A Surveillance Video Real-Time Object Detection System Based on Edge-Cloud Cooperation in Airport Apron

Zonglei Lyu 1,2,* and Jia Luo 1,2

1 College of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China
2 Key Laboratory of Smart Airport Theory and System, Civil Aviation Administration of China, Tianjin 300000, China
* Correspondence: zllv@cauc.edu.cn

Abstract: Airport apron carries a lot of preparations for flight operation, and the advancement of its various tasks is of great significance to the flight operation. In order to build a more intelligent and easy-to-deploy airport apron operation analysis guarantee system, it is necessary to study a low-cost, fast, and real-time object detection scheme. In this article, a real-time object detection solution based on edge cloud system for airport apron operation surveillance video is proposed, which includes lightweight detection model Edge-YOLO, edge video detection acceleration strategy, and a cloud-based detection results verification mechanism. Edge-YOLO reduces the amounts of parameters and computational complexity by using model lightweight technology, which can achieve better detection speed performance on edge-end embedded devices with weak computing power, and adds an attention mechanism to compensate for accuracy loss. Edge video detection acceleration strategy achieves further detection acceleration for Edge-YOLO by utilizing the motion information of objects in the video to achieve real-time detection. Cloud-based detection results verification mechanism verifies and corrects the detection results generated by the edge through a multi-level intervention mechanism to improve the accuracy of the detection results. Through this solution, we can achieve reliable and real-time monitoring of airport apron video on edge devices with the support of a small amount of cloud computing power.

Keywords: object detection; edge computing; edge computing; intelligent video surveillance; edge-cloud cooperation

1. Introduction

Video surveillance has become very important in nowadays. An enormous number of surveillance cameras are installed at public places, especially in railway stations and airports. These surveillance cameras play the role of not only surveillance of infrastructural property and public safety, but also sensor for data collection. The flight operation process is mainly divided into two stages, the flight stage and the ground service stage. During the flight phase, the flight is dispatched by air traffic control to complete the transport of passengers under the service of the crew members. During the ground service stage, the flight needs to complete various pre-flight preparations, which can be divided into five processes including passengers, luggage, cargo, fueling, and cleaning [1]. These operations will be completed on the apron. Consequently, the normal advancement of airport apron operations is very important for the normal operation of the flight, and its various operating data are of great significance for flight security and airport operation analysis. However, for airport apron operations, it is difficult to collect various operational data using traditional sensors. At present, the main method adopted is to use humans to monitor the operation monitoring video to realize the guarantee of the operation process. Nonetheless, the human processing will inevitably lead to a series of problems such as untimely response and human physiological fatigue.
With the development of artificial intelligence technology, some researchers try to use the object detection technology to create a more intelligent video surveillance system [2]. Compared with manual video surveillance, intelligent video surveillance cannot only process faster and better, but also save a lot of unnecessary labor costs. However, it has higher requirements for computing resources and performs poorly on embedded devices with weak computing power. From the perspective of security assurance and intelligent video analysis, it is necessary to conduct real-time detection of surveillance video on the apron. However, taking the current typical one-stage object detection model YOLOv5s [3] as an example, using a standard 640 × 640 input, it can detect about three to four frames per second on the Jetson Nano and about two frames per second on the Raspberry Pi 4B, which is far from real-time detection. The use of GPU-equipped cloud servers can certainly solve the real-time problem, but it will also put a lot of pressure on bandwidth, and its cost and power consumption cannot be ignored, which greatly limits its practical value.

Some researchers use methods such as model pruning and model quantization to reduce the computational load and memory usage of the detection model, so that it can be better deployed on edge embedded devices. Chen [4] proposed a light YOLOv4 pruned using the scientific control-based neural network pruning algorithm for citrus detection under orchard environment. Kim [5] proposed a low bit-based convolutional neural network, which uses 1-bit weights to reduce the kernel parameter size and 8-bit activations to increase the speed for one-class object detection. However, the pruned and the quantized models suffer from decreased detection accuracy and increased deployment difficulty. Meanwhile, the models also face the challenges of insufficient generalization ability when migrated to new scenarios, due to so many monitoring scenarios on the apron. The special equipment is required to speed up the detection of the quantized model. In recent years, model lightweight research in the field of image classification has developed rapidly, and a series of efficient image classification networks such as MobileNet [6–8], ShuffleNet [9,10], and EfficientNet [11] have emerged. Inspired by this, we build a light object detection model for the edge devices by combining model lightweight techniques with object detection tasks and propose a cloud sampling correction mechanism to hold the accuracy of the detection results.

As is shown in Figure 1, there are three layers in the airport apron monitoring hardware architecture used in this paper, including the monitoring layer, edge computing layer, and cloud computing layer. The monitoring layer and the edge computing layer are responsible for collecting video data and fast detection, respectively. The cloud computing layer plays the role of verifying and correcting the detection results.

There is an intelligent analysis framework combing the advantages of edge devices and cloud devices for airport apron surveillance video based on edge–cloud joint detection proposed due to the above architecture. First, an efficient lightweight object detection model based on YOLOv5s, named Edge-YOLO, is designed for the weak edge computing power. Compared with YOLOv5s, the model parameters and FLOPs are greatly reduced in the case of fewer drops in detection accuracy. Second, according to the characteristics of airport apron surveillance video, a T-frame motion and static separation algorithm is used to capture the motion information of airport apron objects in order to improve the real-time performance. Then, the video is cut into small segments with the same number of frames. The Edge-YOLO is called to detect the key frames. The detection results are derived to the non-key frames due to motion information inference, which combines the detection results of key frames and the extracted objects. Further, the full-size detection model, such as YOLOv5, is deployed in the cloud computing layer in order to check and correct the detection results obtained by the edge computing layer.
The main contributions of this paper can be summarized as follows:

- An object detection model Edge-YOLO with better real-time performance on embedded devices with weak computing power at the edge is proposed;
- A detection acceleration strategy to quickly generate non-keyframe detection results based on motion inference is proposed to further improve the detection speed;
- A cloud-based detection result verification and correction mechanism, which can bring the system to a level close to pure cloud detection by using a small amount of cloud computing power.

2. Related Work

2.1. CNN-Based Object Detectors

CNN-based object detection methods can be organized into two main categories, two-stage detection framework with region proposal and one-stage detection framework, i.e., region proposal free framework. In the two-stage framework, category-independent region proposals are generated from an image, CNN features are extracted from these regions, and then category-specific classifiers are used to determine the category labels of the proposals. RCNN series of methods is the representative of this category. Girshick proposed R-CNN in 2014 [12], which generates candidate regions by selective search algorithm [13], uses convolutional neural network for feature extraction, and uses SVM for object classification. Inspired by SPP-net [14], for the problem of feature extraction for candidate regions of R-CNN [12] which resulted in a large number of repeated redundant computations and a large amount of caches, Girshick improved R-CNN and proposed the Fast-RCNN [15], which input the entire image into the convolutional neural network to extract features, and then directly obtain the features of the candidate region from the feature map of the entire image according to the positional relationship, which greatly improves the detection speed. Subsequently, Ren proposed the faster Faster R-CNN [16] by replacing the time-consuming selective search [13] with a CNN-based Region Proposal Network (RPN).

The one-stage object detection method abandons the time-consuming candidate region generation process and directly generates the position coordinates and category probability of the object to be detected from the original image, which greatly improves the real-time performance of object detection. Typical representative methods of this branch include YOLO (You Only Look Once) [17] proposed by Redmon, CornerNet [18] proposed by Hei
and SSD [19] proposed by Liu. One-stage object detection methods are widely used in object detection tasks with high real-time requirements due to good real-time performance.

2.2. Edge Computing and Object Detection Model Lightweight

Edge computing is a way to fill the insufficiency of cloud computing. With the sharp increase in the number of terminal devices in the future, cloud computing will not be able to meet the needs of future network and computing costs. Fortunately, the development of embedded devices has enhanced the capabilities of edge computing to help cloud computing with data processing. Edge computing offers the advantages of low latency, low bandwidth requirements, and low cost. However, compared with cloud devices, edge devices are much weaker in terms of computing power and storage. Therefore, in order to obtain a lightweight model that can meet the hardware requirements of front-end embedded devices, the neural network model needs to be lightweight. For object detection tasks, there are currently three main ways to implement the deployment of models on edge embedded devices: lightweight detection model design, model pruning, and model quantization.

Lightweight Model Design: Lightweight detection model design is to apply a more efficient convolution method or a more efficient structure to reduce the computing power requirement of the object detection model. Aiming at real-time detection of surface defects, Zhou [20] proposed a reusable and high-efficiency Inception-based MobileNet-SSD method for surface defect inspection in industrial environment. Zhang [21] proposed a lightweight object detection based on the MobileNet v2, YOLOv4 algorithm, and attentional feature fusion to address underwater object detection.

Model Pruning: Model pruning refers to evaluating the importance of model weights according to a certain strategy and eliminating unimportant weights to reduce the amount of computation. To address the threat of drones intruding into high-security areas, Liu [22] pruned the convolutional channel and shortcut layer of YOLOv4 to develop thinner and shallower models. Aiming at the problems of low detection accuracy and inaccurate positioning accuracy of light-weight network in traffic sign recognition task, Wang [23] proposed an improved light-weight traffic sign recognition algorithm based on YOLOv4-Tiny.

Model Quantification: The essence of model quantification is to convert floating-point operation into integer fixed-point operation, which can drastically reduce model storage requirements. Zhang [5] proposed a data-free quantization method for the CNN-based remote sensing detection model by using 5-bit quantification. Guo [24] proposed a hybrid fixed point/binary deep neural network design methodology for object detection to achieve low-power consumption.

2.3. Edge–Cloud Cooperation

Edge–cloud Cooperation refers to the integration of cloud computing and edge computing. Edge computing provides users with low-latency, low-power services, while cloud computing is used to optimize the reasoning capabilities of edge computing. The edge–cloud collaboration method has been applied in various fields. Wang [25] proposed a smart surface inspection system using a faster R-CNN algorithm in the cloud–edge computing environment to solve automated surface inspection. Ye [25] used embedded devices to perform preliminary processing of the collected data, and then further analyzed the data through cloud computing to monitor the health of urban pipelines. Xu [26] proposed an edge–cloud cooperation framework to realize real-time intelligent analysis of the surveillance video of coal mine safety production. In this framework, cloud computing is used to process non-real-time and global tasks, and the edge part is responsible for real-time processing of local surveillance videos.
3. Proposed Methods

3.1. Lightweight Detection Model Edge-YOLO

As shown in Figure 2, the proposed lightweight detection model is called Edge-YOLO which is based on the combination of YOLOv5, ShuffleNetv2, and coord attention [2]. It is designed for the edge embedded device to detect objects from airport apron surveillance.

The proposed Edge-YOLO is built by simplifying the structure of YOLOv5 and the new network architecture reduces the amount of parameters by more than 90%. YOLOv5 [3] is an end-to-end method for object detection based on nonregional candidates, and it is the current state-of-the-art method in the field of one-stage object detection. Its network structure is mainly divided into two parts: backbone (feature extraction network) and head (detection head). There are four versions of YOLOv5, namely YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. The model size and accuracy of the four versions increase in turn, and the channel and layer control factors are used to select the appropriate size model according to the application scenario. In order to obtain a model that is easier to apply to resource-limited embedded devices, we choose YOLOv5s with the best real-time performance as the benchmark for optimization. Its network structure is mainly divided into two parts: backbone (feature extraction network) and head (detection head). In order to improve its detection speed, we chose to simplify the structure of both parts.

For backbone, we choose to use a light-weight image classification network for feature extraction. After comparing some light-weight image classification networks, we choose to use shuffleNetv2 [10] with the least amount of parameters and the best real-time performance for feature extraction. ShuffleNetv2 [10] is an image classification network designed for mobile devices and replaces the original backbone. This structure greatly reduces the computation of convolution operations by using group point wise convolutions. Moreover, a mechanism for information exchange between groups called ChannelShuffle, is used to achieve information fusion between groups. Compared with the original YOLOv5s feature extraction network, ShuffleNetv2 has better speed performance on edge devices. Table 1 shows the parameters and calculation complexity comparison of YOLOv5s-backbone and ShuffleNetV2 [10].
Table 1. Parameters and calculation complexity comparison of backbone.

| Model                  | Params | FLOPS  | Model Size (Half Precision) |
|------------------------|--------|--------|----------------------------|
| YOLOv5s backbone       | 3.80 M | 4.42 G | 13.1 MB                    |
| ShuffleNetv2 [10]      | 0.25 M | 0.31 G | 626 KB                     |

For the detection head part, the number of channels is cropped, since the number of categories to be detected in the airport apron scene is much less than that in the general scene. There are no more than 20 categories in the airport apron scene, while the detected categories are 80 in the COCO dataset. Only a quarter of channels is kept in order to get the highest cost-effectiveness in accuracy loss and compression ratio. Table 2 shows the parameters and calculation complexity comparison of YOLOv5s-head and Edge-YOLO-head.

Table 2. Parameters and calculation complexity comparison of head.

| Model                | Params | FLOPS  | Model Size (Half Precision) |
|----------------------|--------|--------|----------------------------|
| YOLOv5s head         | 3.00 M | 2.79 G | 5.78 MB                    |
| Edge-YOLO head       | 0.25 M | 0.21 G | 455 KB                     |

The reduction of the computational complexity of the feature extraction network may cause the weakening of the feature extraction capability. The coord attention [27] module is used to the extracted features, in order to make full use of the extracted features. This module is able to make the model pay attention to the parts that have a more significant impact on the final result. As shown in Figure 2, the coord attention [27] module decomposes channel attention into two 1D feature encoding processes. Since these two processes aggregate features along different directions, the long-range dependencies can be captured along one spatial direction, while precise location information can be preserved along the other. Then, the generated feature maps are separately encoded to form a pair of orientation-aware and position-sensitive feature maps, which is able to enhance the representation of objects of interest according to the complementary feature maps.

3.2. Edge Video Detection Acceleration Based on Motion Inference

Based on the characteristics of airport apron operations, there are two proposed acceleration strategies shown in Figure 3 in order to further accelerate edge detection and reduce the power consumption of edge devices.

The first strategy is to avoid detection of some idle periods. There is a statistic of work time and free time of a certain apron shown in Figure 4. The operation of the apron is not continuous. There is a long idle period between the two jobs that totals more than half of the total time. Obviously, it is not necessary to detect idle periods when no jobs are taking place. Therefore, it is important to distinguish whether the current airport apron is in operation and skip the detection of idle periods to save computing power and power consumption.

The video is divided into small segments with a duration of T (in order to avoid too long detection delay, T should take a small value. T takes 1 s, 2 s, 3 s, 4 s, 5 s in the experiments). According to whether object position changed in the segment, the judger decides whether the detection can be skipped (As shown in Figure 5). The Algorithm 1 is used to determine whether the position of an object has changed in a segment.

In Algorithm 1, the parameter $T_{area}$ represents the pixel value of the minimum object to be detected, which can be determined according to the specific detection task. The parameter $T_{pixel}$ represents the minimum pixel change of the object movement. We recommend a value of 10 to 20.
Algorithm 1: Determine whether exist object position changes in the video segment

**Input:** frame list $F = \{f_0, f_1, \ldots, f_k\}$; minimum object area $T_{area}$; minimum pixel change of object movement $T_{pixel}$

**Output:** the flag of whether the video clip has changed

1. convert frame $f_0$ to gray frame $f_0^{gray}$;
2. $f_S^{mask} = (f_0^{gray} < 0)$;
3. for $f_i$ in $F$:
   1. convert frame $f_i$ to gray frame $f_i^{gray}$;
   2. $f_S^{mask} = f_S^{mask} \cup (f_i^{gray} - f_0^{gray} > T_{pixel})$
   3. contours = find potential object contours in $f_S^{mask}$;
   4. for c in contours:
      1. if area(c) > $T_{area}$:
         1. return true;
   5. return false;

Figure 3. Edge detection acceleration strategy.

The second strategy is to employ motion inference to avoid frame-by-frame detection. The objects in the airport apron usually take regular slowly movements, due to the civil aviation relevant rules. Thus, there should be same object in most of near frames of the video. This means that not all frames need to be detected. Even a few frame detections might be able to get all objects. These frames are defined as key frames. Thus, the video could be divided into small segments with duration. The results for non-key frames are inferred from the detection of key frames, according to the motion information. The results of non-key frames are inferred from the detection of key frames, according to the motion information. There are two methods to extract the motion information of objects: matching based on IOU, and matching based on the improved T frame dynamic and static separation method.
The effect diagram of the way based on IOU matching is shown in Figure 6. For the detection results of two key frames, if the IOU of the two objects is greater than 0.2 and less than 0.9, we regard them as the same object. The IOU calculation formula of two objects $A$ and $B$ is given by Equation (1).

$$IOU(A, B) = \frac{A \cap B}{A \cup B}$$ (1)

**Figure 4.** Statistical chart of an airport apron operation.

**Figure 5.** Whether the detection can be skipped/Work or free.
Figure 6. Matching of moving objects is based on IOU.

However, the IOU-based matching method sometimes misses the relatively small objects on the airport apron such as staffs (as shown in Figure 7). In this case, the motion trajectory of the object may be incomplete. Thus, we proposed the T frame dynamic and static separation method based on the three-frame difference method to handle this problem as a supplement.

Figure 7. IOU matching failure (T = 2 s).

The three-frame difference method is one of the well-known methods to perform moving objects detection. The difference of frame \( i \) and frame \( i - 1 \), and the difference of frame \( i + 1 \) and frame \( i \) is used to obtain the motion mask layers \( D_1 \) and \( D_2 \), respectively.
The final motion mask layer $D$ of frame $i$ is calculated as $D_1$ AND $D_2$. The formula is expressed as Equations (2)–(4).

$$D_1(x, y) = \begin{cases} 1, & \text{if} |F_i(x, y) - F_{i-1}(x, y)| > T \\ 0, & \text{other} \end{cases}$$ (2)

$$D_2(x, y) = \begin{cases} 1, & \text{if} |F_{i+1}(x, y) - F_i(x, y)| > T \\ 0, & \text{other} \end{cases}$$ (3)

$$D(x, y) = D_1(x, y) \cap D_2(x, y)$$ (4)

In the above expression, $D(x, y)$ denotes the value of the position of the motion mask layer $(x, y)$, i.e., the pixel change interpolation of this point. $T$ denotes the minimum pixel change threshold where motion has occurred, and $F_i(x, y)$ denotes the pixel value at frame $i$ in $(x, y)$ position.

The three-frame difference method does not perform well for objects with similar internal pixels, and the effect is very dependent on the parameter $T_{pixel}$, which represents the minimum pixel change of the object movement. Thus, we proposed $T$ frames dynamic and static separation method to make up for its deficiencies. The static layers as the dynamic background of these $T$ frames are obtained according to Algorithm 2. In Algorithm 2, we recommend $T_{pixel}$ to be in the range of 10 to 20, and the motion layer can be obtained by $f_0$ subtracting the dynamic background, which is expressed as Equation (5).

\begin{algorithm}
\caption{Get dynamic background mask.}
\begin{algorithmic}
\State \textbf{Input:} frame list $F = \{f_0, f_1, \ldots, f_k\}$; minimum pixel change of object movement $T_{pixel}$;
\State \textbf{Output:} dynamic background mask;
\State convert frame $f_0$ to gray frame $f^{0}_{gray}$; $bg^{mask} = (f^{gray} > 0)$;
\For{$f_i$ in $F$}:
\State convert frame $f_i$ to gray frame $f^{gray}_i$;
\State $bg^{mask} = bg^{mask} \cap (f^{gray}_i - f^{gray}_0 < T_{pixel})$
\EndFor
\State \textbf{return} $bg^{mask}$
\end{algorithmic}
\end{algorithm}

$$D(x, y) = D(x, y) - D(x, y) \times bg^{mask}$$ (5)

The final motion information is inferred from the motion layer and the IOU matching mechanism. As shown in Figure 8, if the value of the motion mask is greater than a certain threshold, the objects involved to this motion are seen as the same. Thus, the detection results of the key frame can be passed to the non-key frames.

3.3. Cloud-Based Detection Results Verification Mechanism

Although the edge detection we proposed greatly improves the real-time detection, it inevitably brings the decrease of detection accuracy at the same time. However, through experimental analysis, we found that the decrease of detection accuracy is not for the whole video, but is more obvious in some busy segments. Based on this observation, we consider using a small amount of cloud computing power to detect these fragments, improving the overall detection accuracy with little impact on detection speed. Figure 9 shows the cloud calibration mechanism to improve the reliability of detection results.

For each group edge detection results $[EDR_i, EDR_{i+T}]$ (EDR: Edge Detection Result), the cloud detection model DC (Cloud Detector) re-detects its intermediate frame $F_{center}$ ($F_{i+T}/2$). CDR$_{center}$ (CDR: Cloud Detection Result) denotes the result of re-detection. Since $F_{center}$ is inferred from $EDR_i$, $EDR_{i+T}$, and motion layer, $F_{center}$ could show the correctness of the edge detection model and the motion layer. Thus, if $F_{center}$ is consistent with CDR$_{center}$, the set of detection results could be inferred as correct. Otherwise, it means the result of DE (Edge Detector) or DC is incorrect. Then we let DC re-detect the first frame $F_i$ and last frame $F_{i+T}$ of the group to obtain CDR$_i$, CDR$_{i+T}$ and let the motion inference strategy proposed in 3.2 re-generate detection result $[DR_i, DR_{i+T}]$ (DR: Detection Result)
using $\text{CDR}_i$, $\text{CDR}_{i+T}$, and motion layer. The set of detection results is inferred as correct if the generated intermediate frame detection results $\text{DR}_{\text{center}}$ is consistent with $\text{CDR}_{\text{center}}$. Otherwise, the extraction of the motion layer is inferred as incorrect, since the motion of the objects in the segment is not linear. In this case, let DC detects this group of video frames one by one, and use the obtained result $[\text{CDR}_i, \text{CDR}_{i+T}]$ as the final detection result of this video segment. Figure 10 shows the number of frames detected in the cloud for a segment of a T frame in different situations.

Figure 8. Example of object motion matching based on T frame dynamic and static separation ($T = 4$ s).

Figure 9. Cloud-based detection results verification and correct mechanism.
We can check its detailed introduction by [https://en.wikipedia.org/wiki/Raspberry_Pi](https://en.wikipedia.org/wiki/Raspberry_Pi) (accessed on 7 September 2022). The details of the experiment platform are shown in Table 3.

4. Discussion

4.1. Evaluation on Edge-YOLO

The verification of the proposed lightweight object detection model Edge-YOLO is taken on a set of surveillance images captured by surveillance videos in the real airport apron environment. There are 20,885 marked frames of outdoor airport apron surveillance video, in different lighting conditions and weather conditions. There are 10 categories in this set, such as aircraft, people, and each kind of vehicles in the apron. The training set is made up of 80% frames of the set, while the test set made up of the rest 20%. Figure 11 shows some frames of the set as examples.

![Figure 10](image1.png)  
**Figure 10.** The number of frames detected by the cloud in different situations (one T frame segment).

![Figure 11](image2.png)  
**Figure 11.** Some sample pictures from the dataset.

The experiments to test the accuracy and inference speed to analyze the size and calculation amount of the model is taken on the cloud server and the edge device (Raspberry Pi 4B) respectively. Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. We can check its detailed introduction by [https://en.wikipedia.org/wiki/Raspberry_Pi](https://en.wikipedia.org/wiki/Raspberry_Pi) (accessed on 7 September 2022). The details of the experiment platform are shown in Table 3.
The related experimental results are shown in Tables 4 and 5. Compared with the baseline model YOLOv5s, the proposed lightweight model Edge-YOLO has achieved considerable improvements in detection speed, model size, and computational complexity. The accuracy loss of Edge-YOLO is no more than 5% and the loss will be further reduced by using Edge-YOLO-CA which is added the coord attention [27] module. Compared with the large-scale network SSD [19], the accuracy and detection speed of our proposed model have obvious advantages in the airport apron scene. Compared with the lighter models MobileNet-SSD [20] and YOLOv4-tiny [23], our proposed model achieves a large improvement in detection speed with close accuracy. Subsequently, we also proposed some strategies to further improve the detection speed of the model for airport apron operation videos and make up for the loss of detection accuracy.

### Table 3. Experiment platform.

| Name                  | CPU                        | GPU               |
|-----------------------|----------------------------|-------------------|
| Cloud Server          | Intel Core i5-10400F        | NVIDIA Tesla V100 |
| Edge Embedded Device  | ARM Cortex-A72@1.5 GHz     |                   |
|                       | 16 GB                      |                   |

### Table 4. Model accuracy performance with state of art algorithms.

| Methods          | Precision | Recall | mAP@0.5 (%) | mAP (@50:5:95) (%) |
|------------------|-----------|--------|-------------|--------------------|
| SSD [19]         | 0.927     | 0.915  | 90.15       | 72.3               |
| YOLOv5s [3]      | 0.946     | 0.925  | 94.9        | 78.6               |
| MobileNet-SSD [20]| 0.896    | 0.878  | 88.6        | 69.4               |
| YOLOv4-tiny [23] | 0.898     | 0.877  | 92.2        | 73.9               |
| Edge-YOLO        | 0.931     | 0.884  | 90.79       | 72.5               |
| Edge-YOLO-CA     | 0.94      | 0.899  | 91.37       | 73.1               |

### Table 5. FPS on raspberry 4B and model performance comparison.

| Indicator | SSD [19] | YOLOv5s [3] | MobileNet-SSD [20] | YOLOv4 Tiny [23] | Edge-YOLO | Edge-YOLO-CA |
|-----------|----------|-------------|--------------------|-----------------|-----------|--------------|
| FPS       | 0.48     | 1.25        | 1.84               | 1.13            | 4.29      | 4.21         |
| Model Size| 94.7 MB  | 14 MB       | 22.1 MB            | 22.5 MB         | 1.18 M    | 1.21 M       |
| GFLOPS    | 640 × 640| 131.31      | 7.58               | 4.87            | 8.1       | 0.79         |
|           | 416 × 416| 58.75       | 3.20               | 2.11            | 3.42      | 0.34         |
|           | 320 × 320| 34.76       | 1.90               | 1.25            | 2.02      | 0.20         |

#### 4.2. Evaluation on Edge Detect Accelerate Strategy

According to the proposed edge detection strategy, the detection module does not work during the certain idle periods. There are 30 periods of 5-min non-operating apron monitoring video to validate the strategy in this experiment. The acceleration of each period is shown in Figure 12. The experimental results show that this strategy can effectively avoiding idle detection. The failure of this strategy is when the detection is active because of the staff or vehicles passing through the operation area of the apron.

There is another experiment to validate the acceleration strategy 2, with two apron operation videos from Obihiro Airport in Japan and Guiyang Airport in China. As shown in Table 6, the duration of the two videos is 49 min 28 s and 57 min 35 s, respectively. Since the video frame rates of these two videos are 30 and 25, respectively, there are 89,029 frames in Obihiro Airport video and 86,395 frames in Guiyang Airport video. Both of the videos record the whole process from flight taxiing into the parking bay to departure.
When T is greater than 2 FPS, the loss of mAP increases rapidly. Therefore, the parameter T could be estimated by testing on a video segment. Thus, this acceleration strategy is T should be set as 2 FPS for these two videos. Obviously, for other videos, the parameter power. The Figure 13 shows the mAP and time cost of Edge-YOLO-CA with acceleration strategy can perform real-time detection on edge devices with weak computing power. The Figure 13 shows the mAP and time cost of Edge-YOLO-CA with acceleration strategy on raspberry 4B. According to the curves in Figure 13, the mAP remains high. When T is greater than 2 FPS, the loss of mAP increases rapidly. Therefore, the parameter T should be set as 2 FPS for these two videos. Obviously, for other videos, the parameter T could be estimated by testing on a video segment. Thus, this acceleration strategy is effective, which is not limited to the experiment videos.

Tables 7 and 8 show accuracy and time cost of YOLOv5s, Edge-YOLO, and Edge-YOLO with acceleration strategy on these two videos. Experiments show that the proposed acceleration strategy can perform real-time detection on edge devices with weak computing power. The Figure 13 shows the mAP and time cost of Edge-YOLO-CA with acceleration strategy on raspberry 4B. According to the curves in Figure 13, the mAP remains high. When T is greater than 2 FPS, the loss of mAP increases rapidly. Therefore, the parameter T should be set as 2 FPS for these two videos. Obviously, for other videos, the parameter T could be estimated by testing on a video segment. Thus, this acceleration strategy is effective, which is not limited to the experiment videos.

Table 6. Video info.

| Video Id | Country | Airport                                      | Video Duration | Frame Rate/FPS | Total Frames |
|----------|---------|----------------------------------------------|----------------|----------------|--------------|
| 1        | China   | Guiyang Longdongbao International Airport    | 00:57:35       | 25             | 86,395       |
| 2        | Japan   | Tokachi Obihiro Airport                      | 00:49:28       | 30             | 89,029       |

Table 7. The mAP and time consumption of Edge-YOLO-CA on raspberry 4B.

| Model            | Video Id | 1           | 2           |
|------------------|----------|-------------|-------------|
| YOLOv5s [3]      | mAP@0.5 (%) | 93.58      | 92.79       |
|                  | time consumption | 19 h 27 min 35 s | 19 h 53 min 11 s |
| Edge-YOLO-CA     | mAP@0.5 (%) | 90.89      | 89.94       |
|                  | time consumption | 5 h 55 min 23 s | 5 h 44 min 50 s |
Table 8. The mAP and time consumption of Edge-YOLO-CA with acceleration strategy on raspberry 4B.

| Video Id | T          | 1FPS | 2FPS | 3FPS | 4FPS | 5FPS |
|----------|------------|------|------|------|------|------|
| 1        | mAP@0.5 (%)| 90.07| 89.78| 88.43| 87.35| 85.03|
|          | time consumption | 35 min 46 s | 23 min 28 s | 22 min 37 s | 22 min 27 s | 22 min 18 s |
| 2        | mAP@0.5 (%)| 88.49| 88.21| 86.51| 84.53| 82.66|
|          | time consumption | 32 min 49 s | 21 min 47 s | 18 min 34 s | 17 min 35 s | 17 min 29 s |

Figure 13. The mAP and time cost of Edge-YOLO-CA with acceleration strategy on raspberry 4B.

4.3. Evaluation on Whole Edge-Cloud System

Table 9 shows the experiment results to verify the effectiveness of the cloud-based detection results verification mechanism. In this experiment, the Edge-YOLO-CA with acceleration strategy generates detection results of surveillance video on edge devices, while the YOLOv5s deployed on cloud devices verifies and corrects the detection results due to the cloud-based detection results verification mechanism mentioned in Section 3.3. As shown in Table 9, with only checking less than one-tenth of frames, the edge detection results can be corrected to close to the cloud detection results.

Table 9. Performance of Edge-YOLO-CA with acceleration strategy and edge-cloud cooperation on Edge-Cloud System.

| Video Id | T          | 1FPS | 2FPS | 3FPS | 4FPS | 5FPS |
|----------|------------|------|------|------|------|------|
| 1        | mAP@0.5 (%)| 93.47| 93.43| 93.45| 93.39| 93.51|
|          | cloud detect num | 7973 | 8797 | 14,764 | 18,874 | 21,564 |
|          | cloud detect | 8.96% | 9.88% | 16.58% | 21.20% | 24.22% |
| 2        | mAP@0.5 (%)| 92.57| 92.49| 92.61| 92.54| 92.65|
|          | cloud detect num | 8361 | 9748 | 15,732 | 19,746 | 22,893 |
|          | cloud detect | 9.68% | 11.28% | 18.21% | 22.86% | 26.50% |

5. Conclusions

In this article, a real-time object detection solution based on edge cloud system for airport apron operation surveillance video is proposed, which includes lightweight detection model Edge-YOLO, edge video detection acceleration strategy, and cloud-based detection results verification mechanism. The lightweight detection model is based on YOLOv5s. By replacing the backbone with ShuffleNetv2, cropping the channel of the detect head, and adding coord attention, the model size of Edge-YOLO is reduced by more than 90% compared with YOLOv5s, greatly reducing the requirements for deployment equipment. The detection speed of Edge-YOLO on Raspberry Pi 4B is increased by 3.37 times, and its accuracy loss does not exceed 5% by testing on a dataset composed of more than 20,000 real-world airport apron scenes. Edge video detection acceleration
realizes further detection acceleration of Edge-YOLO on airport surveillance video by motion information inference by segmenting the video, then using IOU matching and improved T-frame difference method to extract the motion information of objects, and finally combining the detection results of Edge-YOLO on key frames to quickly generate the detection results of the entire segment. The cloud-based detection results verification mechanism uses a non-lightweight detection model with higher accuracy to verify and correct the results generated by the edge to improve the overall detection accuracy and reduce the proportion of cloud intervention through a multi-level intervention mechanism. Their feasibility is verified by experiments on two real airport apron surveillance videos that are fully annotated and contain the complete operation process. In the future, we will focus on designing more flexible edge–cloud cooperation strategies, and strengthen the versatility of the method and extend it to a wide range of video surveillance scenarios.

Author Contributions: J.L. proposed the network architecture design and the framework of Edge-YOLO. Z.L. proposed the strategy of video detection acceleration and edge–cloud Cooperation. Z.L. and J.L. performed the experiments, analyzed the experimental data, and wrote the article. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds for Central Universities of the Civil Aviation University of China, grant number 3122021088.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank the anonymous reviewers and the editor-in-chief for their comments to improve the article. Thanks also to the data provider. We thank all the people involved in the study.

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

References
1. Liu, C.; Chen, Y.R.; Chen, F.H.; Zhu, P.; Chen, L.Y. Sliding window change point detection based dynamic network model inference framework for airport ground service process. Knowl.-Based Syst. 2022, 238, 107701. [CrossRef]
2. Ajmal, S.; Jo, K. Deep Atrous Spatial Features-Based Supervised Foreground Detection Algorithm for Industrial Surveillance Systems. IEEE Trans. Ind. Inform. 2021, 17, 4818–4826.
3. Xu, Z.; Huang, X.; Huang, Y.; Sun, H.; Fan, F. A Real-Time Zanthoxylum Target Detection Method for an Intelligent Picking Robot under a Complex Background, Based on an Improved YOLOv5s Architecture. Sensors 2022, 22, 682. [CrossRef] [PubMed]
4. Chen, W.; Lu, S.; Liu, B.; Chen, M.; Li, G.; Qian, T. CitrusYOLO: A Algorithm for Citrus Detection under Orchard Environment Based on YOLOv4. Multimed. Tools Appl. 2022, 8, 1–27. [CrossRef]
5. Zhang, R.; Jiang, X.; An, J.; Cui, T. Data-Free Low-Bit Quantization for Remote Sensing Object Detection. IEEE Geosci. Remote Sens. Lett. 2022, 19, 1–5. [CrossRef]
6. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv 2017, arXiv:1704.04861.
7. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
8. Howard, A.; Sandler, M.; Chu, G.; Chen, L.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 20–26 October 2019.
9. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
10. Ma, N.; Zhang, X.; Zheng, H.; Sun, J. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
11. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
12. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 24–27 June 2014.
13. Van De Sande, K.E.A.; Uijlings, J.R.R.; Gevers, T.; Smeulders, A.W.M. Segmentation as selective search for object recognition. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Trans. Pattern Anal. 2015*, 37, 1904–1916. [CrossRef] [PubMed]

15. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Boston, MA, USA, 7–13 December 2015.

16. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. 2017*, 39, 1137–1149. [CrossRef] [PubMed]

17. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.

18. He, L.; Jia, D. CornerNet: Detecting Objects as Paired Keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.

19. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the European conference on computer vision, Amsterdam, The Netherlands, 8–16 October 2018.

20. Zhou, J.; Zhao, W.; Guo, L.; Xu, X.; Xie, G. Real Time Detection of Surface Defects with Inception-Based MobileNet-SSD Detection Network. In Proceedings of the Advances in Brain Inspired Cognitive Systems, Guangzhou, China, 13–14 July 2019.

21. Zhang, M.; Xu, S.; Song, W.; He, Q.; Wei, Q. Lightweight Underwater Object Detection Based on YOLO v4 and Multi-Scale Attentional Feature Fusion. *Remote Sens. 2021*, 13, 4706. [CrossRef]

22. Liu, H.; Fan, K.; Ouyang, Q.; Li, N. Real-Time Small Drones Detection Based on Pruned YOLOv4. *Sensors 2022*, 21, 3374. [CrossRef] [PubMed]

23. Wang, L.; Zhou, K.; Chu, A.; Wang, G.; Wang, L. An Improved Light-weight Traffic Sign Recognition Algorithm Based on YOLOv4-Tiny. *IEEE Access 2021*, 9, 124963–124971. [CrossRef]

24. Guo, J.; Tsai, C.; Zeng, J.; Peng, S.; Chang, E. Hybrid Fixed-Point/Binary Deep Neural Network Design Methodology for Low-Power Object Detection. *IEEE J. Emerg. Sel. Top. Circuits Syst. 2020*, 10, 388–400. [CrossRef]

25. Wang, Y.; Liu, M.; Zheng, P.; Yang, H.; Zou, J. A smart surface inspection system using faster R-CNN in cloud-edge computing environment. *Adv. Eng. Inform. 2020*, 43, 101037. [CrossRef]

26. Xu, Z.; Li, J.; Zhang, M. A Surveillance Video Real-Time Analysis System Based on Edge-Cloud and FL-YOLO Cooperation in Coal Mine. *IEEE Access 2021*, 9, 68482–68497. [CrossRef]

27. Hou, Q.; Zhou, D.; Feng, J. Coordinate Attention for Efficient Mobile Network Design. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021.