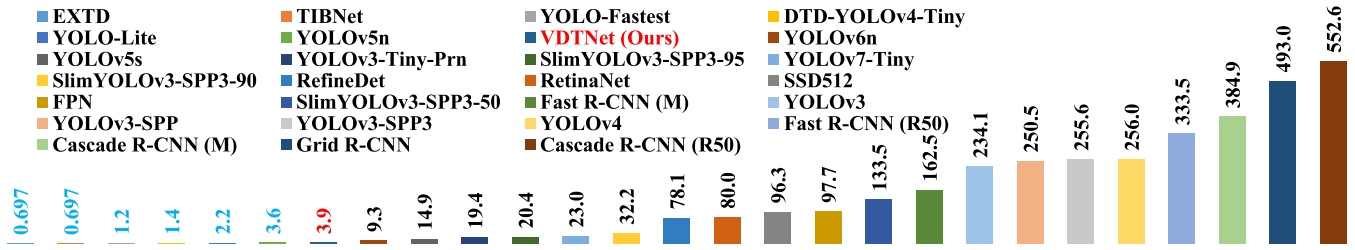


Fig. 11. Comparison model size to SOTA methods, VDTNet memory usage is barely 17% of advanced YOLOv7-Tiny. the 'M' and 'R50' denote the MobileNet and ResNet50 used as the backbone of this network. Please zoom in for the best view.

TABLE V
QUANTITIVE BENCHMARKING RESULTS ON FL-DRONE

Model	mAP↑	Latency↓
SCRDet-H [46]	52%	-
SCRDet-R [46]	52%	-
FCOS [47]	69%	-
Mask-RCNN [48]	68%	57.1ms
MEGA [49]	65%	-
SLSA [50]	61%	-
YOLOv5-tph [14]	69%	40.0
Dogfight [30]	72%	1000.0ms
TransVisDrone [31]	75%	46.3ms
Ours	**98%**	**11.6ms**

[1] '-' denotes no official metric report in the original paper.
[2] The best results in each evaluation metric column are in bold.
[3] The original paper [31] does not report the model size values.

TABLE VI
QUANTITIVE BENCHMARKING RESULTS ON DUT

Model	Backbone	mAP↑	Latency↓
Faster-RCNN [9]	ResNet50	65.3%	78.1ms
	ResNet18	60.5%	51.5ms
	VGG16	63.3%	107.5ms
Casecade-RCNN [43]	ResNet50	68.3%	93.5ms
	ResNet18	65.2%	68.0ms
	VGG16	66.7%	125.0ms
ATSS [51]	ResNet50	64.2%	75.2ms
	ResNet18	61.0%	48.8ms
	VGG16	64.1%	105.3ms
YOLOX [52]	ResNet50	42.7%	46.1ms
	ResNet18	40.0%	18.6ms
	VGG16	55.1%	43.5ms
	Darknet	55.2%	19.5ms
SSD [8]	VGG16	63.2%	30.1ms
VDTNet (Ours)	-	**68.6%**	**11.7ms**

[1] The latency marked with cyan is tested by using RTX 2080super GPU in original paper [2], and the latency of VDTNet is tested on RTX 1080Ti GPU that the performance is inferior to RTX 2080super GPU.
[2] The original paper [2] does not report the model size values.
[3] The best results in each evaluation metric column are in bold.

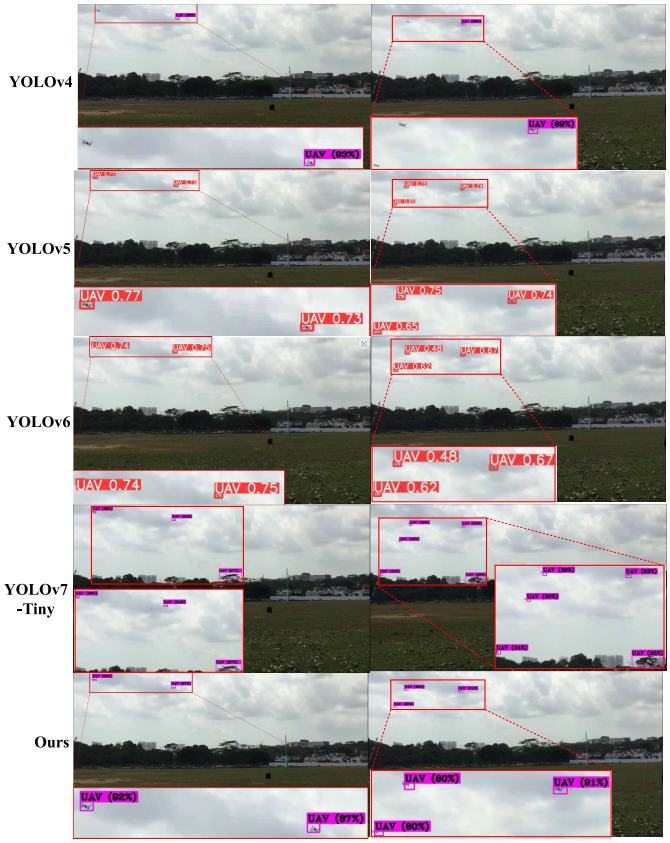


Fig. 12. Comparison with SOTA methods by practical ground-to-air detection, where the two columns represent two different scenarios comprising two and three drones to be detected, respectively. From top to bottom, each row demonstrates the testing results with the same methods, which demonstrates that VDTNet has an accurate detection performance. Please zoom in for the best view.

the two columns of Fig. 12 represent two different scenarios comprising two and three drones to be detected, respectively. From top to bottom, each row demonstrates the testing results of YOLOv4 [11], YOLOv5n [23], YOLOv6n [41], YOLOv7-Tiny [42], and VDTNet for intruding drone detection and tracking. We can find that the YOLOv4 only detects one drone in both scenarios, while the YOLOv5n, YOLOv6n, YOLOv7-Tiny, and VDTNet can detect all drones. However, YOLOv7-Tiny possesses one false positive in both two scenarios by mistakenly detecting the trees and power lines as drones, partly due to the limited feature extraction ability of its backbone. Notably, the confidence score of our VDTNet (0.86 on average) is significantly higher than YOLOv5n (0.73 on average) and YOLOv6n (0.65 on average). In conclusion, our VDTNet shows the strongest generalization ability in real-world scenarios.

2) Deployment for Air-to-Air Drone Detection: As shown in Fig. 13. To demonstrate the deployment capability of VDTNet on edge-computing devices, we deploy our VDTNet on an onboard computer, NVIDIA Jetson Xavier NX device with 7025MB GPU memory and 6 CPU cores. In our experiment, the task of our host drone is to visually detect and track the intruding drone (DJI Mavic Enterprise 200). With input frames at a resolution of 640 × 480, our 416 resolution model can accurately detect the intruding drones and achieve **14.9 fps**

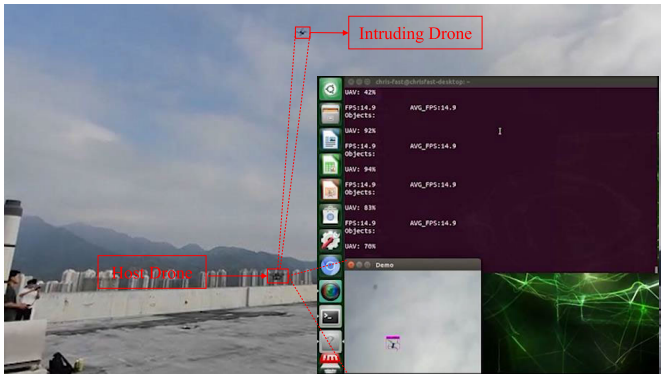


Fig. 13. Air-to-air detection of VDTNet deployed on drone with Jetson NX, the detection speed is 14.9fps, and the confidence is 78% this moment.

without any optimization, which confirms that our network satisfies the requirements for portability and practicality.

VI. Conclusion

An accurate, lightweight, and fast network for real-time drone intrusion detection and tracking is presented in this work. We discovered that adding the spatial attention module to the first ResBlock in the backbone can enhance detection accuracy. Specifically, we proposed SPPS and ResNeck to improve inference speed and accuracy. Our method significantly reduces model size and achieves state-of-the-art performance on four real-world datasets. The proposed technique demonstrates clear advantages, as evidenced by ground-to-air testing in actual settings. We have also demonstrated the feasibility of deploying our network on edge-computing systems for air-to-air detection. In the future, we aim to further enhance detection speed and improve multi-drone detection and tracking accuracy.

References

[1] J. Shen, B. Wang, B. M. Chen, R. Bu, and B. Jin, "Review on wind resistance for quadrotor UAVs: Modeling and controller design," *Unmanned Syst.*, vol. 11, no. 1, pp. 5–15, Jan. 2023.

[2] J. Zhao, J. Zhang, D. Li, and D. Wang, "Vision-based anti-UAV detection and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25323–25334, Dec. 2022.

[3] M. Lan, S. Lai, T. H. Lee, and B. M. Chen, "A survey of motion and task planning techniques for unmanned multicopter systems," *Unmanned Syst.*, vol. 9, no. 2, pp. 165–198, Apr. 2021.

[4] B. M. Chen, "On the trends of autonomous unmanned systems research," *Engineering*, vol. 12, pp. 20–23, May 2022.

[5] S. O. Ajakwe, D.-S. Kim, and J.-M. Lee, "Drone transportation system: Systematic review of security dynamics for smart mobility," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14462–14482, Aug. 2023.

[6] S. O. Ajakwe, V. U. Ihekoronye, D.-S. Kim, and J.-M. Lee, "ALIEN: Assisted learning invasive encroachment neutralization for secured drone transportation system," *Sensors*, vol. 23, no. 3, p. 1233, Jan. 2023.

[7] Z. Cao, J. Li, D. Zhang, M. Zhou, and A. Abusorrah, "A multi-object tracking algorithm with center-based feature extraction and occlusion handling," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4464–4473, Apr. 2023.

[8] W. Liu et al., "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.

[10] W. Hammedi, B. Brik, and S. M. Senouci, "Toward optimal MEC-based collision avoidance system for cooperative inland vessels: A federated deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2525–2537, Mar. 2022.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[12] Y. Zheng, Z. Chen, D. Lv, Z. Li, Z. Lan, and S. Zhao, "Air-to-air visual detection of micro-UAVs: An experimental evaluation of deep learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1020–1027, Apr. 2021.

[13] H. Zhai and Y. Zhang, "Target detection of low-altitude UAV based on improved YOLOv3 network," *J. Robot.*, vol. 2022, pp. 1–8, Mar. 2022.

[14] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 2778–2788.

[15] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.

[16] K. Feng, W. Li, J. Han, and F. Pan, "Low-latency aerial images object detection for UAV," *Unmanned Syst.*, vol. 10, no. 1, pp. 57–67, Jan. 2022.

[17] M. Derome, A. Plyer, M. Sanfourche, and G. L. Besnerais, "Moving object detection in real-time using stereo from a mobile platform," *Unmanned Syst.*, vol. 3, no. 4, pp. 253–266, Oct. 2015.

[18] J. Guo et al., "Pigeon cleaning behavior detection algorithm based on light-weight network," *Comput. Electron. Agricult.*, vol. 199, Aug. 2022, Art. no. 107032.

[19] L. Chen, Q. Ding, Q. Zou, Z. Chen, and L. Li, "DenseLightNet: A light-weight vehicle detection network for autonomous driving," *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10600–10609, Dec. 2020.

[20] H. Sun, J. Yang, J. Shen, D. Liang, L. Ning-Zhong, and H. Zhou, "TIB-Net: Drone detection network with tiny iterative backbone," *IEEE Access*, vol. 8, pp. 130697–130707, 2020.

[21] R. Jiang, Z. Ye, Y. Peng, G. Xie, and H. Du, "Lightweight target detection algorithm for small and weak drone targets," *Prog. Laser Optoelectron.*, vol. 59, no. 8, pp. 109–120, 2022.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[23] G. Jocher. (2020). *YOLOv5 by Ultralytics*. [Online]. Available: https://github.com/ultralytics/yolov5

[24] C.-Y. Wang, H. M. Liao, P.-Y. Chen, and J.-W. Hsieh, "Enriching variety of layer-wise learning information by gradient combination," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2477–2484.

[25] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 5, pp. 879–892, May 2017.

[26] M. Z. Anwar, Z. Kaleem, and A. Jamalipour, "Machine learning inspired sound-based amateur drone detection for public safety applications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2526–2534, Mar. 2019.

[27] Á. D. de Quevedo, F. I. Urzaiz, J. G. Menoyo, and A. A. López, "Drone detection with X-band ubiquitous radar," in *Proc. 19th Int. Radar Symp. (IRS)*, Jun. 2018, pp. 1–10.

[28] L. Dressel and M. J. Kochenderfer, "Hunting drones with other drones: Tracking a moving radio target," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1905–1912.

[29] S. Basak, S. Rajendran, S. Pollin, and B. Scheers, "Combined RF-based drone detection and classification," *IEEE Trans. Cognit. Commun. Netw.*, vol. 8, no. 1, pp. 111–120, Mar. 2022.

[30] M. W. Ashraf, W. Sultani, and M. Shah, "Dogfight: Detecting drones from drones videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7063–7072.

[31] T. Sangam, I. R. Dave, W. Sultani, and M. Shah, "TransVisDrone: Spatio-temporal transformer for vision-based drone-to-drone detection in aerial videos," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2023, pp. 6006–6013.

[32] S. Singha and B. Aydin, "Automated drone detection using YOLOv4," *Drones*, vol. 5, no. 3, p. 95, Sep. 2021.

[33] Q. Cheng, X. Li, B. Zhu, Y. Shi, and B. Xie, "Drone detection method based on MobileViT and CA-PANet," *Electronics*, vol. 12, no. 1, p. 223, Jan. 2023.

[34] S. Mehta and M. Rastegari, "MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer," 2021, *arXiv:2110.02178*.

[35] Y. Lv, Z. Ai, M. Chen, X. Gong, Y. Wang, and Z. Lu, "High-resolution drone detection based on background difference and SAG-YOLOv5s," *Sensors*, vol. 22, no. 15, p. 5825, Aug. 2022.
[36] H. Liu, K. Fan, Q. Ouyang, and N. Li, "Real-time small drones detection based on pruned YOLOv4," *Sensors*, vol. 21, no. 10, p. 3374, May 2021.
[37] P. Zhang, Y. Zhong, and X. Li, "SlimYOLOv3: Narrower, faster and better for real-time UAV applications," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 37–45.
[38] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.
[39] X. Zhu, D. Cheng, Z. Zhang, S. Lin, and J. Dai, "An empirical study of spatial attention mechanisms in deep networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6687–6696.
[40] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
[41] C. Li et al., "YOLOv6: A single-stage object detection framework for industrial applications," 2022, *arXiv:2209.02976*.
[42] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, *arXiv:2207.02696*.
[43] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into high quality object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6154–6162.
[44] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[45] Y. Yoo, D. Han, and S. Yun, "EXTD: Extremely tiny face detector via iterative filter reuse," 2019, *arXiv:1906.06579*.
[46] X. Yang et al., "SCRDet: Towards more robust detection for small, cluttered and rotated objects," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8231–8240.
[47] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9626–9635.
[48] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[49] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10334–10343.
[50] H. Wu, Y. Chen, N. Wang, and Z.-X. Zhang, "Sequence level semantics aggregation for video object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9216–9224.
[51] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9756–9765.
[52] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.

Xunkuai Zhou (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree in control science and engineering with Tongji University, Shanghai, China. He also involved in research works as a Visiting Ph.D. Student with the Department of Mechanical and Automation, The Chinese University of Hong Kong, Hong Kong, China. His current research interests include model compression, object detection, object tracking, and the application of unmanned systems in architecture.

Guidong Yang (Graduate Student Member, IEEE) received the B.Eng. degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2018, the M.Eng. degree from SJTU, and the M.Sc. degree from the Polytechnic University of Milan, Milan, Italy, in 2021. He is currently pursuing the Ph.D. degree in mechanical and automation engineering with The Chinese University of Hong Kong, Hong Kong, China. His current research interests include object detection and 3-D reconstruction.

Yizhou Chen is currently pursuing the Ph.D. degree in mechanical and automation with The Chinese University of Hong Kong, Hong Kong, China. His current research interests include parallel computing in mapping, path planning, task planning, and objection detection.

Li Li (Member, IEEE) received the Ph.D. degree from the Shenyang Institute of Automation, Chinese Academy of Science, in 2003.

She joined Tongji University, Shanghai, China, in 2003, where she is currently a Professor with the Department of Control Science and Engineering. Her research interests are in data-driven modeling and optimization and computational intelligence.

Ben M. Chen (Fellow, IEEE) is currently a Professor of mechanical and automation engineering with The Chinese University of Hong Kong (CUHK), Hong Kong. He was a Provost's Chair Professor with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), before joining CUHK, in 2018. He was an Assistant Professor with the Department of Electrical Engineering, State University of New York at Stony Brook, NY, USA, from 1992 to 1993. He has authored/coauthored 100s of journals and conference papers, and a dozen research monographs in control theory and applications, unmanned systems, and financial market modeling. His current research interests are in unmanned systems and their applications.

He is a fellow of the Academy of Engineering, Singapore. He has served on the editorial boards for a dozen international journals, including *Automatica* and IEEE TRANSACTIONS ON AUTOMATIC CONTROL. He is serving as the Editor-in-Chief for *Unmanned Systems* and an Editor for *International Journal of Robust and Nonlinear Control*.