Aprus: An Airborne Altitude-Adaptive Purpose-Related UAV System for Object Detection

Weiqiang Wang[*†], Haiyang Chen[*†], Xingzhou Zhang[‡§], ✉Wei Zhou[*†], Weisong Shi[¶]
[*]Engineering Research Center of Cyberspace, Yunnan University, Kunming 650091, China
[†]School of Software, Yunnan University, Kunming 650091, China
[‡]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[§]University of Chinese Academy of Sciences, Beijing 100190, China
[¶]Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA

Abstract—**Unmanned aerial vehicle (UAV) system plays an important role in new edge scenarios such as disaster relief, situational awareness, and so on. In order to improve efficiency, computing tasks such as object detection are carried out on the drone-mounted computing unit. However, there is a trade-off between flight altitude and mission accuracy. Higher altitudes result in smaller images, which influence the accuracy. In comparison, shorter altitudes reduce the difficulty of detection while decreasing the recognition range, which affects user experience. Therefore, this paper proposes an airborne altitude adaptive UAV System, Aprus, which selects the most suitable object detection models based on a self-adaptation strategy at different altitudes. The selection strategy is based on a purpose-related evaluation indicator, PEI. It comprehensively considers the model's accuracy, recall, and inference speed at the current altitude according to the user's purpose. Moreover, Aprus sends the original image to a divider instead of scaling the image before pushing it to the model, thus ensuring that the original picture information will not be lost. To evaluate the system, we build a high-resolution UAV dataset with altitude, UDWA, which contains 46037 images. From the experiments, Aprus obtained 58.52%mAP, 94.17%mAP^{50}, 66.17%mAR, 1.61FPS results on DJI Manifold 2G when the purpose is set to *Balance* mode, and the system can be adjusted by multiple preset purposes or according to the user customs.**

Index Terms—**Altitude adaptive, Purpose related, Object detection, UAV system, Edge Computing, Embedded system**

I. Introduction

Unmanned aerial vehicles (UAVs) have taken on important tasks in many areas of edge computing, such as emergency rescue, cargo transportation, remote sensing telemetry, power grid inspection at oil sites, etc. In these scenarios, object detection is a basic computing process. The main job of object detection is to describe objects in an image through bounding boxes and attribute categories.

Object detection on UAV edge devices, however, encounters more difficulties than object detection in traditional scenarios. Firstly, traditional object detection is performed from a perspective view. However, UAV object detection is served from a bird's eye view. It introduces many optical problems, such as high shooting height, mostly overhead angles, small target objects, many blank backgrounds, and apparent inconsistencies in light intensity in the area caused by a large viewing angle. Secondly, object detection on UAVs is limited by the computing resources of drones. The current UAVs have been equipped with a 4K resolution camera as standard, which requires strong computing power to process. The drone's energy limits the length of the mission.

Therefore, there is a trade-off between the flight altitude and the accuracy of object detection on UAV edge devices. This brings up the two main problems:

- The flight altitude affects the accuracy of the model. The existing models that can detect tiny objects directly from huge original resolution images required too significant resources to run on edge devices. However, if scaling the picture to a smaller size, although the resources of the edge device can meet the needs of the operation, the scaling will also cause the object to be detected to be compressed to a size that is difficult to detect.
- A single model cannot achieve the best performance at all altitudes. Since many drone cameras cannot zoom, the change of flight altitude will bring about apparent differences in the visual range. And the focus of different purposes is also not the same. It cause a single model to fail to reach the global optimum because models at different altitudes have their advantages and disadvantages.

To solve those problems, we designed an airborne altitude-adaptive purpose-related UAV system for object detection, Aprus. We also publish a high-resolution UAV dataset with altitude, UDWA, to evaluate the performance of Aprus.

As is shown in Figure 1, Aprus runs on the UAV's airborne edge computing device. It first acquires the drone's current video frame and altitude and uses the remoter to request the user's purpose. Then the altitude and purpose are sent to the altitude-driven self-adaptation algorithm to obtain the target size of the image divider and the most suitable model. The high-resolution image divider will convert the original video frame to fit the model's input requirements and use the most convenient model to complete the inference in the next step. The sharded results of the model inference will go through a reassembler with the divide info to splice into a complete result. Finally, finish the object detection and show it to the user through the remoter. In addition, compared with some works using on-ground edge computations [4, 6, 37],

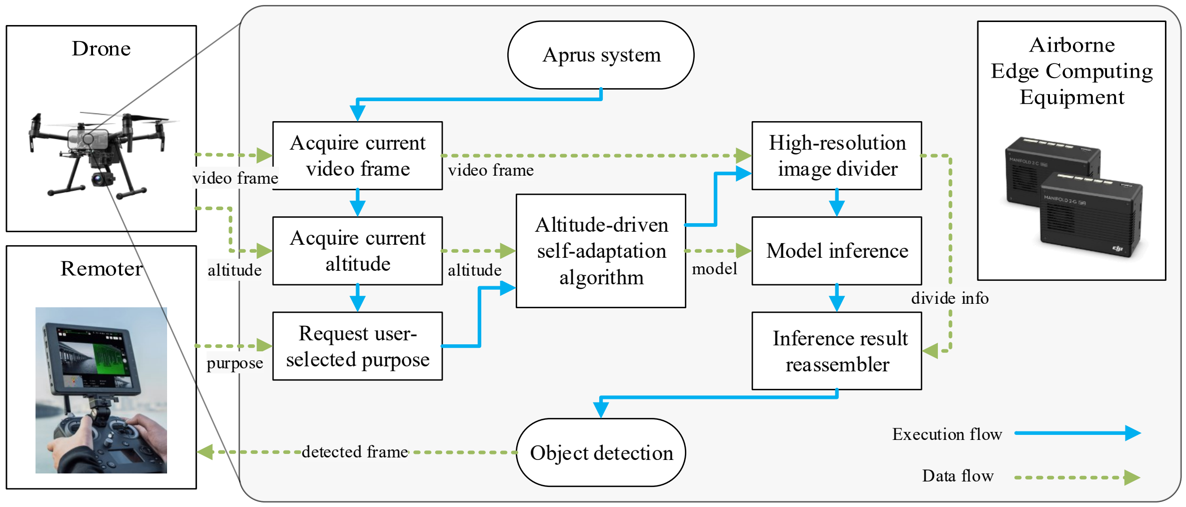

Drone
Remoter
Aprus system
Airborne Edge Computing Equipment
video frame
altitude
purpose
Acquire current video frame
Acquire current altitude
Request user-selected purpose
video frame
altitude
Altitude-driven self-adaptation algorithm
model
High-resolution image divider
Model inference
Inference result reassembler
divide info
detected frame
Object detection
Execution flow
Data flow

Fig. 1. System Structure.

computing on airborne edge devices can eliminate the need to ensure stable and low-latency high-bandwidth communication between air and ground and provide many application scenarios without on-ground edge support. However, it may not be able to match the computationally rich ground resources in terms of processing speed comparison.

The high-resolution UAV dataset, `UDWA`, contains the altitude information and other geographic information taken, which is used to understand the model's performance at different altitudes and provide data support for the development of the `Aprus` system. The most significant feature of this dataset is that it provides image data at different altitudes at the same location, which is extremely important for the altitude-driven self-adaptation algorithm.

Based on the experiment results, we can conclude that the actual performance of `Aprus` is better than all single models, and it can self-adaptive achieve the best detection results at all different altitudes by user's purpose.

The contributions in this paper are as follows:

- We constructed an airborne UAV system for object detection, `Aprus`. It obtains real-time images and altitude from the drone, and adaptive selects the most suitable model for detection according to the user's purpose.
- We published a high-resolution UAV dataset with altitude, `UDWA`, which contains 39 scenes, six altitudes, and a total of 46037 pictures with a resolution of 3840×2160. `UDWA` is the first UAV object detection dataset focused on altitude, as far as we know.
- We obtained a dynamic image pre-processing method of dividing one picture into multiple small-size images to replace the down-scale step in the detection, ensuring that the original picture information will not be lost during the training and testing process. It can generally run under the equipment with limited resources.
- We designed a purpose-related evaluation indicator, `PEI`, and an altitude-adaptive algorithm. The indicator includes seven purpose which can be selected by users. The algorithm is designed to build the airborne detection system for auto-selection at different altitudes.

The rest of the paper is organized as follows. Section II described `UDWA`, the UAV Dataset with Altitude. Section III presented the design and implement of the altitude-adaptive airborne system, `Aprus`. Section IV showed the experiment setup and analysed the experimental results. Section V introduced the related work. Section VI summarized the paper.

II. `UDWA`: UAV Dataset with Altitude

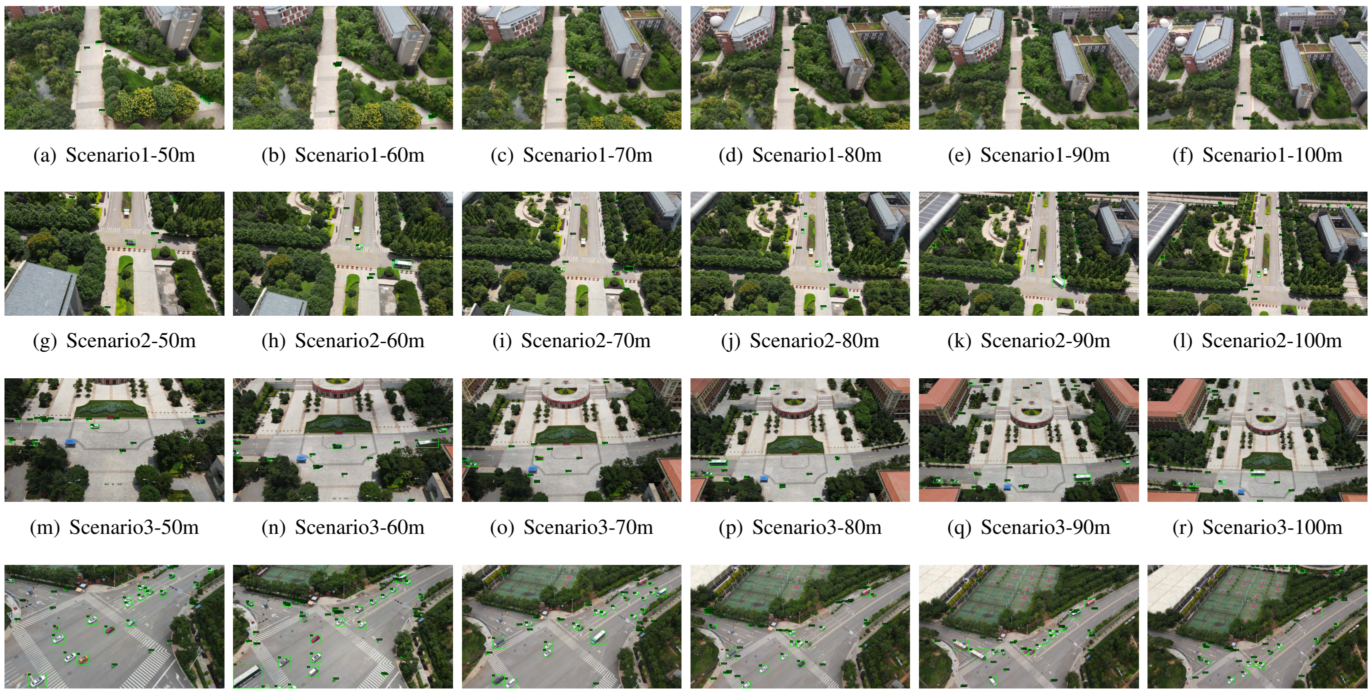
A. *Why Need a New Dataset*

In recent years, researchers have proposed many datasets for drone object detection. However, these datasets lack the altitude label which is necessary to build the altitude-related drone system. For example, VisDrone [39] dataset does not provide the altitude information and the altitude of AU-AIR [2] dataset is limited to 30 meters. Based on the current height of buildings in the city, we set the minimum altitude to 50 meters to avoid collisions. The maximum altitude should not exceed 100 meters for legal and flight safety reasons. Both VisDrone and AU-AIR took a large number of photos at a fixed location, which were repeatedly moved horizontally at similar heights. Therefore these two datasets are not suitable for altitude-driven tasks. These push us to collect a new drone dataset for building the altitude-adaptive system.

B. *Collection Platform*

We used DJI Mavic Air 2[1] and DJI Fly[2] to capture the video stream. As a portable drone, Mavic Air 2 has the ability of capturing high-resolution video, in *4K Ultra HD HDR* mode, the video resolution is 3840×2160 and the sampling frequency is 30 frames per second (FPS).

[1]https://www.dji.com/mavic-air-2
[2]https://www.dji.com/dji-fly



(a) Scenario1-50m (b) Scenario1-60m (c) Scenario1-70m (d) Scenario1-80m (e) Scenario1-90m (f) Scenario1-100m

(g) Scenario2-50m (h) Scenario2-60m (i) Scenario2-70m (j) Scenario2-80m (k) Scenario2-90m (l) Scenario2-100m

(m) Scenario3-50m (n) Scenario3-60m (o) Scenario3-70m (p) Scenario3-80m (q) Scenario3-90m (r) Scenario3-100m

(s) Scenario4-50m (t) Scenario4-60m (u) Scenario4-70m (v) Scenario4-80m (w) Scenario4-90m (x) Scenario4-100m

Fig. 2. UDWA Examples: Four Scenarios with Annotations in Different Altitudes (from 50 meters to 100 meters) .

C. Dataset Description

By leveraging the collection platform, this paper collected the UAV Dataset with Altitude, UDWA. It contains 179 original video clips and the total video length is about 13 hours. The length of each clip is about 4 minutes and 50 seconds. We extracted it to 46028 images in JPEG format.UDWA are publicly available at GitHub[3].

For altitude-driven, we chose 6 flight altitudes: 50 meters, 60 meters, 70 meters, 80 meters, 90 meters, and 100 meters. To focus on the different altitudes, the horizontal position under one location will not change. There are only six certain altitudes, and the camera's downward angle remains unchanged at 45 degrees.

For purpose-related, we selected 39 different places for shooting and data dividing. These places are mainly schools, subway stations, commercial streets, and tourist attractions. We choose these places because they can cover most of the usage scenarios of drones in cities with different purposes.

Moreover, all pictures extracted from the video maintain a high resolution of 3840×2160, and no down-sampling is performed. The original video is 30fps. According to the actual speed of the object to be detected, the first frame of each second is selected as the extracted picture.

All data was captured in public venues or with the permission of the venue owner. Furthermore, based on the shooting height and angle of view, there will be no clear faces in the dataset, avoiding potential privacy issues.

[3]https://github.com/Aprus-system/UDWA

D. Dataset Annotation

The proposed system mainly focus on the road and pedestrian monitoring scenarios, so the tags of the dataset have only two categories: *person* and *car*. Here we define that the person who appears in the picture regardless of the posture is marked as *person*, and all four-wheeled and three-wheeled vehicles are marked as *car*. It should be noted that motorcycles and bicycles themselves do not belong to *car*, but their drivers and passengers belong to *person*.

To complete the self-adaptation system, we selected four places with fewer goals and marked a total of 2866 pictures. Each of these pictures contains about 10-50 marker boxes, and most of them include objects of two categories simultaneously. It takes about 5 minutes on average to mark such a picture. We carry out 8 hours of marking a day and finally complete the marking of 2866 pictures in 30 days. We will continue to annotate other pictures and use them for future research. Figure 2 presents the example of UDWA, which includes four scenarios and each contains the annotated information from 50 meters to 100 meters.

The 2866 annotated pictures are divided with 8:2 into the training set and validation set. We call the remaining unlabeled pictures the test-challenge set. The final training set contains 2290 pictures, the validation set contains 576 pictures, and the test-challenge set contains 43162 pictures. These pictures are divided according to 50 meters, 60 meters, 70 meters, 80 meters, 90 meters, 100 meters, which is convenient for testing the accuracy and other performance at different altitudes. In the future, we will continue expanding the UDWA dataset and improve the annotation work with the help of the community.

III. The system design and implementation of Aprus

Aprus is designed to run on airborne edge computing devices, and it obtains data through the hardware interface between the edge device and the drone. The system will continuously process data and return the detection results.

The main component of Aprus consists of three parts: a high-resolution image divider, a purpose-related evaluation indicator, and an altitude-driven self-adaptation algorithm.

A. High-resolution Image Divider

Aprus includes a dynamic high-resolution image divider, as is shown in Figure 3, which divides the raw image into multiple small-size images. Resizing the raw image is the first step of object detection, which is used to meet the input requirements of different detect models. Meanwhile, resizing the image can reduce the hardware resources required for computing. In order to apply to the specific scenarios of UAVs, Aprus made two innovations in the resizing process.

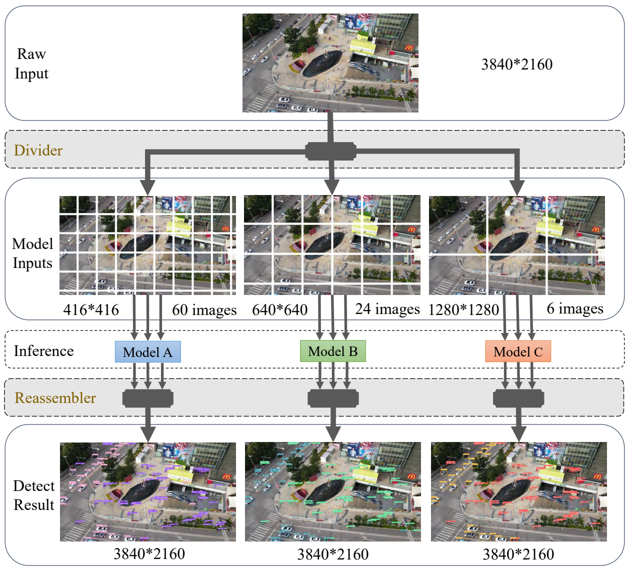


Fig. 3. Schematic Diagram of Image Divider.

Firstly, Aprus adopts the dividing method instead of down-scaling when resizing the high-resolution images.

Due to the small proportion of the object in the original image in the drone scene, the traditional scaling method scales the high-resolution image to a small-size image. It then takes this small image as the input of the model, which results in a large amount of information lost [21]. For example, most persons will be completely invisible if the image is scaled to the widely-used 640*640 size. However, if the scaled size is further increased beyond 1280*1280, most detection models will be hard to train and cannot run on airborne edge devices.

The difference is that Aprus splits a high-resolution image into multiple smaller images of a specific size, and then the reassembler will generate the final result. The dividing method refers on the CNN sliding window algorithm [25]. Figure 3 shows the process of dividing the original image into three commonly used input sizes of object detection models: 416*416, 640*640, 1280*1280, and recombining them after model inference. It reserved every detail in the original image.

Secondly, Aprus dynamically changes the order of divisions when passing in a new image.

Since the divider turns a complete image into multiple small copies, it may reduce the detection rate of objects on the boundaries of the images. In order to solve this problem, Aprus's divider dynamically changes the sliding window order when running, such as switching from left-right to right-left or from top-bottom to bottom-top. It will also add padding to the edge before the sliding split and delete it in the sub-image to change the position of the borderline in the original while keeping the total size of the output unchanged. Aprus uses these two methods to avoid objects on the boundary line that are often unable to be detected by the model because the boundary line is always in a fixed position.

In addition, the reduction of local detection rate can be ignored in the long-term detection of UAV scenes. Since both the drone and the detected object are constantly in motion, the object will not always be located on the boundary line, so it will not have a long-term impact on the overall result.

B. Purpose-related Evaluation Indicator

Different configurations of the detection model can generate a large number of sub-models that behave differently. Therefore, Aprus designs a new indicator for evaluating the model's actual performance at different altitudes intuitively.

This paper chooses *mean Average Precision* (mAP), *mean Average Precision at Intersection over Union equals 0.5* (mAP^{50}), and *mean Average Recall* (mAR) in the MS COCO assessment protocol [19] as part of the new evaluation indicator. As the main indicator of COCO, mAP can well reflect the detection ability of the module. However, since it is the average evaluation result in the interval of [0.5, 0.95] with a step length of 0.05 as the value of the *Intersection over Union* (IoU). mAP^{50} is evaluated with a fixed IoU value of 0.5, which can reflect the user's perception in the actual detection task. As a measure of the missed detection rate, mAR is not taken seriously in many object detection studies. Here selects mAR as one of the indicators since it is vital in practical scenarios with small and crowded detection objects.

$$\begin{aligned} PEI_{(i,h)} =& \frac{1}{\alpha+\beta+\gamma}(\alpha * \sqrt{\frac{{mAP_{(i,h)}}^2 + {mAP^{50}_{(i,h)}}^2}{2}} \\ &+ \beta * mAR_{(i,h)} + \gamma * (1 + \frac{\arctan{(-t_{(i,h)})}}{\pi/2})) \end{aligned} \tag{1}$$

This paper proposes a Purpose-related Evaluation Indicator, $PEI_{(i,h)}$, to select the most suitable model for application scenarios at different altitudes and determine the weight of accuracy, recall, and speed based on the user's purpose.

$PEI_{(i,h)}$ is defined in Eq (1), which indicate the score of the i-th model under h altitude. Among them, we use the root mean square to measure the impact of mAP and mAP^{50} on the accuracy rate. mAR presents the recall rate. To keep the speed consistent with the other two dimensions, we leverage an arctangent function to normalize the speed to [0, 1], which is often used in related work [12, 32].

Where mAP, mAP^{50}, and mAR are the COCO indicators of the i-th model at altitude h, and t is the average time required for the model to infer once. α, β, and γ are preset coefficients for different purposes.

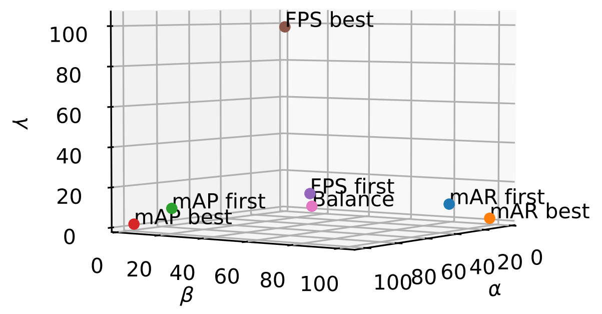


Fig. 4. 3D Indicator Space Consisting of mAP, mAR, and FPS.

As is shown in Figure 4, we make up a three-dimensional space of mAP, mAR, and FPS and select seven specific points in this 3D space to form the new evaluation indicator. The maximum points of the three axes are regarded as the *best* purposes, the three points that only deviate from one of the axes as the *first* purposes, and the average point on the three axes is regarded as the *Balance* purpose. Therefore, the seven points are defined as seven typical purposes: *mAR first, mAR best, mAP first, mAP best, FPS first, FPS best*, and *Balance*. The specific parameter values are shown in Table I.

TABLE I
PRESET COEFFICIENTS IN TYPICAL PURPOSES

Purpose	mAP α	mAR β	FPS γ
mAR first	8	84	8
mAR best	1	98	1
mAP first	84	8	8
mAP best	98	1	1
FPS first	43	43	14
FPS best	1	1	98
Balance	46	46	8

In most usage scenarios, *Balance* purpose can achieve optimal accuracy, recall, and speed. However, in some scenarios, specific indicators may be more critical. For example, the first thing to ensure in disaster relief is that no target is missed. In this case, *mAR first* is more worthy of choice than *Balance*. In urban road monitoring, object detection accuracy is more important because we cannot accept the award of cars as persons, so we should choose *mAP first*. In high-speed intersection flow monitoring, the target's speed is breakneck, and the detection speed of the system is more important. In this case, *FPS first* is more suitable. At the same time, to cover some more extreme scenarios, we also designed three *best* purposes to cope with the higher requirements for a single indicator. In summary, the selected seven points can completely cover all corners of the three-dimensional space.

C. Altitude-driven Self-adaptation Algorithm

Based on the methods in the previous two sections, we design the Algorithm 1 to build an airborne altitude-adaptive purpose-related UAV system for object detection.

Algorithm 1:

```
   Input: User selected purpose: $P$
1  while not terminated do
2      $h, v \leftarrow$ Get the current altitude and video frame from the drone;
3      if The algorithm is the first time running or The user is requesting to modify the purpose then
4          $P \leftarrow$ Get new purpose from user's request;
5          if $P$ in Preset Coefficients then
6              $PEI \leftarrow$ Get the pre-computed $PEI$ table by $P$;
7          else
8              $\alpha, \beta, \gamma \leftarrow$ Get the coefficients from the user's custom setting;
9              $PEI \leftarrow$ Calculate each model's $PEI$ by $\alpha, \beta, \gamma$;
10     if The algorithm is the first time running or The current altitude had been changed or The $PEI$ table had been changed then
11         $i_{max} \leftarrow$ Get the $i$ in MAX($PEI_{(i,h)}$);
12         $f(x) \leftarrow$ Get the $i_{max}$-th of models;
13         $divider(v) \leftarrow$ Get the image divider with the input sizes of $f(x)$;
14     $R_{raw} \leftarrow$ An empty list to bearer detection results;
15     foreach $x_i$ in $divider(v)$ do
16         $r_i \leftarrow$ Evaluate $f(x_i)$;
17         Extend $R_{raw}$ by $r_i$;
18     $reassembler(l) \leftarrow$ Get the reassembler from $divider(v)$;
19     $R \leftarrow$ Evaluate $reassembler(R_{raw})$;
20     Send $R$ to the user;
```

Algorithm 1 defines a loop that continuously obtains the drone's current video stream and altitude information. The PEI's preset coefficients are determined according to the user's purpose. The PEI of all models at different altitudes and for different preset purposes have been evaluated before system operation. If the user chooses to customize PEIs coefficients, the system will recalculate the PEI value based on each model's existing altitude-related evaluation data. Therefore, it is only necessary to adaptively select the model with the highest PEI value for reasoning based on the current

TABLE II
DIFFERENT MODEL CONFIGURATIONS EXCERPT (12/75)

Basic Network	Detection Framework	Deep Learning Framework	Network Configuration	Divide Size(px)	mAP	mAP^{50}	mAR	FP16 Time(s)	TRT Time(s)
yolox	YOLOX	Pytorch	nano	416	0.385	0.766	0.5	1.53	0.592
yolox	YOLOX	Pytorch	s	640	0.479	0.872	0.597	3.937	0.907
yolox	YOLOX	Pytorch	x	1280	0.59	0.945	0.664	31.932	N/A
yolov5	ultralytics/yolov5	Pytorch	n5-6	640	0.466	0.85	0.572	1.671	0.363
yolov5	ultralytics/yolov5	Pytorch	m6	1280	0.593	0.946	0.671	7.428	1.476
yolov5	ultralytics/yolov5	Pytorch	x6-6	1280	0.597	0.948	0.678	25.504	4.699
ppyolo	PaddleDetection	PaddlePaddle	tiny	640	0.354	0.74	0.454	1.262	0.638
ppyolo	PaddleDetection	PaddlePaddle	r18vd	640	0.443	0.832	0.578	4.095	0.82
ppyolov2	PaddleDetection	PaddlePaddle	r101vd_dcn	1280	0.595	0.948	0.705	14.773	6.459
faster_rcnn	MMDetection	Pytorch	swin-t_fpn_ms-crop	1280	0.568	0.934	0.646	20.699	N/A
retinanet	MMDetection	Pytorch	pvtv2-b0_fpn	1280	0.541	0.907	0.631	19.373	N/A
faster_rcnn	PaddleDetection	PaddlePaddle	dcn_r50_fpn	1280	0.548	0.915	0.646	19.816	19.853

information. The image will also be cut according to the input size required by the model and reorganized after the inference is completed. Finally, the complete detection result is sent to the user, and the loop continues to be executed.

Notably, we implemented the system only in the situation that when loading a few selected models with the purpose changing. This steps from the fact that the user will not change the purpose frequently in a flight mission, and the number of models selected under a single purpose is small and fixed. This can effectively reduce the performance overhead caused by reloading the model when the altitude changes. It can also avoid loading too many models at one time beyond the range that the memory of the edge device can accommodate.

IV. EXPERIMENT

A. System Equipment and Model Preparation

To obtain the real-time video stream and current status information of the drone, we chose the combination of DJI M210 V2[4] and Manifold 2-G[5]. DJI M210 V2 is a medium-sized drone that can be used in various industrial applications. Manifold 2-G is an edge computing device that can be mounted on drones by this combination. And with the help of the official Onboard SDK[6] to get various data streams and control drones in real-time.

This paper selects seven object detection models as the benchmarks: YOLOX [11], YOLOV5 [15], PPYOLO [22], Swin [20], PVT [36, 35], YOLOV3 [26], and Faster-R-CNN, referring to the VisDrone Object Detection in Images Challenge [3]. As the general detection models, YOLOX, YOLOV5, and PPYOLO have been widely used in various situations and have better stability and reliability. They are easier to deploy on airborne edge devices since they have optimized resource-constrained scenarios and require less computation. Swin [20] and PVT [36, 35] are two modern transformer-based detection networks, which all perform well on the COCO dataset [19], so we hope to explore their role in the Aprus system as well. Two traditional models, YOLOV3 and Faster-R-CNN network optimized by DCN [5, 40] are also used as part of the experimental comparison to increase the robustness of the results further.

Before evaluating the model, we need to train the model first. To avoid overfitting during training, we mix the `UDWA` training set and VisDrone-DET. At the same time, to ensure the objectivity of comparison, the different configurations of YOLOX, YOLOV5, PPYOLO, and YOLOV3 are all trained at a batch size of 8. However, the other models such as Swim and PVT can only be trained with a batch size of 1 due to GPU memory limitations. All training was run under NVIDIA A100-PCIe-40GB with 36 epochs, and it took 4 GPU cards and two months to complete the training. The different configurations of the seven detection models generated seventy five sub-models.

After completing the training, we obtain the inference speed of different models on edge devices. We convert all models to TensorRT and use FP16 inference to achieve a faster speed. We did not use TensorRT in INT8 mode because it requires calibration and significantly impacts inference accuracy.

As the result of model preparation, Table II details the basic network, detection framework, deep learning framework, network configuration, and divide size adopted by the model. The inference time of all models before and after TensorRT acceleration is also compared. It should be noted that if the inference time in the table is N/A, the model cannot be accelerated by TensorRT. In addition, the mAP, mAP^{50} and mAR of each model are the average values at all altitudes, which are collected from the same scenario and only provided for reference. Due to space constraints, this table shows only the most representative 12 configurations, and the results by altitude will not be listed and will be used directly for subsequent model selection. For the configuration of other models, please refer to the full table on the GitHub[7].

From the experimental results, the YOLO series model has achieved a maximum speed of 5x by TensorRT acceleration compared with the unconverted model. In contrast, other models have no noticeable effect or even a slight decline. As

[4]https://www.dji.com/matrice-200-series-v2
[5]https://www.dji.com/manifold-2
[6]https://developer.dji.com/onboard-sdk/

[7]https://github.com/Aprus-system/table/blob/main/model_cfg.md

a result, YOLO series models will use TensorRT acceleration in FP16 mode in subsequent experiments, and other models will only use FP16 inference. Furthermore, we also observe that the YOLO series always outperforms the other series regardless of the framework based on it, while the state-of-the-art Swin and PVT perform much worse than expected. This may be related to the inflexibility to support high-resolution images and limitations of local attention are perhaps the main bottlenecks on the transformer-based models[30].

Result 1: The accuracy of the detect model will be affected by the image divide size. The smaller the size, the lower the accuracy. Therefore, `Aprus`*' divider selects the image with the largest size within the computing power of the edge device.*

B. Self-adaptation Model Selection

Based on the `PEI` and the `UDWA` verification set, which was partitioned by altitudes, we can quickly obtain model selection results by the altitude-driven self-adaptation algorithm for different purposes. The results of self-adaptation selection can be seen in Table III.

TABLE III
RESULTS OF SELF-ADAPTATION SELECTION

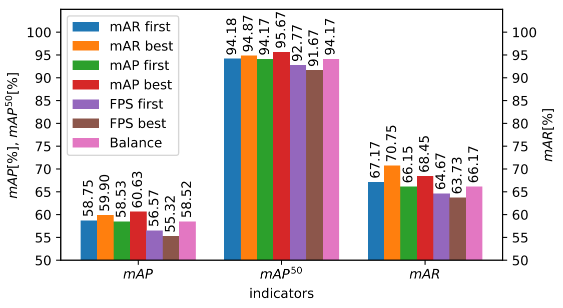
Purpose	50M	60M	70M	80M	90M	100M
mAR first	yolov5 s5 1280	yolov5 s6 1280	yolov5 s5 1280	yolov5 s6-6 1280	yolov5 s6 1280	ppyolo v2r101 1280
mAR best	ppyolo v2r101 1280	ppyolo v2r101 1280	ppyolo v2r101 1280	ppyolo v2r50 1280	ppyolo v2r50 1280	ppyolo v2r101 1280
mAP first	yolov5 s6 1280	yolov5 s6-6 1280	yolov5 s6 1280	yolov5 s6-6 1280	yolov5 s6 1280	yolov5 s5 1280
mAP best	ppyolo v2r50 1280	yolov5 l6-6 1280	yolov5 l6 1280	yolov5 x5 1280	yolov5 s5-6 1280	yolov5 x5 1280
FPS first	yolov5 n6-6 1280	yolov5 n6-6 1280	yolov5 n6-6 1280	yolov5 s6-6 1280	yolov5 s6 1280	yolov5 n6-6 1280
FPS best	yolov5 n6-6 1280	yolov5 n6-6 1280	yolov5 n6-6 1280	yolov5 n6-6 1280	yolov5 n6-6 1280	yolov5 n6-6 1280
Balance	yolov5 s6 1280	yolov5 s6-6 1280	yolov5 s5 1280	yolov5 s6-6 1280	yolov5 s6 1280	yolov5 s5 1280

From the results of self-adaptation model selection, all the selected models are YOLO series. For most purposes, the model based on YOLOV5 has been selected the most times. The model based on PPYOLOV2 performs best under the purpose of focusing on mAR. The YOLOX-based and YOLOV3-based model, like other series models, was not selected to enter the system, indicating that its comprehensive performance is worse than the above two models.

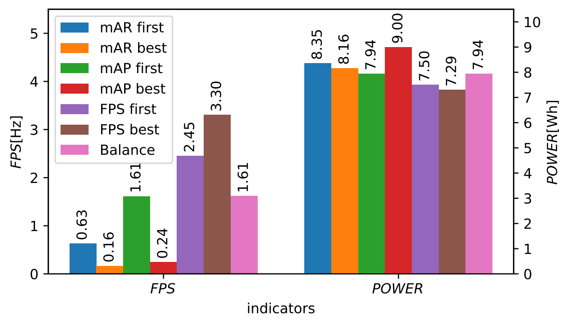
C. System Performance Evaluation

After completing the model selection, we built the object detection system based on the results of the module selection and evaluated the overall altitudes performance of the system. The results are shown in Image 5.

From the test results, *FPS first*'s comprehensive `PEI` increased the most, with 46.80%, while *mAR first* increased the least, but it also had 6.64%. This proves that the proposed



(a) mAP, mAP^{50} and mAR



(b) FPS and $POWER$

Fig. 5. System Performance in Different Purposes.

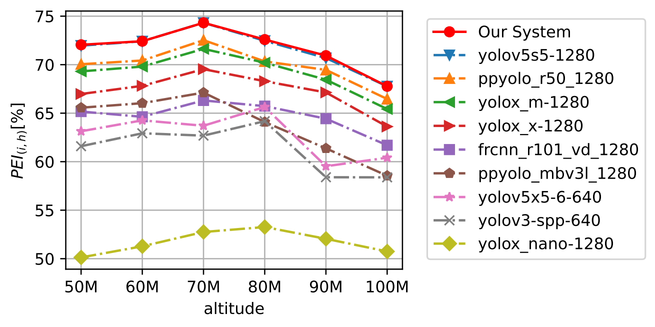
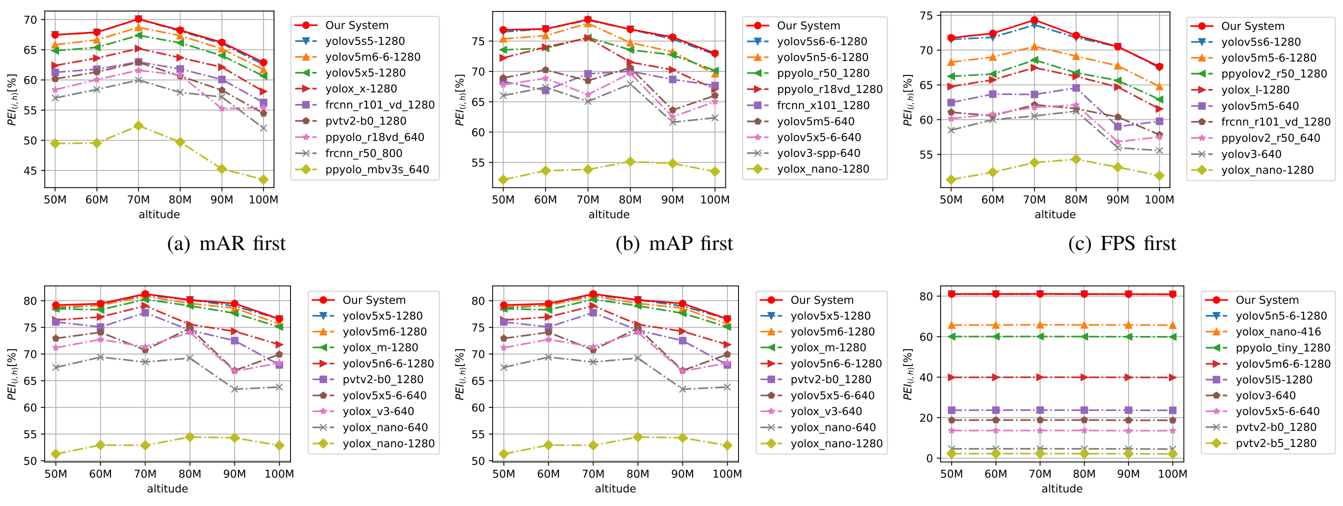


Fig. 6. Evaluate System in Balance Purpose.

`Aprus` is effective. In addition, considering all the purposes, the detection performance of `Aprus` increased from 50 meters, reached the highest point at 70 meters, and then began to descend until the minimum at 100 meters.

From the performance evaluation data, the system realizes the focused selection of different modules for different purposes. Among them, we also found that if we focus on mAR, the reduction in FPS is more evident than when we focus on mAP. If you pay attention to FPS, other evaluation indicators will be reduced. All indicators are more average in the *Balance* purpose, which should be the most suitable choice for most practical tasks.

Power consumption is a key focus of drone researchers, so this paper evaluates the energy consumption of `Aprus` for different purposes and show it in the last column of Figure 5(b). The battery capacity of the drone is 174.6Wh and the maximum flight time is 34 minutes. So the $POWER$ of the system is defined as the total power consumed by `Aprus`



(a) mAR first (b) mAP first (c) FPS first

(d) mAR best (e) mAP best (f) FPS best

Fig. 7. Evaluate System in Different Purposes.

to infer images at different altitudes for 34 minutes continuously. All power consumption data comes from connecting a high-precision ammeter between the airborne edge computing equipment and the drone.

Result 2: According to the experimental results, `Aprus` *itself has little impact on the power consumption of the whole drone system. The system will consume a maximum of 9.00Wh, which is about 5.15% of the battery capacity, and minimum consumption of 7.29Wh, which is about 4.17% of the battery capacity.*

At the same time, we compare the adaptive approach proposed in this paper with other single-model without adaptation approaches. We first set the `Aprus` in the *Balance* purpose. The results are shown in Figure 6. For the convenience of the display, here, only 9 of the 75 single models are shown for comparison with `Aprus`.

Result 3: `Aprus` *is better than all single models, and it can achieve the best detection results at all different altitudes. Compared to randomly selecting one of the single models,* `Aprus`*'s comprehensive* `PEI` *at all altitudes increased by 6.64%. In the sub-altitude comparison, the highest increase is 6.87% at an altitude of 90 meters, and the lowest increase is 5.78% at 80 meters.*

Similarly, we also evaluate the performance under other purposes, and the results are shown in Figure 7 and consistent with the conclusions under the *Balance* purpose.

Outlook: As seen from Figure 7(f), the results of our `PEI` *indicator under the purpose of* FPS best *are relatively unitary, unlike other purposes with apparent changes.*

This result is that the inference speed of different models does not change because the field of view of the input image is changed at different altitudes, not the size of the input image. However, if the overall weight of speed on the indicator is too large, the influence of other indicators will be mostly ignored, resulting in a single evaluation result. This problem can also be seen in Table III's selection. In the future, we will consider more models with different inference speeds at different altitudes, which can enrich the meaning of the purpose of *FPS best* and improve the robustness of `Aprus`.

V. Related Work

A. UAVs Datasets

Datasets play an essential role in the training of deep learning networks, and their quality directly affects accuracy. Among them, the MS-COCO [19] and Pascal-VOC [9] datasets are popular to train and evaluate the mainstream object detection models. With the proliferation of UAV-based applications, aeronautical vision datasets that is used for training object detection models have gradually emerged. Two data sets with the greatest reference value are as follows:

1) VisDrone Dataset: The VisDrone [39] Dataset contains 179,264 frames and 10,209 static images acquired from the aerial perspective collected under various weather and lighting conditions using different drone platforms. It manually annotated more than 2.6 million bounding boxes.

2) AU-AIR Dataset: The AU-AIR [2] Dataset is a multi-modal aerial dataset captured by a UAV. AU-AIR meets vision and robotics for UAVs. The most unusual are frames labeled with time, GPS, IMU, altitude, linear velocities of the UAV.

At the same time, many data sets for UAVs were proposed, such as UAVDT [8], UAVid [24], UAV-BD [33]. However, none of these datasets contains images and labels at different altitudes for the same location, which is necessary to build an altitude adaptive drone system. Therefore, this paper built a dataset with altitude labels in Section II.

B. Object Detection Model

Object detection models are divided into two routes according to whether deep learning is used. For those that do not use deep learning, three classic detectors: Viola-Jones [31], HOG [7] and DPM [10] are mainly used.

However, with the manual selection of features, the performance of the technology tends to be saturated. Currently, the most effective object detection model is based on deep learning. It is mainly divided into two categories:

1) two-stage model: It first proposes an image area that may contain an object by a region-based CNN and then classifies the area into a predefined object category. Faster-R-CNN [28] is one of the well-known two-stage models.

2) one-stage model: It has only one network to directly convert the object detection problem into the bounding box, without the need for a separate image classifier to perform the second-stage processing. YOLO [27] and RetinaNet [18] are popular object detectors that belong to one-stage models.

C. Airborne UAV System Design

At present, Airborne UAV systems is mostly designed for collecting data and executing the lightweight tasks to meet the application needs. This section presents several common airborne UAV system research and development directions:

1) environment monitoring: In 2017, a change detection system based on histogram equalization and RGB local binary pattern (RGB-LBP) operator [1] was proposed for wide-area monitoring of small low-altitude drones. In 2020, a agricultural automatic detection system [38] was proposed, which uses YOLOV3 and YOLOV3-tiny for weed detection.

2) human detection: In 2019, an unsupervised human detection system for on a fully autonomous drone [23] was proposed. It combines global navigation satellite system for accurate human detection and rescue equipment freed. In 2020, two drone-based surveillance applications using aerial thermal imaging human detection systems [16, 17] were proposed by the same developer. They were based on FCOS and YOLOv4 to detect humans from thermal data.

There have also recently been some UAV systems for general object detection [14, 13, 34]. Among them, a system used the most advanced object detection algorithm to perform real-time object detection on UAV [29] was proposed in 2017, with the great reference value for this paper. It provides the best object detection configuration selection, which makes us germinate the idea of self-adaptation algorithm selection.

VI. Conclusion

In this paper, we constructed an airborne altitude-adaptive purpose-related UAV system for object detection, `Aprus`. To develop the system, we collected and provided a high-resolution UAV dataset with altitude, `UDWA`, which is publicly available at GitHub. `UDWA` contains 39 scenes, 6 altitudes (from 50 meters to 100 meters), and 46037 pictures with a resolution of 3840*2160. We leveraged 2866 annotated pictures to build the system. We elaborately constructed seven different purposes based on three technologies: the high-resolution image divider, purpose-related evaluation indicator (`PEI`) for model selection, and altitude-driven self-adaptation algorithm. Among the most commonly used *Balance* purposes, the `Aprus` obtained 58.52%mAP, 94.17%mAP^{50}, 66.17%mAR, and 1.61FPS results on DJI Manifold 2G. It consumes only 7.94Wh of power during one flight. and the `PEI` has increased by 6.64% compared to without the `Aprus` system.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 62162067 and 62101480, in part by the Yunnan Province Science Foundation under Grant No.202005AC160007, No.202001BB050076, Research and Application of Object detection based on Artificial Intelligence, in part by the China Postdoctoral Science Foundation under Grant No.2021M693227, and in part of Zhejiang Lab under Grant No.2020KE0AB02.

References

[1] Danilo Avola et al. “Aerial video surveillance system for small-scale UAV environment monitoring”. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2017, pp. 1–6.

[2] Ilker Bozcan and Erdal Kayacan. “Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8504–8510.

[3] Yaru Cao et al. “VisDrone-DET2021: The Vision Meets Drone Object Detection Challenge Results”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2847–2854.

[4] Haowei Chen et al. “AdaDrone: Quality of Navigation Based Neural Adaptive Scheduling for Edge-Assisted Drones”. In: *2022 IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2022.

[5] Jifeng Dai et al. “Deformable Convolutional Networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017.

[6] Minghui Dai et al. “Vehicle assisted computing offloading for unmanned aerial vehicles in smart city”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.3 (2021), pp. 1932–1944.

[7] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*. Vol. 1. Ieee. 2005.

[8] Dawei Du et al. “The unmanned aerial vehicle benchmark: Object detection and tracking”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 370–386.

[9] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[10] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. “A discriminatively trained, multiscale, deformable part model”. In: *2008 IEEE conference on computer vision and pattern recognition*. Ieee. 2008.

[11] Zheng Ge et al. “YOLOX: Exceeding YOLO Series in 2021”. In: *arXiv preprint arXiv:2107.08430* (2021).

[12] Yue-Jiao Gong et al. “Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.2 (2011), pp. 254–267.

[13] Sagar Jha et al. “Visage: Enabling timely analytics for drone imagery”. In: *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 2021, pp. 789–803.

[14] Shiqi Jiang et al. “Flexible high-resolution object detection on edge devices with tunable latency”. In: *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 2021.

[15] Glenn Jocher et al. “ultralytics/yolov5: v5. 0-YOLOv5-P6 1280 models”. In: *AWS, Supervise. ly and YouTube integrations* (2021).

[16] Prashanth Kannadaguli. “FCOS Based Human Detection System Using Thermal Imaging for UAV Based Surveillance Applications”. In: *2020 IEEE Bombay Section Signature Conference (IBSSC)*. IEEE. 2020.

[17] Prashanth Kannadaguli. “YOLO v4 based human detection system using aerial thermal imaging for uav based surveillance applications”. In: *2020 International Conference on Decision Aid Sciences and Application (DASA)*. IEEE. 2020, pp. 1213–1219.

[18] Tsung-Yi Lin et al. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[19] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[20] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[21] Ziming Liu et al. “HRDNet: high-resolution detection network for small objects”. In: *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2021, pp. 1–6.

[22] Xiang Long et al. “PP-YOLO: An Effective and Efficient Implementation of Object Detector”. In: *arXiv preprint arXiv:2007.12099* (2020).

[23] Eleftherios Lygouras et al. “Unsupervised human detection with an embedded vision system on a fully autonomous UAV for search and rescue operations”. In: *Sensors* 19.16 (2019), p. 3542.

[24] Ye Lyu et al. “UAVid: A semantic segmentation dataset for UAV imagery”. In: *ISPRS journal of photogrammetry and remote sensing* 165 (2020), pp. 108–119.

[25] George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. “Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 390–399.

[26] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[27] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[28] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015), pp. 91–99.

[29] Nils Tijtgat et al. “Embedded Real-Time Object Detection for a UAV Warning System”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*. Oct. 2017.

[30] Zhengzhong Tu et al. “Maxim: Multi-axis mlp for image processing”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5769–5780.

[31] Paul Viola and Michael J Jones. “Robust real-time face detection”. In: *International journal of computer vision* 57.2 (2004), pp. 137–154.

[32] Gaolin Wang et al. “Adaptive compensation method of position estimation harmonic error for EMF-based observer in sensorless IPMSM drives”. In: *IEEE Transactions on Power Electronics* 29.6 (2013).

[33] Jinwang Wang et al. “Bottle detection in the wild using low-altitude unmanned aerial vehicles”. In: *2018 21st International Conference on Information Fusion (FUSION)*. IEEE. 2018, pp. 439–444.

[34] Junjue Wang et al. “Bandwidth-efficient live video analytics for drones via edge computing”. In: *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE. 2018, pp. 159–173.

[35] Wenhai Wang et al. “PVT v2: Improved baselines with Pyramid Vision Transformer”. In: *Computational Visual Media* (2022), pp. 1–10.

[36] Wenhai Wang et al. “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[37] Bo Yang et al. “Offloading optimization in edge computing for deep-learning-enabled target tracking by internet of UAVs”. In: *IEEE Internet of Things Journal* 8.12 (2020), pp. 9878–9893.

[38] Rufei Zhang et al. “Weed location and recognition based on UAV imaging and deep learning”. In: *International Journal of Precision Agricultural Aviation* 3.1 (2020).

[39] Pengfei Zhu et al. “Vision meets drones: A challenge”. In: *arXiv preprint arXiv:1804.07437* (2018).

[40] Xizhou Zhu et al. “Deformable convnets v2: More deformable, better results”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9308–9316.