applied sciences

MDPI

Article

Traffic Monitoring from the Perspective of an Unmanned Aerial Vehicle

Ondrej Kainz *, Matúš Dopiriak, Miroslav Michalko, František Jakab and Ivana Nováková

Department of Computer and Informatics, Technical University of Kosice, 040 01 Kosice, Slovakia
* Correspondence: ondrej.kainz@tuke.sk

Abstract: The paper is focused on the development of the experimental web-based solution for image processing from the perspective of an Unmanned Aerial Vehicle (UAV). Specifically, the research is carried out as part of the broader study on drone utilization in traffic at the Technical University of Kosice. This contribution explores the possibility of using the UAV as a tool to detect the temporal state of the traffic in multiple locations. Road traffic analysis is enabled through the detection of vehicles from the user-defined region of interest (ROI). Its content then serves as the input for motion detection, followed by the detection of vehicles using the YOLOv4 model. Detection of other types of objects is possible, thus making the system more universal. The vehicle is tracked after recognition in two consecutive frames. The tracking algorithm is based on the calculation of the Euclidean distance and the intersection of the rectangles. The experimental verification yields lower hardware requirements for CPU and GPU by about two FPS when using optimization techniques, such as ROI or reference dimensions of objects. The accuracy of detection and the subsequent tracking of cars reaches almost 100% while providing accurate trajectory determination.

Keywords: computer vision; deep learning; object tracking; unmanned aerial vehicle; YOLO

Citation: Kainz, O.; Dopiriak, M.; Michalko, M.; Jakab, F.; Nováková, I. Traffic Monitoring from the Perspective of an Unmanned Aerial Vehicle. *Appl. Sci.* **2022**, *12*, 7966. https://doi.org/10.3390/app12167966

Academic Editor: Feng Guo

Received: 24 June 2022
Accepted: 5 August 2022
Published: 9 August 2022

1. Introduction

Monitoring of traffic is usually done using conventional methods by traffic cameras. This approach requires static fixed cameras and existing infrastructure, which may not only be cost-ineffective but also ineffective in situations where temporary monitoring is required. With the recent advancements in technology, Unmanned Aerial Vehicles (UAVs) may provide a solution to this problem.

UAVs have become popular in recent years. Such devices bid a wide range of applications that include aerial photography, cinematography, express shipping and delivery, and monitoring of surroundings. Nowadays, applications of UAVs tend to move to the next level and make them intelligent and autonomous as well. The field of computer vision is essential for the utilization of UAVs in cities to gain beneficial information from the real world.

The purpose of this research is to investigate the possible situation in traffic monitoring using UAVs and design a system that provides information on the number of vehicles, their visual representation, and estimates of their direction. Specifically, the research described in this paper proposes and experimentally implements a web-based service for traffic environment monitoring, specifically its flow and load. Once information from the video is extracted, we will be able to estimate the number of cars that utilize the road in specific time intervals and also their direction. Such data can be used as the input for the *Slovak Road Administration*; until now, such a process is being done manually.

The proposed web application offers a variety of personalized services that include a user profile, in which all data are stored, device-independent and free access, and a minimum hardware load of the user's device. The same holds for the UAVs; they are becoming common to use mobile web applications (e.g., online visualization of data in

research by Chodorek et al. [1]), solutions for the processing of imagery from UAVs in research by Guimaraes [2] of their employment to ensure safety in transportation (research by Shvetsova, Shvetsov [3], or by Lyovin et al. [4]). The experimental web-based solution may automatically estimate a specific category, which may include humans, cars, or other objects. The subsequent step is a visualization of the detection results accessible to the user. The primary input element to the proposed system is a video in a specific format captured by a UAV. Input to the system may also be from other sources, e.g., surveillance cameras. The principal idea is to use a portable surveillance device to monitor the traffic from a certain view and obtain as much data from the real environment as possible.

The purpose of this research is to propose an experimental solution for the monitoring of traffic from the perspective of a UAV. In the sections below, the algorithms are analyzed for object detection, and, based on this, the design of the experimental software solution is created, which is then experimentally verified. The contribution of this work may be summarized as follows:

- Assessment of the UAV's potential to monitor the temporal state of the traffic in multiple locations and determine the suitable scenarios;
- Design and implement the experimental software solution for the detection of vehicles, estimation of their number, and direction;
- Verify the algorithms for video analysis following sample scenarios.

The idea behind the monitoring of the traffic using UAV may bring several advantages compared to a static traffic monitoring solution. Rather, complex installation is not required, monitoring of traffic from multiple locations in specific day intervals may be achieved, or variable environments that contain vehicles may be monitored at any time during the day.

The utilization of UAVs may also bring difficulties, starting with changing weather conditions. Strong winds or rain may not only deviate from the tracking of the vehicles but render it completely unusable. Taking a shot in the fog is the next variable that may deviate from the results, and, if providing strong fog is present, almost no detection would be possible. In addition, depending on the type of UAV, the camera should have night vision capabilities. Another thing to consider is battery life: with a static camera, such things are no issue; inter alia, the real-time analysis of video input is more difficult with UAVs since such processing requires a device with sufficient hardware power. The comparison between UAVs, static traffic cameras, and dash cameras used in vehicles is presented in Table 1.

Table 1. Comparison of monitoring mechanisms: UAV, static camera, dash camera.

	Advantages	**Disadvantages**
UAVs	Variable locations/setups. Prior infrastructure not required. Traffic overview. Variable coverage. Replacement of multiple static cameras. Enhanced extensions (e.g., possible 3D reconstruction). Rapidly developing over the last decade.	Battery life. Limited resources for operation (all-in-one solutions). Weather impact performance. Advanced operation (control). Remote users cannot access stream.
Static traffic cameras	Traffic overview. Weatherproof. Easier operation. Always available stream. Remote connection for users. External power supply.	Fixed location/setup. Required pre-installed infrastructure. Fixed coverage. Multiple cameras are usually required.
Dash cameras	Cost. Easy operation. Plug-and-play. External power supply.	Limited mobility. Limited traffic overview. Fixed view. Remote users cannot access stream.

2. Object Detection Algorithms

Object detectors are generally divided into single-stage (e.g., YOLO, SSD) and two-stage (e.g., R-CNN, Mask R-CNN) [5]. Two-stage detectors generate region proposals that

may be classified into a particular category. The most significant feature is the detection accuracy, which reaches higher results. In contrast, single-stage detectors solve this task as a regression problem, and all computations are performed only in one network. The weakness of this approach is low accuracy. On the other hand, the performance is the most significant advantage.

2.1. You Only Look Once

YOLO belongs to the most used deep learning algorithms due to its compromise between speed and accuracy. One of the key features is the ability to run on CPU and a variety of its implementations that reach remarkable results on non-GPU machines.

The initial version realizes object detection as a regression problem that uses only one CNN consisting of 24 layers to predict bounding boxes. Predictions constitute tensors composed of the input image in the form:

$$S \times S(B \times 5 + C), \tag{1}$$

where $S \times S$ is a grid, B are bounding boxes consisting of 5 predictions, and C is class probabilities. Each cell in the grid is supposed to detect the searched object. Class probability is computed as well. Figure 1 depicts the crucial parts of the detection process. The model was pretrained on ImageNet and evaluated on the PASCAL VOC dataset. The prediction is one of the limitations of a single object with multiple assigned bounding boxes. This case can be partially solved by non-maximal suppression. The model struggles the most with small and clustered objects. Experimental verification has shown that real-time detection is feasible because the model runs at a speed of 45 FPS. The modified version, so-called Fast YOLO, contains CNN consisting of nine layers that reach a performance of 150 FPS.

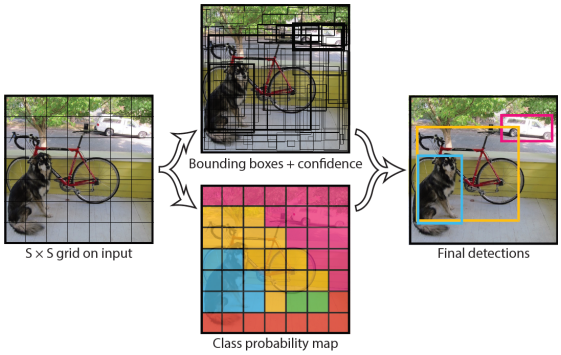


Figure 1. The YOLO model. Source of the image: [6].

In 2020, YOLOv4 [7] was presented and compared with its predecessor. The main difference is the use of CSPDarknet53 as a base network and the integration of several features that include the CBM module, which includes *Conv* convolution layer, *BN* batch normalization layer, and *Mish* activation [8], which can be described as Equation (2):

$$f(x) = x \times tanh(softplus(x)) \tag{2}$$

where

$$softplus(x) = ln(1 + e^x). \tag{3}$$

CSPDarknet53 also includes five CSP modules, where each reduces the image in half. YOLOv4 implements a prediction network (*YOLOhead*) and feature pyramid (*SPP* spatial pyramid pooling and *PANet* path aggregation network). The structure of YOLOv4 is shown in Figure 2. Experiments evaluated on the COCO dataset have shown that AP improves by 10% and performance by 12% with respect to the YOLOv3 version.

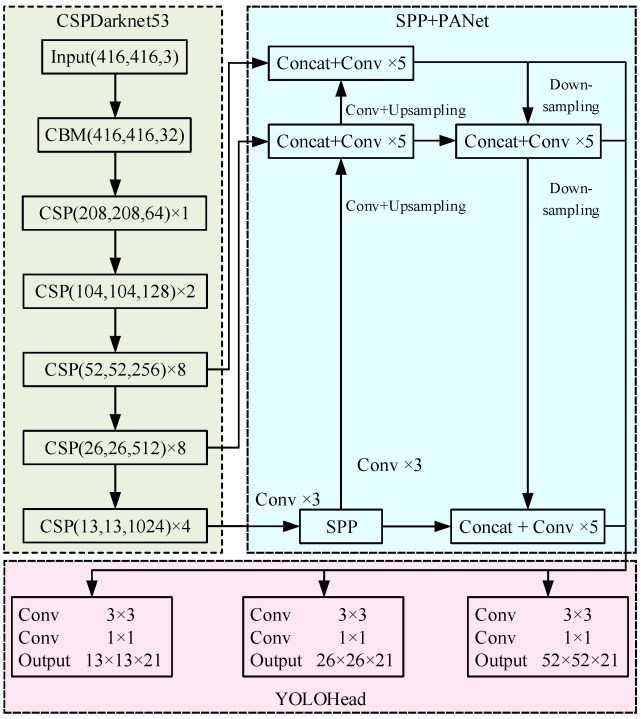


Figure 2. The structure of YOLOv4, source of the image: [8].

Follow Ref. [5], which proposes a new algorithm called SlimYOLOv3, based on the YOLOv3 version suggested for real-time object detection on a UAV platform. The most suitable approach is the utilization of channel pruning of convolutional layers to a particular ratio to simplify the deep model and remove its unnecessary connections. Finally, the algorithm is fine-tuned for performance optimization. SlimYOLOv3 evaluates on the VisDron2018-Det dataset. Experimental verification demonstrated that the proposed algorithm attains comparable accuracy mAP of 25.8% with respect to YOLOv3 mAP of 26.4%. On the other side, performance is nearly two times faster, and the amount of BFLOPs produces one-third against YOLOv3.

Authors [9] make a new dataset based on aerial imagery for vehicle detection that contains six thousand images. Verification of the dataset performs on several deep learning models that include Modified Faster R-CNN, YOLOv4, and MobileNetv3. The highest mAP of 96.91% reaches YOLOv4 and is eligible for vehicle detection under different conditions.

The modified version Scale-YOLOv4 was adopted by Chang et al. [10], for the detection of obstacles. Authors report accurate results when utilized in an unmanned ground vehicle. More closely related to our research is the recognition and detection of traffic elements by Huang et al. [11]. Authors, in this case, used YOLOv4 with a mAP value above 90%.

2.2. Single Shot Multibox Detector

The research study [12] introduces another single-stage detector, which attempts to solve the disproportion between speed and accuracy in contrast to YOLO and Faster R-CNN. The development of SSD-based models that take into account hardware is suitable for a CPU run. Ref. [13] utilizes an SSD-based model along with Mobilenetv2 and Feature Pyramid Network. Experimental results showed that the proposed model achieves 75.6% mAP and 21 FPS trained and tested on the PASCAL VOC dataset. Detection accuracy improves by 3.2% against Mobilenet-SSD. Verification confirmed that the presented model detects various categories of objects at various distances and conditions that contain a low amount of light.

Authors propose Feature-Fused SSD [14] adapted to detect small objects. The model is based on SSD that utilizes VGG16 as a base network. The network detects small entities in

shallow layers and larger ones in deeper layers. Experimental verification was performed with the utilization of the PASCAL VOC2007 dataset during the training phase. Both models experienced a slight decrease in performance, which represents a reduction of about 7 to 10 FPS, in contrast to SSD300, which reached 50 FPS. On the other hand, the concatenation model and element-sum model are characterized by higher accuracy of 78.8 mAP and 78.9 mAP, respectively.

Detection using imaginary from the UAV was conducted by Sivakuman et al. [15]. Authors [16] utilized SSD, with Inception v2 architecture, and compared it with Faster R-CNN, yielding similar detection performance.

2.3. RetinaNet

Lin et al. [17] propose another one-stage detector called RetinaNet. This model solves the imbalance problem directly connected to one-stage detectors such as YOLO or SSD. The issue is based on the generation of too many locations, in which an object could be present.

Li and Ren [18] analyzed the performance of the RetinaNet model, which is decomposed into several parts marked with an appropriate count of FLOPs. Experimental results indicated that the proposed technique, called LW-RetinaNet-v1, is characterized by higher mAP by 0.1% along with the drop of GFLOPs of 1.15 times with respect to the original value of 156 GFLOPs acquired from RetinaNet-800.

Ahmad et al. [19] aim at RetinaNet optimization to improve the detection accuracy of tiny objects. The research is based on anchor optimization that utilizes the Crow search algorithm to find ratios and scales during the training phase. Experimental verification on the VisDrone dataset has shown the progress of 4 mAP in contrast to standard RetinaNet characterized by fixed anchors.

Psiroukis et al. [20] included RetinaNet in their research on the estimation of the maturity levels of broccoli crops from UAV input. Based on the research, RetinaNet demonstrates lower performance than Faster R-CNN, which is highlighted as the best, along with CenterNet.

2.4. Region-Based CNN

Region-based Convolutional Neural Network (R-CNN) algorithms are characterized by high accuracy because they use region proposals to perform object detection. This approach requires special hardware allowing parallel computation to reduce detection time.

The research study [21] introduces Faster R-CNN that continues the work of its predecessors. This algorithm utilizes the Region Proposal Network to generate region proposals to Fast R-CNN. Selective search is removed because it causes network congestion. Experimental results showed a mAP of 75.9% on the PASCAL VOC 2012 test set and mAP of 78.8% on the PASCAL VOC 2007 test set with a test-time speed of around 200 ms per image. The highest reported performance was 17 FPS, and the lowest was 5 FPS.

The research study [22] evaluates Faster R-CNN based on records from UAV. The model is trained and evaluated on the custom dataset, which includes traffic videos captured at different heights in the range of 100 m to 150 m. Experiments have shown completeness of 96.40% and correctness of 98.26%. Conditions based on the amount of illumination decreased the completeness to 94.16%. This value is still sufficient for the purposes. Results of the model are not affected by rotation changes or variable count of detected objects.

Object detection from UAV imagery using the Multi-Stream approach was also done by Avola et al. [23], with positive results for different heights of the flight. Overall performance for Multi-Stream yielded mAP of 97.3% and 95.6% for Faster R-CNN.

2.5. Mask R-CNN

Mask R-CNN is deemed to be an extension of Faster R-CNN, which shares several features that include RPN to generate region proposals, etc. The main dissimilarity is the implementation of instance segmentation and calculation of mask. It is the additional

output of the network to detect objects at a pixel level. Such a technique can distinguish among objects, even overlapping objects of the same category.

One of the possible applications of Mask R-CNN can be utilization in cities, in which the count of vehicles and issues associated with it grows every year. In 2019, Piedad et al. [24] presented a system able to count vehicles based on the mentioned model. Four categories of interest include trucks, buses, cars, and motorbikes. This approach decreases the amount of input data to the resulting model. The system recognized several issues connected to overlapping objects, which distorted the detection process. The testing phase showed 97.62% accuracy during car detection.

2.6. Evaluation of Object Detection Algorithms

The research study by Wang et al. [25] makes a comparative analysis of the algorithms that contain Faster R-CNN, YOLOv3, SSD, and RetinaNet. Experiments solve vehicle detection tasks on the KITTI dataset and set the parameters for evaluation. Experimental results have demonstrated that the two-stage detector Faster R-CNN has higher accuracy of 81.09% AP, but the performance of 0.68 FPS is still lower in contrast to single-stage. On the other hand, RetinaNet is considered to have excellent accuracy, which overcomes the preceding algorithm with 89.83% accuracy and optimized performance of 3.59 FPS. In this case, YOLOv3 and SSD belong to the models, which have the worst performance and accuracy parameters. Nevertheless, SSD's performance of 14.27 FPS is suitable for devices with limited hardware. The study also gives increased attention to the variability of possible environments, which extensively affect correct detection.

The single-stage detectors are suitable for execution on CPUs, mainly SSD and YOLO. Two-stage detectors are adapted to run on GPUs. Faster R-CNN is adapted for a wide range of variable conditions and achieves higher accuracy. However, RetinaNet and newer YOLO versions are real competitors against the two-stage detector.

3. Materials and Methods

In this section, we describe the process of selecting a suitable situation for traffic monitoring using UAVs. Next, the testing of the detection algorithms is described and evaluated. The proposed approach for efficient localization and tracking of the object, along with the overall functionality of the proposed experimental system and related video analysis is presented.

3.1. Selection of Suitable Situations

Surveillance cameras can offer quality videos from traffic for subsequent analysis. It is important to note that there are many roads in the cities but also between them. It follows that it is not possible to install cameras for analytical purposes in all places. It is the most significant disadvantage of such cameras. Thus, it is important to search for an alternative that has to meet several requirements; this includes mobility in the environment, simple control, costs, and efforts associated with the operation. One such option are UAVs, which are adapted to such type of work; they are mobile and make meaningful records for analysis.

Four specific situations that capture cars in traffic were considered and are shown in Figure 3. The view above the road level (Figure 3a) represents a front view, which can be recorded by a UAV. Such a view provides a relatively detailed frame of the car from the front side. Bird's eye view from the side (Figure 3b) represents a side view in which cars are too small to be detected with a high confidence score. However, such a view provides enough information about a particular car. Bird's eye view (Figure 3c) represents a view from above, in which cars are too small, and only rectangles can be recognized. The last picture (Figure 3d) poses an illustration of a video captured from a road level, which is not suitable for UAVs. However, it allows an effortless way to read a number plate. The distance from the camera is sufficient as well. Each situation has its pros and cons. The most significant features include a view from the front side or side view and a sufficient

distance from the camera suitable for UAVs. It follows that the proposed experimental software service will conduct the best analysis for the views above the road level.

(**a**) (**b**)

(**c**) (**d**)

Figure 3. Candidate situations. (**a**) View above the road level; (**b**) bird's eye view from the side; (**c**) bird's eye view; (**d**) road level view. Source of the input video: [26].

The UAV should provide stable flight at a constant height in a vertical plane and stay in one designated spot when capturing the videos. Thus, the UAV should focus on a specific scene, while not being in the intentional movement, i.e., act as a static camera. The benefit, in this case, is that the UAV may be navigated to specific locations automatically and start the recording. However, there are possible challenges, mainly the severe weather conditions, e.g., wind disturbances or fog, which may introduce unintentional movements and following errors in the tracking within the region of interest (ROI). An approach to overcome this challenge was introduced by Jayaweera and Hanoun [27] with very promising results.

3.2. Testing of Object Detection Algorithms

Bearing in mind the proposed approach and various scenarios, a comparison of suitable object detection algorithms was conducted. Each frame is to be resized to 1280 pixels in width and 720 pixels in height. The model has to consider the accuracy of the bounding box, confidence score, count of bounding boxes, and performance.

Following testing on the CPU, YOLOv3-tiny is not suitable for expected scenarios due to zero detected cars in the first scenario despite the highest performance up to 30.46 FPS in the third scenario. Other models are similar to each other, but YOLOv4 shows the highest confidence score at almost 99%, and precisely located bounding boxes that guarantee stability and successful tracking. SSD MobileNet model has potential due to its well-demonstrated results, yet tests have proved to have an insufficient ability to accurately locate a real object.

However, further testing on GPU shows that the RetinaNet model has a low performance of around 4 FPS and a confidence score of around 85%. Faster R-CNN ResNet-50 FPN surpasses other models of the same type in the majority of criteria. However, the performance around 4 FPS is considerably slow. This weakness can be largely eliminated via motion detection. YOLOv4 surpasses all models, especially in terms of performance up to 18 FPS, and stability of detection in terms of the real location of bounding boxes, which are their dominant features. The performance of YOLOv4 is at least three times higher in contrast to the top Faster R-CNN model. In the research, the YOLOv4 algorithm is used for the detection of vehicles.

3.3. Object Tracking Algorithm

It is indispensable to implement a lightweight, straightforward, and effective tracking algorithm. The main reason is that object detection is a computationally expensive part of the entire process. One of the straightforward algorithms to track detected objects is to calculate the distance between center points of the current and previous rectangle or intersection between them. Direction detection is the product of tracking information. It is based on analyzing trajectory and calculation of the difference between center points.

One of the main aspects is to maintain the information of the vehicle that moves in the ROI and make sure that it is the same and not another vehicle. The entire process is based on object detection. Therefore, this operation has to be stable in terms of confidence score. The feature should not have large deviations. Many algorithms currently exist. It holds true that the better the algorithm, the more computationally demanding it is. One of the priorities we consider is computational ease. Thus, object detection has the highest priority in terms of maximum load on the resources. The other load should be minimized as much as possible. For this reason, we propose an algorithm based *on distance* and *intersection area* calculation. The entire process depicts Figure 4.

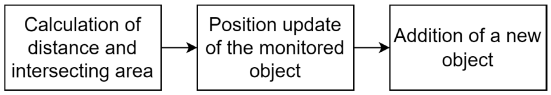


Figure 4. Object tracking.

Object detection produces bounding boxes in the form of a rectanglar shape. The goal is to compute two substantial values:

1. *Calculation of intersecting area*—This includes the calculation of the intersecting area of the valid values between the bounding box of the current frame and the previous frame. Detection loss is marked as an invalid value. Only intersecting rectangles are considered to be the same. On the contrary, deformed bounding boxes by overlapping are not marked as identical;
2. *Calculation of distance*—Calculation of the Euclidean distance of the valid values between the center points of a bounding box of the current frame and the previous frame. Vehicles are identical when the distance between their center points is less than the determined value. We have to bear in mind that it may be difficult to determine the appropriate value. Therefore, the intersecting area is implemented, which is not subject to deformation.

In addition, we set the size of the tracking history. Once the position update of monitored objects occurs, we add a new object based on the above criteria, and such a candidate has to appear at least in the two frames.

3.4. Proposed Approach

Object detection brings substantial information about localization and category of detected objects along with their confidence score as a guarantee that it is the object that is in the center of interest.

The entire process can be divided into three main steps:

1. *Motion detection*—Creation of a binary frame, in which only moving objects are recognized by white pixels. The rest of the pixels are black. Sufficient motion signalizes triggering computationally more expensive object detection. Otherwise, only motion detection analyzes the selected environment;
2. *Object detection*—Localization and recognition of multiple objects with an adequate confidence score in the frame;
3. *Object tracking*—This part ensures that detected objects are the same during the trajectory change.

One of the possible situations is shown in Figure 5, where the view is above the road level. In such circumstances, vehicles move from top to bottom. Undoubtedly, moving

objects can point in the opposite direction. The whole frame is not in the center of interest for assorted reasons, such as parts of the frame that represent background in the form of verdure or the sky. The only center of interest is the road. Vehicles that enter the ROI are detected but counted if the car is completely within the specified region. The trajectory is thus tracked, and we are able to analyze whether it is the same object. Of course, it is important to determine the margin from which we define that the object locates inside.

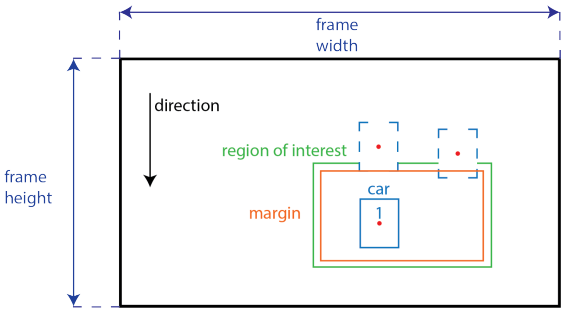


Figure 5. Proposed approach.

One of the possible ways to capture the mentioned scenario is to adjust the view of the camera in front of the vehicle. Figure 6 depicts this scenario and outlines the ROI along with the margin. The vehicle is located inside the margin and is considered to be analyzed.

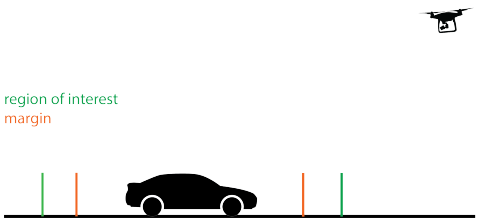


Figure 6. View of the camera.

In this part, we determine several enhancements to increase the chance to obtain the most reliable results and also reduce the hardware requirements; these include the use of motion detection, the implementation of a ROI in form of the bounding box, the requirement for the whole vehicle to be visible in the image, and use of the confidence score.

Several non-optimal scenarios may occur during the process of capturing or detection, and many of these are depicted in Figure 7. The first case represents incorrect detection as a result of an exceedingly small bounding box. The second case is similar and presents the opposite scenario, in which the size of a bounding box is too big. The solution is to introduce the reference size of the bounding box based on previous trusted bounding boxes. The object is considered trusted only when its confidence score exceeds the set limit. The third scenario deals with a partially visible bounding box, which is not involved in the analysis because it is not fully visible in the ROI. The last case depicts a potential issue with detection loss in several frames caused by the detection algorithm. In our case, we try to reduce the ROI as much as possible.

In this stage of development, the ROI does not compensate for the larger movements of the UAV's camera caused by the weather or other related unintentional movement. Should the rapid change occur within the ROI, the user is notified of the possible loss of this region. Then, it is possible to redefine it at a specific time. Other approaches are also possible. The ROI may be tracked, and this would add another step to the processing of the input and would also increase the computational load on the hardware, thus slowing down the whole detection. This approach is also more convenient in case only the video is available from the UAV. Another option is to compensate for the movement directly on the UAV while acquiring video data; research on such tracking of the objects was done by Yeom [28]. The

author concludes that the system for tracking is suitable for surveillance applications. The approach proposed by the author may render the need for a ROI redundant.

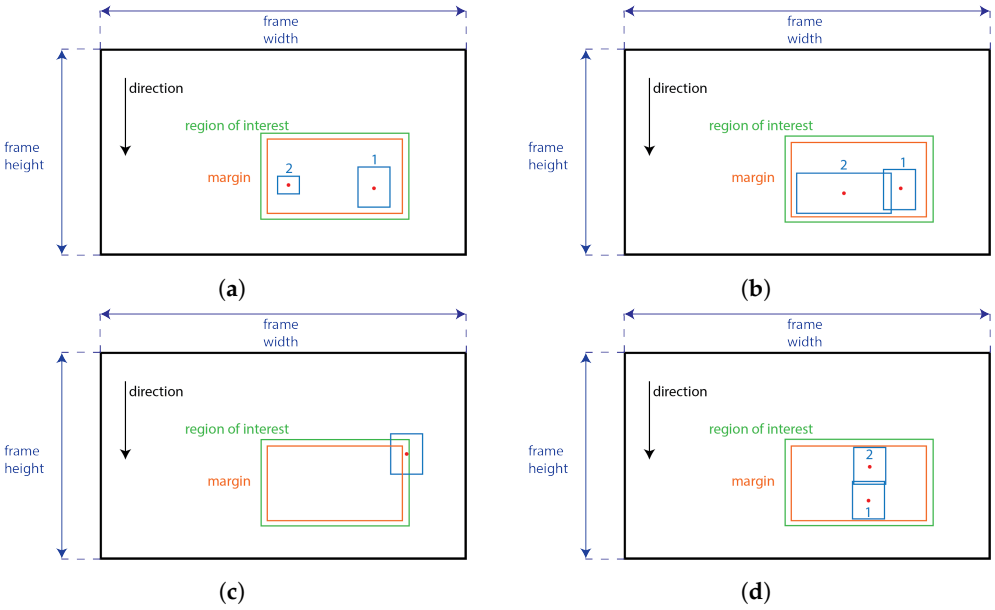


Figure 7. Situations appearing in the proposed approach. (**a**) Incorrectly detected object is too small; (**b**) incorrectly detected object is too big; (**c**) object is only partially visible; (**d**) detection loss.

3.5. Overall Functionality of the System

The architecture diagram of the proposed solution is depicted in Figure 8; the diagram is divided into four main parts that include the client-side, server-side, database, and core. The user triggers the website via a device that enables it to run the browser. The client-side utilizes several libraries to render the website, communicate with the server, and manage the state that contains all current data. The website also communicates with Firebase used for authentication purposes.

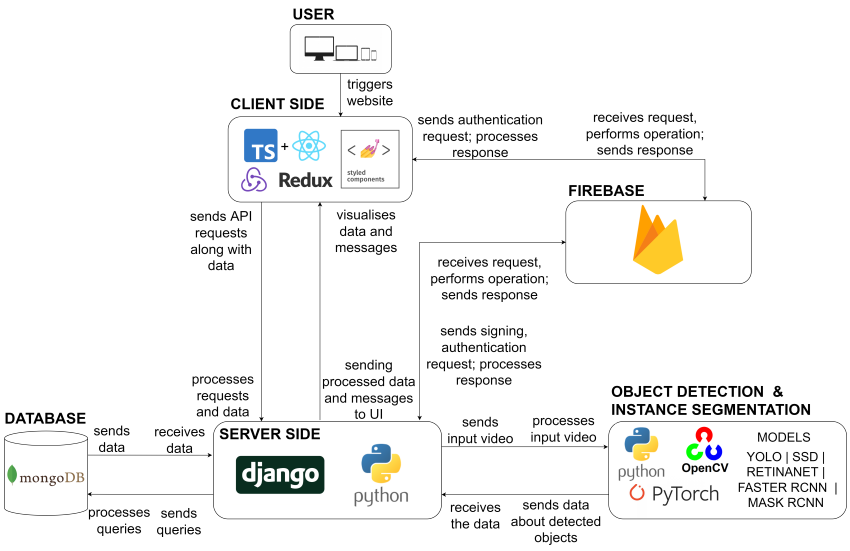


Figure 8. Architecture diagram.

The proposed software solution is rather unique in its nature since it provides an all-in-one solution for the processing of the input from UAVs built on open-source technology. Due to the nature of the solution, the video input from the static traffic cameras is also possible. A rather recent paper from Neupane, Horanont, and Aryal [29] is following the analogous approach as this study; however, the authors utilize static surveillance cameras,

rather than UAVs. The ROI is also utilized but differently; rather than selecting significant parts of the image, the authors use polygons to select the road lanes. Such an approach in our case would be critical, due to the nature of UAVs. Another software solution was done by Li et al. [30]; again, authors use a static camera, but, in this case, mounted to a car. Authors implement a solution for the MatLab environment. One of the side, but also important, contributions of this paper is the described architecture of the proposed experimental software solution that is based on open-source technologies.

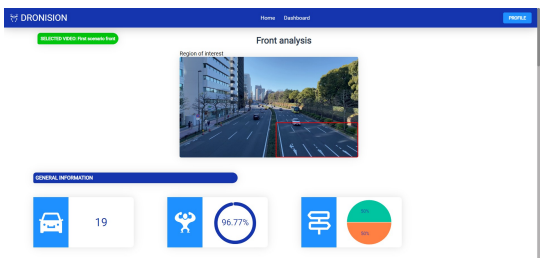
3.6. Video Analysis

Naturally, only a signed-in user can utilize all the functionalities. Account security is provided by Firebase, which checks the user's identity through token authentication. The selected video is prepared for analysis; however, before that, the choice of the ROI is a necessary step. This operation is depicted in Figure 9a. Once done, the analysis is done on the server-side. Communication between the user and the core functionality provides the Django framework. It also communicates with the database to store information. Each video contains a list of analyses. Figure 9b,c show a sample of a single analysis that consists of a count of vehicles, average confidence score, and list of detected vehicles along with their image, confidence score, and detection time.

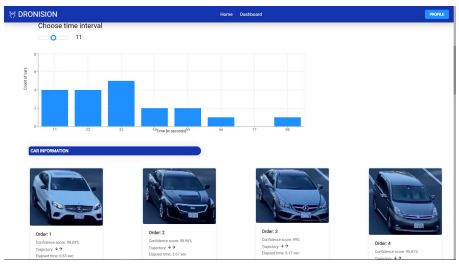
The analysis in the currently implemented solution includes:

- number of detected vehicles;
- average confidence score;
- list of detected vehicles with their visual representation;
- estimation of the vehicle's direction;
- detection time.

(**a**)



(**b**)



(**c**)

Figure 9. The user interface. (**a**) Region of interest section; (**b**) an analysis section; (**c**) an analysis section.

4. Experimental Results

The experimental web-based software solution was evaluated by a sample of users. However, controlled testing on a set of input videos was necessary.

4.1. Performance Evaluation

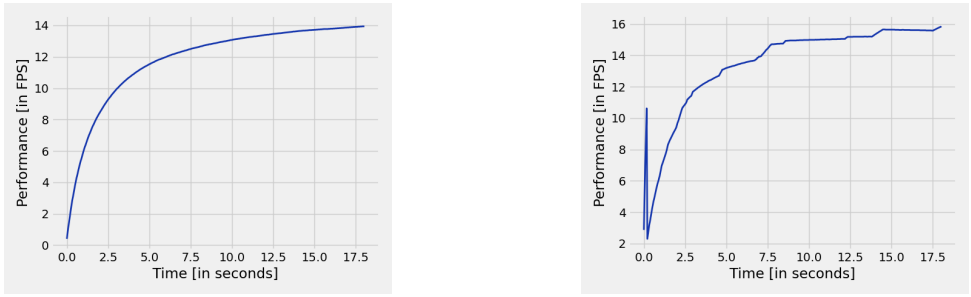
The significant difference between the measurements is the use of enhancements such as motion detection and the use of the ROI. It is important to note that the ROI occupies

15.28% of the total image. Thus, the first measurement uses the entire image without motion detection. The input selection from the video does not contain excessive parts where no motion is detected. This is because object detection is the process that burdens the hardware the most. All the measurements utilize the video with a length of 18 s depicted in Figure 10a.

(**a**) (**b**)

Figure 10. First frames of the Sample scenario #1. (**a**) Front view of the Sample scenario #1; (**b**) rear view of the Sample scenario #1. Source of the input video: [26].

Two charts that have values acquired from the detection using GPU are shown in Figure 11. From the results, it is clear that the use of enhancements increases performance by approximately 2 FPS. This is due to motion detection, which does not have high hardware requirements, and the input image was in this case only 15.28% compared to the whole image. Evaluation on the CPU showed comparable results, but the performance reached 4 FPS.



(**a**) (**b**)

Figure 11. Performance evaluation of the YOLOv4 on the GPU. (**a**) Performance of the full frame on the GPU; (**b**) performance of the part of the frame on the GPU.

4.2. Evaluation of the Object Tracking Algorithm

Our goal is to verify the implementation of our own proposed algorithm using the *intersection of rectangles* that represents the vehicle, i.e., the car. Unfortunately, it is challenging to determine the appropriate distance, mainly due to the loss of detection and thus the increasing distance. As the YOLOv4 algorithm reports stable detection with high accuracy, it is not suitable for the proposed evaluation. For this reason, we replace it with the Faster R-CNN MobileNetV3-Large 320 FPN algorithm, which is characterized by lower accuracy and higher loss of detection; for a detailed comparison, see the review by Chai et al. [31].

The measurement is performed on the same input video as in the performance evaluation. We set the limit distance of 25 pixels and the limit intersection of 25%. Table 2 demonstrates that higher detection loss causes the distance between two bounding boxes to be too large. It may be relatively challenging to set the appropriate distance. The results show that intersection is *key* in the cases in which the distance is greater than determined. The detection loss exists in several frames of the video. Consequently, two vehicles are counted twice without the *use of an intersection*.

Table 2. Evaluation of the object tracking algorithm.

Object Detection Model	Count of Cars without Using Intersecting Area	Count of Cars Using Intersecting Area	Distinction [in %]
Faster R-CNN MobileNetV3-Large 320 FPN	8	10	20

4.3. Sample Scenario #1

An input video is divided into two parts based on the direction of the cars. The quality of the record is quite decent, cars are in the shadows, and their metallic can blend in with the background. Figure 10 depicts the first frames of the input videos to be analyzed.

The results of the analysis concerning general information are shown in Table 3. Remarkably, the time for analysis of vehicles from the front view is almost the same as from the rear view. On the contrary, the count of bounding boxes is only 1.2 times higher than from the rear view. Such results are caused by the fact that cars that move from the camera get smaller; thus, more bounding boxes are detected than otherwise.

Specific information about video analysis is shown in Table 4. The average confidence score reaches almost 100%, and the direction of the vehicles corresponds to the real state based on the camera view. The rear view reaches an accuracy of 93% in the estimation of the number of vehicles. Detection loss can be caused by the fact that the vehicle is considered a van, which is not a personal vehicle. The front view reaches 90%, and two cars are missed. It can be caused by the fact that the vehicle blends in with the background or the neural network cannot recognize the object category with a high confidence score.

Table 3. General information of Sample scenario #1.

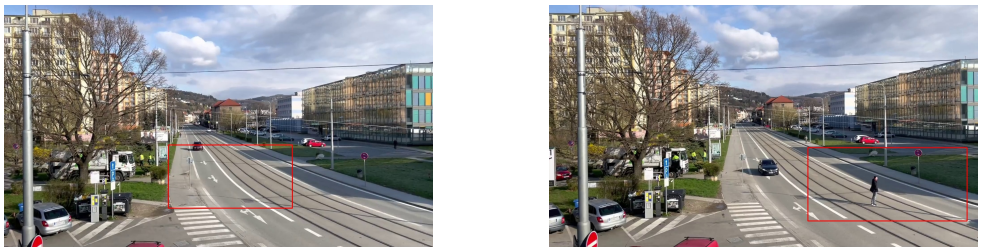
View	Object Detection Model	Performance [in FPS]	Area of Analysis [in %]	Count of Bounding Boxes	Total Time [in s]
Front	YOLOv4	16.72	15.28	413	80
Rear	YOLOv4	15.46	10.15	345	27

Table 4. Specific information of the Sample scenario #1.

View	Object Detection Model	Average Confidence Score [in %]	Direction of Cars	Count of Cars	Real Count of Cars	Accuracy [in %]
Front	YOLOv4	97.99	to back	19	21	90
Rear	YOLOv4	98.31	to front	13	14	93

4.4. Sample Scenario #2

The sample video was made on a sunny day with excellent conditions for vehicle detection. We present two views based on the direction of cars to and from the camera. Figure 12 depicts the first frames of the input videos to be analyzed.



(**a**) (**b**)

Figure 12. First frames of the Sample scenario #2. (**a**) Front view of Sample scenario #2; (**b**) rear view of Sample scenario #2.

The results of the analysis with respect to general information are shown in Table 5, and specific information is shown in Table 6. Both videos last a relatively long time with

a low traffic density. This factor causes high performance. Road character determines the optimal area of the ROI, which occupies only 7.55% of the total area. The rear view occupies 11.24% of the total area. Excellent visibility contributed to a high confidence score, which reaches 98.84%, and the direction of cars is detected correctly, i.e., down. Vehicles are counted with accuracy that reaches almost 100%. In this case, only one car is not detected, which seems to be a van and occupies a larger area.

Table 5. General information of Sample scenario #2.

View	Object Detection Model	Performance [in FPS]	Area of Analysis [in %]	Count of Bounding Boxes	Total Time [in s]
Front	YOLOv4	25.32	7.55	675	271
Rear	YOLOv4	26.64	11.24	261	219

Table 6. Specific information of Sample scenario #2.

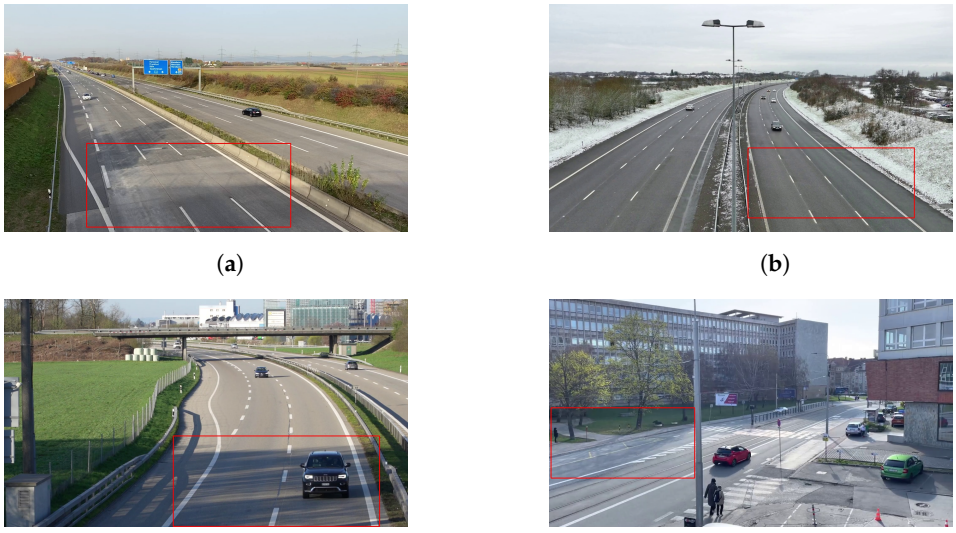
View	Object Detection Model	Average Confidence Score [in %]	Direction of Cars	Count of Cars	Real Count of Cars	Accuracy [in %]
Front	YOLOv4	98.36	to back	18	19	95
Rear	YOLOv4	99.38	to front	10	10	100

4.5. Other Scenarios

The above described are two sample scenarios. However, multiple input videos were evaluated as a part of the solution, see Figure 13. Videos were taken using the UAVs at the premises of the Technical University of Kosice; another source of the videos was [26]. The extracted data from the videos are shown in Table 7. Confidence score reaches 99.10%. The ROI took up to 20% of the total area. The direction, both in the front and the rear view, was tested and detected correctly. All vehicles were counted, and accuracy reached 100%. This data confirmed the solution to be usable in favorable weather conditions.

Table 7. Summary from other scenarios.

View	Object Detection Model	Performance [in FPS]	Area of Analysis [in %]	Average Confidence Score	Accuracy [in %]
Front	YOLOv4	24.93	13.30	99.10	100



(**a**) (**b**)

(**c**) (**d**)

Figure 13. Set of testing scenarios. (**a**) Front view of Sample scenario #3; (**b**) front view of Sample scenario #4; (**c**) front view of Sample scenario #5; (**d**) front view of Sample scenario #6.

5. Conclusions

The main goal of the research was to implement an experimental software solution for video analysis. The core of the proposed service was to bring information from videos recorded from a UAV perspective. However, data from static cameras may also be used as input.

Experimental verification proved the real behavior of the software solution to be successful. The first part of verification included performance evaluation with a focus on the utilization of optimizations such as motion detection, region of interest, etc. We confirmed that the enhancements reduce the hardware requirements. The highest detection stability of YOLOv4 compared to Faster R-CNN ResNet-50 FPN was successfully proven. The proposed simple object tracking algorithm was also evaluated. We conclude that the proposed approach performs as expected with acceptable and above-average results at an accuracy of around 90% in Sample scenario #1 and accuracy from 95% to 100% in Sample scenario #2. Accuracy in the case of the other four scenarios was 100%.

UAVs can move to various parts of the city and make recordings in a sufficient time. On the contrary, unfavorable wind conditions lead to a significant camera movement, which may distort results. Proposed countermeasures prominently reduce data degradation through their architecture and optimizations.

As a result of the mobility, there is optimization in the detection process and our own simple tracking algorithm. The overall solution makes the collection of temporal traffic easier and efficient. Currently, in the Slovak Republic, such monitoring is done manually by counting vehicles or using fixed traffic cameras, which require already existing infrastructure.

The proposed experimental solution can be extended and provide more data from the real environment—for instance, the utilization of neural networks for color detection of cars, car models, or other object categories. We can also remove the background of images that represent detected cars using Mask R-CNN, which aims at instance segmentation. A significant task is the modification of the neural network architecture and subsequent training for adaptation to small objects. Input from the UAVs carried out as a part of this research, included favorable weather conditions, i.e., mostly sunny, with no wind or other external disturbances. Additional improvements directly related to UAVs will cover countermeasures for the disturbances caused by the weather (wind, fog, or rain).

Author Contributions: Conceptualization, O.K. and M.D.; methodology, M.D.; software, M.D.; validation, F.J. and M.M.; formal analysis, F.J. and I.N.; investigation, M.M. and I.N.; resources, M.M.; data curation, F.J.; writing—original draft preparation, M.D. and O.K; writing—review and editing, M.M. and F.J.; visualization, F.J. and I.N.; supervision, F.J.; project administration, O.K.; funding acquisition, F.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This publication was realized with the support of the Operational Programme Integrated Infrastructure in the frame of the project: Intelligent systems for UAV real-time operation and data processing, code ITMS2014+: 313011V422 and co-financed by the European Regional Development Fund.

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

References

1. Chodorek, A.; Chodorek, R.R.; Yastrebov, A. The Prototype Monitoring System for Pollution Sensing and Online Visualization with the Use of a UAV and a WebRTC-Based Platform. *Sensors* **2022**, *22*, 1578. [CrossRef] [PubMed]
2. Guimarães, N.; Pádua, L.; Adão, T.; Hruška, J.; Peres, E.; Sousa, J.J. VisWebDrone: A Web Application for UAV Photogrammetry Based on Open-Source Software. *ISPRS Int. J. Geo-Inf.* **2020**, *9*, 679. [CrossRef]

3. Shvetsova, S.; Shvetsov, A. Ensuring safety and security in employing drones at airports. *J. Transp. Secur.* **2021**, *14*, 41–53. [CrossRef]
4. Lyovin, B.; Shvetsov, A.; Setola, R.; Shvetsova, S.; Tesei, M. Method for remote rapid response to transportation security threats on high speed rail systems. *Int. J. Crit. Infrastruct.* **2019**, *15*, 324–335. [CrossRef]
5. Zhang, P.; Zhong, Y.; Li, X. SlimYOLOv3: Narrower, Faster and Better for Real-Time UAV Applications. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27–28 October 2019. [CrossRef]
6. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
7. Bochkovskiy, A.; Wang, C.; Liao, H. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
8. Qiu, Z.; Zhu, X.; Liao, C.; Shi, D.; Qu, W. Detection of Transmission Line Insulator Defects Based on an Improved Lightweight YOLOv4 Model. *Appl. Sci.* **2022**, *12*, 1207. [CrossRef]
9. Lin, H.-Y.; Tu, K.-C.; Li, C.-Y. VAID: An Aerial Image Dataset for Vehicle Detection and Classification. *IEEE Access* **2020**, *8*, 212209–212219. [CrossRef]
10. Chang, B.R.; Tsai, H.-F.; Lyu, J.-L. Drone-Aided Path Planning for Unmanned Ground Vehicle Rapid Traversing Obstacle Area. *Electronics* **2022**, *11*, 1228. [CrossRef]
11. Huang, L.; Qiu, M.; Xu, A.; Sun, Y.; Zhu, J. UAV Imagery for Automatic Multi-Element Recognition and Detection of Road Traffic Elements. *Aerospace* **2022**, *9*, 198. [CrossRef]
12. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
13. Chiu, Y.-C.; Tsai, C.-Y.; Ruan, M.-D.; Shen, G.-Y.; Lee, T.-T. Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems. In Proceedings of the 2020 International Conference on System Science and Engineering (ICSSE), Kagawa, Japan, 31 August–3 September 2020. [CrossRef]
14. Xie, X.; Cao, G.; Yang, W.; Liao, Q.; Shi, G.; Wu, J. Feature-fused SSD: Fast detection for small objects. In Proceedings of the Ninth International Conference on Graphic and Image Processing (ICGIP 2017), Qingdao, China, 14–16 October 2017.
15. Veeranampalayam Sivakumar, A.N.; Li, J.; Scott, S.; Psota, E.; Jhala, A.; Luck, J.D.; Shi, Y. Comparison of Object Detection and Patch-Based Classification Deep Learning Models on Mid- to Late-Season Weed Detection in UAV Imagery. *Remote Sens.* **2020**, *12*, 2136. [CrossRef]
16. Wei, B.; Barczyk, M. Experimental Evaluation of Computer Vision and Machine Learning-Based UAV Detection and Ranging. *Drones* **2021**, *5*, 37. [CrossRef]
17. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
18. Li, Y.; Dua, A.; Ren, F. Light-Weight RetinaNet for Object Detection. *arXiv* **2019**, arXiv:1905.10011.
19. Ahmad, M.; Abdullah, M.; Han, D. Small object detection in aerial imagery using RetinaNet with anchor optimization. In Proceedings of the 2020 International Conference on Electronics, Information and Communication (ICEIC), Barcelona, Spain, 19–22 January 2020. [CrossRef]
20. Psiroukis, V.; Espejo-Garcia, B.; Chitos, A.; Dedousis, A.; Karantzalos, K.; Fountas, S. Assessment of Different Object Detectors for the Maturity Level Classification of Broccoli Crops Using UAV Imagery. *Remote Sens.* **2022**, *14*, 731. [CrossRef]
21. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
22. Xu, Y.; Yu, G.; Wang, Y.; Wu, X.; Ma, Y. Car detection from low-altitude UAV imagery with the faster R-CNN. *J. Adv. Transp.* **2017**, *2017*, 2823617. [CrossRef]
23. Avola, D.; Cinque, L.; Diko, A.; Fagioli, A.; Foresti, G.L.; Mecca, A.; Pannone, D.; Piciarelli, C. MS-Faster R-CNN: Multi-Stream Backbone for Improved Faster R-CNN Object Detection and Aerial Tracking from UAV Images. *Remote Sens.* **2021**, *13*, 1670. [CrossRef]
24. Piedad, E.J.; Le, T.-T.; Aying, K.; Pama, F.K.; Tabale, I. Vehicle count system based on time interval image capture method and deep learning mask R-CNN. In Proceedings of the TENCON 2019—2019 IEEE Region 10 Conference (TENCON), Kochi, India, 17–20 October 2019. [CrossRef]
25. Wang, H.; Yu, Y.I.; Cai, Y.; Chen, X.; Chen, L.; Liu, Q. A comparative study of state-of-the-art deep learning algorithms for vehicle detection. *IEEE Intell. Transp. Syst. Mag.* **2019**, *11*, 82–95. [CrossRef]
26. Pexels Videos. Free Stock Videos. Available online: https://www.pexels.com/videos (accessed on 14 April 2021).
27. Jayaweera, H.M.P.C.; Hanoun, S. Path Planning of Unmanned Aerial Vehicles (UAVs) in Windy Environments. *Drones* **2022**, *6*, 101. [CrossRef]
28. Yeom, S. Long Distance Ground Target Tracking with Aerial Image-to-Position Conversion and Improved Track Association. *Drones* **2022**, *6*, 55. [CrossRef]
29. Neupane, B.; Horanont, T.; Aryal, J. Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network. *Sensors* **2022**, *22*, 3813. [CrossRef]
30. Li, Y.; Wang, J.; Huang, J.; Li, Y. Research on Deep Learning Automatic Vehicle Recognition Algorithm Based on RES-YOLO Model. *Sensors* **2022**, *22*, 3783. [CrossRef]
31. Chai, J.; Hao, Z.; Li, A.; Ngai, E. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Mach. Learn. Appl.* **2021**, *6*, 100134. [CrossRef]