




# Deep Learning-based Monocular Obstacle Avoidance for Unmanned Aerial Vehicle Navigation in Tree Plantations

## Faster Region-based Convolutional Neural Network Approach

H. Y. Lee<sup>1</sup> · H. W. Ho<sup>1,2</sup>  · Y. Zhou<sup>1,2</sup>

Received: 17 February 2020 / Accepted: 28 October 2020 / Published online: 8 December 2020  
© Springer Nature B.V. 2020

### Abstract

In recent years, Unmanned Aerial Vehicles (UAVs) are widely utilized in precision agriculture, such as tree plantations. Due to limited intelligence, these UAVs can only operate at high altitudes, leading to the use of expensive and heavy sensors for obtaining important health information of the plants. To fly at low altitudes, these UAVs must possess the capability of obstacle avoidance to prevent crashes. However, most current obstacle avoidance systems with active sensors are not applicable to small aerial vehicles due to the cost, weight, and power consumption constraints. To this end, this paper presents a novel approach to the autonomous navigation of a small UAV in tree plantations only using a single camera. As the monocular vision does not provide depth information, a machine learning model, Faster Region-based Convolutional Neural Network (Faster R-CNN), was trained for the tree trunk detection. A control strategy was implemented to avoid the collision with trees. The detection model uses image heights of detected trees to indicate their distances from the UAV and image widths between trees to find the widest obstacle-free space. The control strategy allows the UAV to navigate until any approaching obstacle is detected and to turn to the safest area before continuing its flight. This paper demonstrates the feasibility and performance of the proposed algorithms by carrying out 11 flight tests in real tree plantation environments at two different locations, one of which is a new place. All the successful results indicate that the proposed method is accurate and robust for autonomous navigation in tree plantations.

**Keywords** Autonomous UAVs · Tree avoidance · Faster R-CNN · Monocular vision · Smart farming

## 1 Introduction

Unmanned Aerial Vehicles (UAVs) are defined as aircraft without a human pilot on-board for navigation and control. These flying vehicles are controlled either remotely by a ground crew or navigated automatically by a pre-programmed control system. UAVs have been applied in precision

agriculture due to their higher flexibility and capability compared to labor-dependent techniques.

UAVs have been used to collect aerial images and other important information with the on-board sensors. A mission can be performed efficiently and effectively by processing the data obtained from UAVs. These data support the farmers to carry out several essential tasks in plantations, such as the farming analysis and planning [8], plantation surveillance [19], and the subsequent monitoring of fields to ascertain health and growth, including crop monitoring [30] and soil sampling analysis [23].

With current technological advancements, UAVs accomplish precision agriculture tasks, such as tree counting and monitoring [9], by flying at a high altitude. At this height, they have a wide field of view for a better observation and an ample flying space without any obstacle. However, it requires more expensive sensors or equipment to obtain a reasonable quality of data. It is also difficult to perform

---

✉ H. W. Ho  
aehannwoei@usm.my

Y. Zhou  
Y.Zhou-6@tudelft.nl

<sup>1</sup> School of Aerospace Engineering, Universiti Sains Malaysia, 14300 Nibong Tebal, Pulau Pinang, Malaysia

<sup>2</sup> Faculty of Aerospace Engineering, Delft University of Technology, 2629HS Delft, The Netherlands

certain tasks from a distance. On the other hand, UAVs that are capable of operating at low altitudes can perform many other missions with more affordable solutions. For instance, they can closely monitor crops and spray fertilizer with a suitable quantity based on the soil condition. These UAVs can use a tiny camera to capture images at a near distance to the target objects, such as the soil or crop, for further data processing and analysis. The applications of UAVs at low altitudes offer more cost-effective and efficient solutions compared to the labor in a vast field.

However, the greatest challenge to overcome in a low-altitude flight is to prevent UAV crashes in the cluttered environment while navigating. Experienced human pilots are able to recognize and avoid obstacles from line-of-sight operations or live-stream videos. However, the plantation environment poses a daunting challenge to human pilots due to the heavy workload. Additionally, the recruitment of experienced pilots is exceptionally costly. To this end, autonomous navigation with the capability of detecting and avoiding obstacles is needed. Obstacle detection methods can be used to identify objects in the surroundings. Based on the surrounding information, the flight controller produces a motion command to avoid the detected obstacles. Thus, it requires on-board sensors and control algorithms, for the detection and the motion control, respectively.

Many existing methods for obstacle avoidance use active sensors, such as Light Detection and Ranging (LiDAR) [38], inertial sensors, ultrasonic and infrared range finders [10], or passive sensors, such as the RGB-D camera [24] and multiple cameras [43]. While these methods can provide accurate measurements, they have a limited perception range. Additionally, the cost, weight, and power consumption of these instruments impose a significant constraint on the UAV mission. Therefore, a single camera is preferable for obstacle avoidance, as it is small, light-weight, and power-efficient.

Generally, monocular obstacle detection methods can be divided into two categories: motion-based and knowledge-based methods. Optical flow is a typical motion-based approach used to detect and estimate the depth with motion information [12]. The motion information can be obtained by tracking pixel-by-pixel between two consecutive image frames, or the so-called dense optical flow. The dense optical flow requires heavy computational loads and memory sizes. Another approach is to detect feature points in one image and track them in the next image. This approach is called sparse optical flow and is preferable, especially for small UAVs. However, the obstacle may be missed due to the noisy features in the image. Since monocular vision does not allow accurate and robust distance geometric measurement, knowledge-based solutions have been proposed using

machine learning or the approaches which are combined with the optical flow methods [20].

Since Alexnet [28] won ImageNet competition in 2012, Convolutional Neural Network (CNN) has become the benchmark for image classification. CNN approaches have outperformed humans in the ImageNet challenge. Lately, CNN has been applied with great success to the detection, segmentation, and recognition of objects in images. For instance, CNN-based methods were used in vehicle detection [39], medical image detection [21], and fruit detection [42].

Faster Region-based CNN (Faster R-CNN) [40] is a network that uses a CNN network to perform both region proposal generation and object classification, or the so-called region-based object detection task. The Faster R-CNN model has demonstrated an outstanding performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and Common Object in Context (COCO) 2015 competitions. From the other side, the Faster R-CNN algorithm is publicly available in the open sources in TensorFlow and MATLAB. The users can either construct the network from scratch or import a pre-trained network to be re-trained as a detector. Hence, this paper proposed to use the Faster R-CNN as the obstacle detector in the UAV obstacle avoidance system. To the best of the authors' knowledge, this study is the first approach to adapt Faster R-CNN to UAV navigation in tree plantation.

The contributions of this paper are three-fold: 1) Faster R-CNN model is trained to recognize features of tree trunks in images, rather than the whole trees. This approach enables accurate tree detection, especially those closer to the UAV, as they are partly visible when navigating at low altitudes. This paper analyzes and compares the performance between the Faster R-CNN networks constructed from scratch and pre-trained to find the most suitable one. 2) A control strategy is proposed to identify obstacles that endanger the UAV during navigation by predicting the depth information of obstacles based on their size. It also determines the desire heading of the UAV, which has the widest flight path. 3) The obstacle avoidance (i.e., the detection and control strategy) has been successfully implemented on a Parrot Bebop 2 and tested in real-world environments.

The remainder of the article is set up as follows: In Section 2, related work on monocular obstacle avoidance is discussed in details. Section 3 presents the proposed Faster R-CNN networks to learn the visual features of tree trunks. In Section 4, the control strategy for obstacle avoidance is explained. Section 5 shows and describes the performance results of different Faster R-CNN networks. Then, Section 6 demonstrates flight experiments where the

proposed algorithms detect and avoid obstacles with a UAV. Finally, we draw conclusions in Section 7.

## 2 Related Work

There are several remarkable achievements using monocular images as inputs to control UAVs in autonomous navigation [5, 26]. However, with monocular vision, the depth information is not directly available. Few approaches were proposed to overcome this limitation for UAV autonomous navigation.

Structure from Motion (SfM) approaches reconstruct the scenes using the UAV movement. A regularized depth map was computed and subsequently used for the waypoint generation from a small set of consecutive images [3]. A direct depth estimation approach was proposed by enabling real-time computation of dense depth maps and navigation in a cluttered outdoor environment [11]. However, in the SfM-based obstacle avoidance scheme, the UAV cannot avoid dynamic obstacles that move during mapping or between mapping cycles. Moreover, the mapping cycles require an enormous amount of computational memory to store and compare the consecutive frames of the scene to obtain the depth information.

Optical flow-based methods are also used to predict depth measurements. This method was proposed to extract image's depth in a 3D environment based on the gradient method of Lukas-Kanade [17]. By comparing the sequential images, the model found out whether the obstacle was getting closer. With this information, the UAV movement was navigated using a steering command, which is inversely proportional to the optical flow difference between the two sides of the image [1].

Moreover, a variety of bio-inspired optical flow navigation methods have also been proposed [49]. For instance, the translational optical flow method inspired by insect navigation for the collision-free flight was demonstrated [44]. However, the optical flow-based methods cannot acquire precise distances, which may limit their usages in some specific missions.

By contrast, Simultaneous Localization and Mapping (SLAM) methods were proposed to provide precise metric maps with a sophisticated algorithm. This method allows the UAVs to navigate and avoid obstacles with more environment information. The information is usually obtained from ultrasonic sensors or infrared range finders [14]. For instance, a LiDAR-based SLAM for UAV navigation was demonstrated successfully in the indoor environments [4].

To achieve real-time obstacle avoidance, an Oriented fast and Rotated Brief SLAM (ORB-SLAM) was introduced to process the frontal camera's video stream [13]. First, it

computed the 3D locations of the UAV and generated a sparse point cloud map. Then, it enriched the sparse map to denser. Lastly, it created a collision-free roadmap by applying the potential field method and Rapidly exploring Random Tree (RRT) algorithms. These SLAM-based obstacle avoidance methods can perform much more complex tasks but usually fail at high speeds since they reconstructed the environment from frame to frame triangulation.

Some approaches detect the presence of the frontal obstacles and then generate the motion control to avoid them. These approaches are commonly known as the sense-and-avoid mechanism. A strategy used a classification algorithm to identify the types of environment and determine the UAV flight direction based on its known perspective cues [6]. However, this approach was limited to the simple indoor environment. To this end, motion planning techniques were used to determine the optimal path in more complex scenarios [31]. A method combined local trajectory optimisation with learning from demonstrations was proposed to improve the initial solutions for the local optimization [45]. Besides reducing poor local minima, the learned model generated a smooth trajectory. Another approach used an adaptive control scheme to deal with uncertainties of the trajectory due to external disturbances [32].

There are some methods using the change of the feature size for obstacle avoidance. Speeded Up Robust Features (SURF) was proposed to detect, describe, and match features in images with the obstacles' database [2]. The position and orientation of the UAV with respect to the obstacles were estimated and fed into a controller to achieve fast obstacle avoidance. Another method reconstructed the 3D information of the obstacles by merging the outline of the obstacles with their feature points' spatial coordinates [29]. The outline of the obstacles was approximated using Multi-Scale Oriented Patches (MSOP), while the spatial coordinates of the feature points were calculated using the Scale Invariant Feature Transformation (SIFT). However, the drawback of these algorithms is that they only work for the obstacles stored in the database.

Several approaches have been applied to a similar environment as our study. The Dagger algorithm was presented to learn and predict the human expert's control through dense forest environments [41]. Furthermore, a hybrid collision avoidance scheme consisting of an RRT algorithm as the global path planner and a fuzzy logic method as the local collision avoidance mechanism. An extended Kalman filter was also utilized for improving the cross-track error of the flight in the hazardous environment [33].

Recently, deep learning solutions have been proposed to improve real-time performance in a complex unknown environment [46]. Convolutional Neural Network (CNN) was used to learn a control strategy that mimics an expert pilot's choice of action to navigate autonomously in the

indoor [27] and outdoor environment [16]. For instance, in a navigation approach demonstrated in the outdoor environment, the camera orientation estimation was framed as a three-class motion classification: Left, Front, and Right. A set of forest trail images was captured with three head-mounted cameras, each pointing in one direction. Given one frame input, the model decided the next optimal move. This work was demonstrated to follow a specific path, but this study focuses on general navigation.

There have been a few researchers who presented deep learning solutions to predict the depth of the scenes. A fully convolutional network which is fed with both images and optical flow was designed to obtain fast and robust depth estimation [35]. Another algorithm combines fully convolutional network and color classification to extract Regions of Interest (RoI) in images and quickly identify the designated moving targets [48]. The position and direction of the moving targets were then estimated using coordinate transformation and Kalman filter. Besides, a two-stage obstacle avoidance deep reinforcement learning system was proposed. It was composed of a fully convolutional neural network followed by a Double-Q Network (D3QN). The CNN acts as a depth predictor, while the D3QN uses a convolutional network and a dueling network to predict the Q-value of angular actions and linear actions in parallel [47].

Recently, a novel CNN architecture, Joint Monocular Obstacle Detection (J-MOD<sup>2</sup>), was proposed to learn the depth estimation and the obstacle detection from the image features extracted by the fine-tuned VGG19 network [36]. This approach was tested and evaluated in a virtual forest scenario on the Unreal Engine software environment. Also, a saliency detection algorithm was developed using a deep CNN to extract monocular visual cues. A Radial Basis Function (RBF) neural network in an actor-critic reinforcement learning module was then used to control the motion of the UAV [34].

To our knowledge, there is only one other study that also used Faster R-CNN on a UAV to perform obstacle detection. The study in [22] used a strategy that combines Faster R-CNN and Kernelized Correlation Filters (KCF) to detect and track electric towers. The Faster R-CNN uses a ZF-Net model [37] to perform tower detection. The real-time inspection performance was improved using the KCF.

In this article, we investigate the performance of using scratch models and pre-trained models (e.g., Inception v2 [25] and Resnet-50 [18]) in Faster R-CNN for tree detection. From the analysis, the model achieving the best accuracy and fastest speed is chosen and used in the UAV navigation. Compared to [22], the pre-trained models we propose achieve much higher accuracy than the ZF-Net model [7]. Furthermore, this study takes on new challenges with complex, real-world environment cluttered with many

obstacles. This leads to a significant extension of the autonomous flight capabilities of UAVs.

## 3 Obstacle Detection

### 3.1 Network Architecture and Training

Encouraged by the outstanding performance of current detection methods, Faster R-CNN [40] is adopted as the tree detector. Faster R-CNN merges two modules: 1) a deep fully convolutional network that proposes regions, and 2) a Fast R-CNN detector [15] that uses the proposed regions. The use of deep fully convolutional network significantly improves the computational time for the region proposal and enables real-time object detection. In this section, the Faster R-CNN networks are constructed from scratch and pre-trained models. Their performances in terms of detection accuracy and speed are analyzed and compared to finalize one detector for the obstacle avoidance mission.

#### 3.1.1 Scratch Model

A total of four different detectors were created to investigate the effect of the number of training data and the number of convolutional layers on the detection performance. These four detectors, namely Detectors A, B, C, and D, were trained with different specifications. The first manipulated parameter was the number of training images. Detector A was trained with more than 4,000 images, whereas the other three detectors were trained with 1,500 images only. The images in the training dataset of the other three detectors were extracted from the training dataset of Detector A. In other words, the training data of Detectors B, C, and D are the subset of training data of Detector A.

The second manipulated parameter was the number of repeated blocks consisting of convolutional and Rectified Linear Unit (ReLU) activation layers. The convolutional layer extracts the features and generates a feature map. This map is fed into the last layer to identify the presence of the tree trunks and their locations in the image. Detectors B, C, and D were trained with a different number of the repeated convolutional blocks in the range of 2 to 5 layers. Table 1 and Fig. 1 show the architecture of CNN of Detectors A and B. Table 2 summarizes the differences between the detectors.

After the network architecture was defined, the Faster R-CNN detectors were trained to update the parameters in both Region Proposal Network (RPN) and CNN. The algorithm used in the training was Stochastic Gradient Descent with Momentum (SGDM) optimizer with 0.001 as the learning rate.



**Table 1** The description of each layer in the CNN architecture of Detectors A and B

No.	Layer Name	Description
1	Image Input	32x32x3 images with ‘zero-center’ normalization
2	Convolutional	32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
3	ReLU	ReLU activation layer
4	Convolutional	32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
5	ReLU	ReLU activation layer
6	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
7	Fully Connected	64 fully connected layer
8	ReLU	ReLU activation layer
9	Fully Connected	2 fully connected layer
10	Softmax	Softmax Activation Layer
11	Output	Classification Output

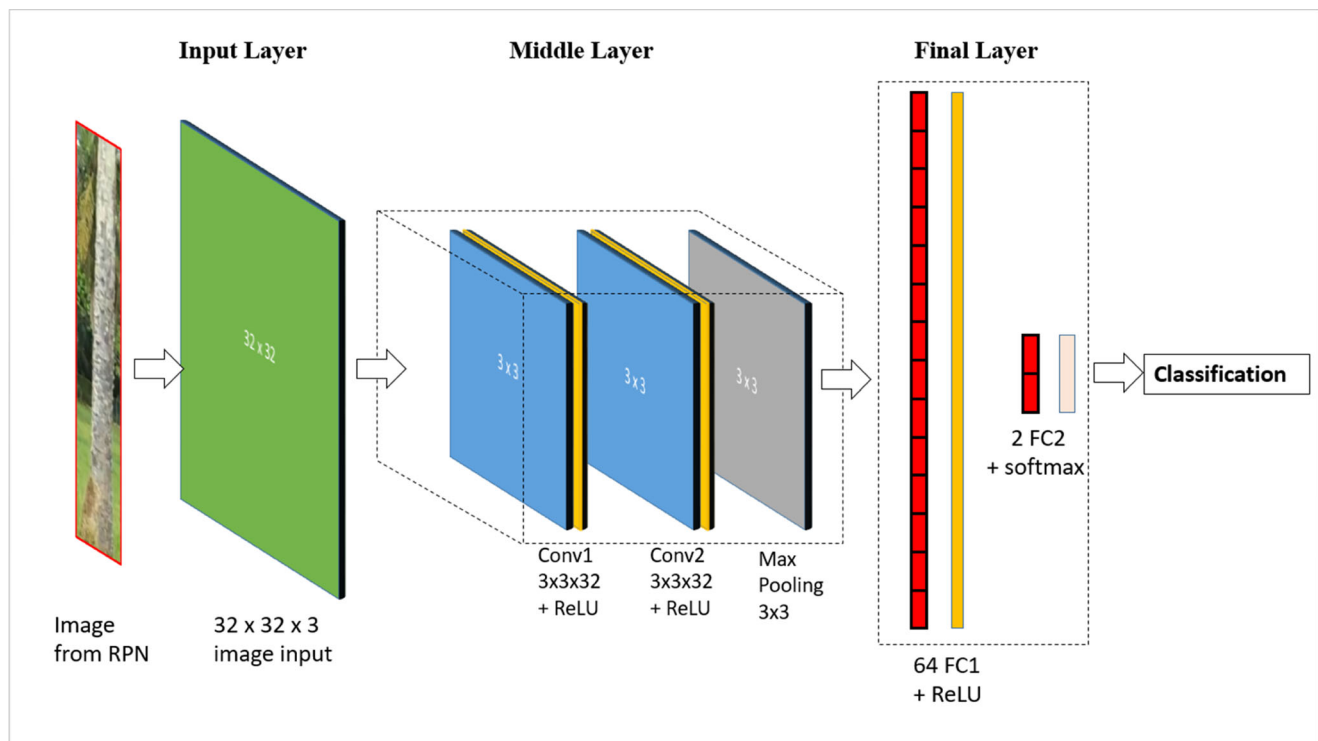
### 3.1.2 Pre-trained Model

In addition to the scratch models, two existing pre-trained models were selected to be re-trained for the tree trunk detection. Both models were trained previously on the COCO dataset.

The first network chosen is Inception v2 [25], the second machine learning model that surpasses human-level performance in ILSVRC in the image classification history. The unique, wide architecture of Inception model shows excellent performance at a relatively low computational cost.

The second pre-trained model is Resnet-50 [18] as the CNN base in the Faster R-CNN configuration. Resnet achieved state-of-the-art results in image classification and detection and won the first place in ILSVRC 2015 and Common Object in Context (COCO) 2015 competitions on the tasks of classification, detection, localization, and segmentation.

Similar to the trained scratch model, the reconstructed Resnet-50 and Inception network were fine-tuned with Stochastic Gradient Descent with Momentum (SGDM) optimizer. However, the learning rate in the training was set to be 0.0002, which is much smaller than the one in



**Fig. 1** The CNN architecture of Detectors A and B. ‘Conv.’ represents convolutional layers, whereas ‘FC’ represents fully connected layers

**Table 2** The specifications of the four detectors

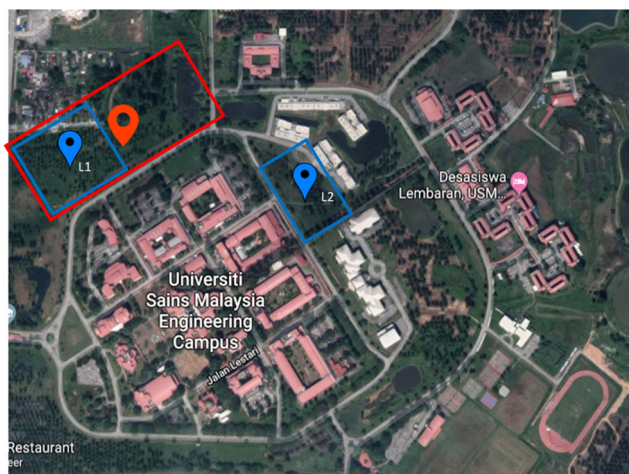
Parameter	Detector			
	A	B	C	D
Number of training images	4000	1500	1500	1500
Number of Conv. layers	2	2	3	5

the scratch model (0.001). This is to reduce the risk of losing the previously extracted features without severely distorting the weight of the layers. The performance of these models was compared with the optimum designed scratch model. The one with the best performance was selected to be implemented on the UAV platform.

### 3.2 Collection and Annotation of Flight Data

In order to train and validate a detection model, it requires a large dataset consisting of the images of the target environment. In this study, a Parrot Bebop 2 is used as the airborne platform to record the flight videos in an oil palm plantation. Note that the proposed methodology can be applied to other targets, such as pine trees or banana trees. The location where the exact dataset is collected is shown in Fig. 2. The dataset consists of the videos of the UAV random flights among oil palm trees for training the detector with a real flight scenario. The data collection is conducted in three different blocks of time within a day (i.e., morning, afternoon, and evening) in the same environment to ensure the model can detect the obstacles in different lighting conditions.

The videos recorded by the UAV are originally in size of  $1920 \times 1080$  pixels. The videos are resized to  $426 \times$



**Fig. 2** The location map in which the dataset is collected. The red box shows the area of the location where the dataset is collected, while the blue boxes indicate the test flight locations, which are labelled as Location 1 (L1) and Location 2 (L2), respectively

240 pixels before proceeding to the object annotation to save the computational power during training. The dataset is labelled manually in Video Labeler application in MATLAB. All the tree trunks are labelled with rectangular Regions of Interest (RoI) in every frame of the videos with the aids of the built-in automated algorithm. Figure 3 shows the example images with the rectangular RoI labelled around the object of interest, i.e., the tree trunk.

## 4 Avoidance Motion

### 4.1 Localization of Detected Obstacles

The detection result is used to decide the motion control in the UAV autonomous navigation. Each raw image is resized to match the input layer size of the detection model and then fed into the trained model to detect the obstacles and their locations. The detection model outputs the coordinates of the bounding boxes of obstacles by referring the origin at the top left corner of an image. The image coordination system  $(x_i, y_i)$  is shown in Fig. 4. The top left and bottom right coordinates of the bounding boxes are defined as  $(x_{i,j,min}, y_{i,j,min})$  and  $(x_{i,j,max}, y_{i,j,max})$ , respectively.

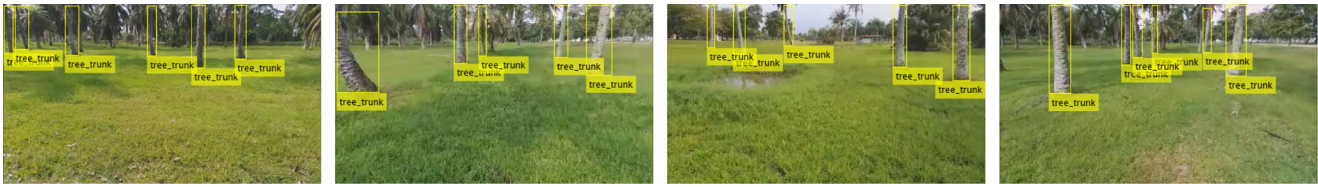
In our obstacle avoidance system, the detection model detects the presence of the tree trunks and localizes them in the images. Only the tree trunks detected with the confidence score more than 0.7 are visualized in the output images and recognized as the obstacles in the frames. Most bounding boxes with low confidence scores are false-positive results, or the tree trunks that are very thin and small in the image, which indicates that the objects are far away. These results can be neglected as they do not endanger the current navigation of the UAV. This can eliminate the excess amount of the bounding boxes with low confidence scores and save the computation in the avoidance control stage.

### 4.2 Identification of Critical Obstacles

After the coordinates of the detected obstacles are obtained, the approaching obstacles along the forward path of the UAV need to be identified. In this paper, they are called “critical obstacles”. The heights of the bounding boxes are used as the depth clues in the algorithm. In the controller, the UAV is commanded to move forward with a constant velocity until it is triggered to stop when a critical obstacle is detected.

Based on the above scenario, the critical obstacles can be identified with two conditions:

- Condition 1.** it is approaching to the UAV, and
- Condition 2.** it is inside a safety boundary blocking the flight path.



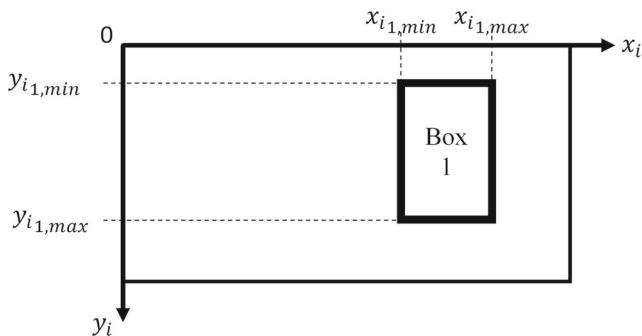
**Fig. 3** Examples of the ground truth labelled in the images. The yellow boxes are the labelled bounding boxes to represent the tree trunks in the images

The safety boundary is set to be 0.5 m from the UAV camera, as shown in Fig. 5. It is to ensure that the UAV is free from collisions while navigating forward. It covers the size of UAV and some accepted spaces for the UAV to navigate safely. Also, the algorithm should allow the UAV to detect obstacles at least 1 m away. This safety precaution can solve some practical issues which may be encountered during the flight, such as the delay caused by the transmission of the image data and motion control commands via Wi-Fi.

As mentioned above, the heights of the detected bounding boxes in images are used as the depth clues. This parameter can be affected by the height of the UAV. Although the UAV height can be controlled using the sonar measurement feedback, the UAV hardly maintains at an exact constant height in real flights. This is due to the external environmental factors, such as soft and uneven ground, side wind, or gust, which affect the measurement and UAV stability.

To this end, a parameter called height ratio,  $h_{ratio}$ , which is the ratio of an obstacle height in the image to the image height, is introduced. It is used to decide whether the obstacle is critical at a measured height. Hence, the height ratio of the obstacle when it is 1 m ahead from the camera at different flying heights,  $Z$  in meters, is determined using Eq. 1:

$$h_{ratio} = \frac{[Z + \tan(\frac{VFOV}{2})] \cdot f}{h_{img}} \tag{1}$$



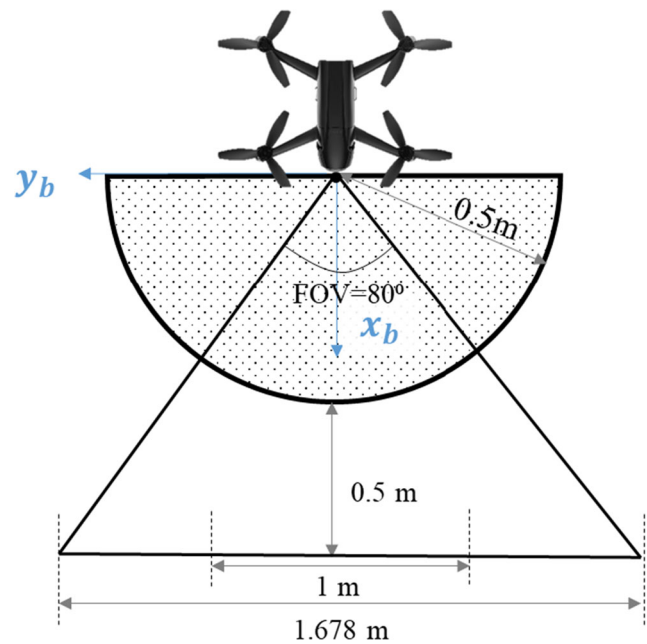
**Fig. 4** The image axis with the coordinates of a bounding box surrounding a detected obstacle, Box 1

where  $VFOV$  is the Vertical Field of View in degrees,  $h_{img}$  is the image height in pixels, and  $f$  is the focal length of the camera in pixels which can be obtained from camera calibration.

To examine Condition 1,  $y_{i,j,max}$  of the bounding box, as shown in Fig. 4, is used to represent the height of the object in the image. When the ratio of  $y_{i,j,max}$  to  $h_{img}$  is greater than or equal to  $h_{ratio}$  according to Eq. 2, the detected obstacle fulfills Condition 2, which means that the obstacle is approaching to the UAV.

$$y_{i,j,max} / h_{img} \geq h_{ratio} \tag{2}$$

Note that  $y_{i,j,max}$  is used, instead of the actual obstacles height in the image, because the bounding boxes may not fully cover the obstacle from the top boundary of the image,  $y_i = 0$ , e.g., the upper part of the tree trunk may be covered by the tree leaves.



**Fig. 5** The UAV body reference axis from the top view. The shaded area represents the safety boundary of the UAV with a radius of 0.5 m. The horizontal field of view of the on-board camera is 80°. The length of the actual horizontal view when the scene is 1 m ahead of the camera is determined as 1.678 m by using the trigonometry method

To test Condition 2, the locations of the obstacles in the x-axis of the image frame are calculated. The  $x_{ij}$ -coordinates of the bounding boxes, as presented in Fig. 4, are used to analyze whether the locations of obstacles are inside the safety boundary of the UAV. In this study, the on-board camera is assumed to be a pinhole camera as the distortion has been corrected by camera calibration and ROS framework. Hence, the range of  $x_{ij}$ -coordinates of the obstacle in the safety boundary can be defined as:

$$(0.5 \cdot w_{img} - w_{obj}) \leq (x_{ij,min} \text{ or } x_{ij,max} \text{ or } x_{ij,c}) \leq (0.5 \cdot w_{img} + w_{obj}), \tag{3}$$

where  $x_{ij,min}$ ,  $x_{ij,max}$ , and  $x_{ij,c}$  are the minimum, maximum, and center of x-coordinates of the bounding box, respectively. When the object is 1 m ahead from the camera,  $w_{obj}$  can be calculated using Eq. 4:

$$w_{obj} = \frac{w_{img}}{4 \tan(\frac{HFOV}{2})}, \tag{4}$$

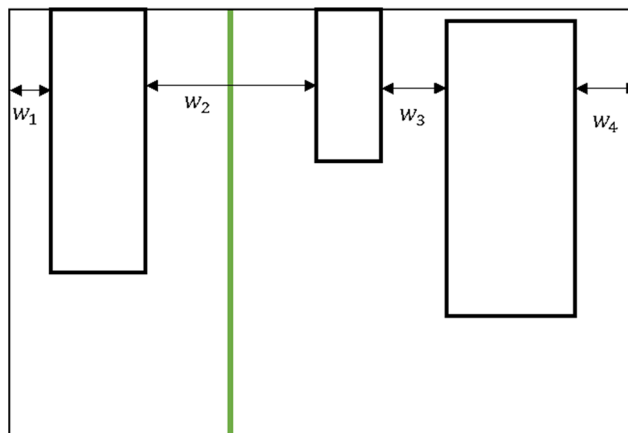
where  $HFOV$  is the Horizontal Field of View of the camera in degrees and  $w_{img}$  is the width of the image in pixels. By using these two conditions stated in Eqs. 2 and 3, the critical obstacles can be identified.

### 4.3 Estimation of the Desired Heading Reference

After a critical obstacle is detected, the UAV is steered to the desired heading angle. The desired heading reference is determined by also considering the approaching but not yet critical obstacles. This type of obstacle is known as the “warning obstacle” in this paper. For these obstacles, the height of their bounding box is more than  $\frac{1}{3}$  of the image height. By taking these obstacles into account, the new desired heading angle can guide the UAV to a smoother path. This reduces the excessive steering in the navigation and makes the route more relaxed than the approaches that only sense and avoid the critical obstacles.

The desired heading reference is determined by finding the largest obstacle-free region in the image x-axis. Figure 6 shows an example of the desired heading reference with the bounding boxes representing the detected obstacles. The symbol  $w_i$  represents the horizontal distances between the bounding boxes of the warning and critical obstacles and between the obstacles and image borders. The subscript  $i$  is the number of free spaces in the image along x-axis. The section with the highest value of  $w_i$  represents the largest obstacle-free region, indicated as a green vertical line in Fig. 6. The center of the largest obstacle-free region,  $x^*$ , is calculated to determine the desired heading reference.

In case, when the highest value of  $w_i$  is less than 80 pixels, which is equivalent to 0.32 m width when the UAV is 1 m ahead, the free space is too small for the UAV to pass through. For this situation,  $x^*$  is set to be the right



**Fig. 6** An example of the desired heading reference that is represented by the green vertical line in the field of view of the camera.  $w_1$  and  $w_4$  are the horizontal distances between the detected obstacles and image borders, while  $w_2$  and  $w_3$  are the horizontal distances between obstacles (both warning and critical)

border of the image  $w_{img}$ . This is to steer the UAV away from the current scene as there is insufficient spaces for the UAV to move through safely.

After  $x^*$  is calculated, the steering angle,  $\psi^*$ , is determined using Eq. 5. The steering angle is positive in the clockwise direction, and vice versa. The desired heading reference,  $\psi_{desired}$  can be obtained by adding  $\psi^*$  to the current heading angle of the UAV,  $\psi_{heading}$  as shown in Eq. 6. Algorithm 1 presents the framework of the algorithm to stop the UAV and identify the steering angle.

$$\psi^* = \frac{x^* - \frac{1}{2}w_{img}}{\frac{1}{2}w_{img}} \times \frac{HFOV}{2}. \tag{5}$$

$$\psi_{desired} = \psi_{heading} + \psi^*. \tag{6}$$

### 4.4 Motion Control of the UAV

In this study, a straightforward approach in motion control is implemented to the flight controller to validate the proposed strategy. The UAV is programmed to move forward with a constant velocity from the starting position until the detection model triggers the UAV to stop. The height of the UAV is regulated to a constant height of 1.5 m during the flight using the feedback of sonar measurement. When a critical obstacle is detected, the UAV is commanded to turn to the desired heading angle. The angular velocity about the  $z_b$  axis,  $\omega_\psi$ , is controlled using Eq. 7:

$$\omega_\psi = K_p(\psi_{heading} - \psi_{desired}), \tag{7}$$

where the value of  $K_p$  is obtained to be 0.5 to achieve the desired heading angle in the shortest period. The  $z_b$  axis is perpendicular to the  $x_b y_b$ -coordinate plane and pointing downward, as presented in Fig. 5. Before the UAV can



**Algorithm 1** UAV steering decision framework.**Require:** Raw Image received from the UAV**Ensure:** Corresponding steering angle

- 1: Resize the image into  $426 \times 240$  pixels
- 2: Detect the presence of tree trunks in the images
- 3: Output the bounding boxes coordinates  $(x_{i_j, min}, y_{i_j, min}, x_{i_j, max}, y_{i_j, max})$
- 4: Determine the centroid coordinates for each bounding box
- 5: **for** Each obstacle detected **do**
- 6:     **if** An obstacle is a critical obstacle **then**
- 7:         Command the UAV to stop and hover at the same position
- 8:     Determine the largest obstacle-free space
- 9:     Calculate the corresponding steering angle
- 10:    **end if**
- 11: **end for**

continue to move forward, the current scene is examined to ensure there is no critical obstacle ahead. Note that, for safety reasons, there are some precaution steps taken during the real-world experiments. The UAV is controlled to move forward at a slow constant speed of  $0.1$  m/s to ensure that the expected image transmission delay does not affect the flight. Furthermore, when a critical obstacle is detected, the UAV first hovers and then turns according to the desired heading reference. It is to ensure that the image of the current scene is used in the computation of the heading reference for achieving precise navigation. In this paper, the experiments are carried out to validate the proposed obstacle avoidance strategy. The flight performance can be further enhanced using a more advanced platform in our future applications.

## 5 Result of the Obstacle Detection

### 5.1 Performance of the Different Scratch Models

This paper trained four scratch models as detectors with different specifications, as described in Section 3.1.1. This is to investigate the effect of the total number of training images and the number of convolutional layers on the performance of the detection models. The performance of each detector was tested using 60 images, which are not included in the training dataset of all the detectors. To evaluate the overall performance of the detectors, the precision-recall curve of each detector was plotted in Fig. 7.

From the result, Detector A achieves the best performance in terms of the average precision value among all the detectors. The maximum recall value of Detector A is up

to 0.76, which is the highest among the detectors. In other words, Detector A could detect more actual tree trunks in the test images, which can increase the success rate of detecting the critical obstacles in the image.

Hence, this shows that extensive training data can improve the performance of the network in the detection mission. Theoretically, the more training data, the better the detector performs. It is because the detector can learn more different features of the object of interest in the training process. Based on the training outcome, the detector is able to predict, localize, and categorize the object of interest accurately. Therefore, a detection model should be trained with more quantity, and better quality data as the noisy or blurry training image data will affect the training process and performance.

Besides, the effect of the number of convolutional layers on tree detection was investigated by comparing Detectors B, C, and D. The average precision of Detector B is the highest, followed by Detectors D and C. From the test results, Detector C has the lowest maximum recall, which indicates that it has the most inadequate tree trunk detection performance in terms of sensitivity. This shows that the additional convolutional layers do not increase the average precision of the detector. The model with two convolutional layers is the best choice in the tree trunk detection task, among other shallow architecture models.

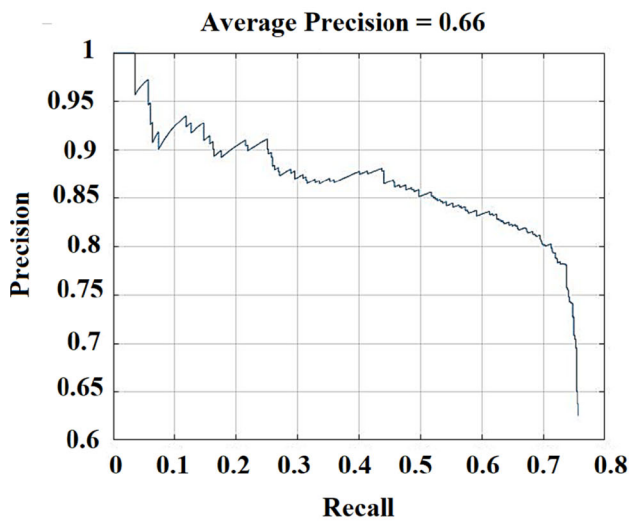
Therefore, one model with the best performance among the scratch models is selected and trained as the tree trunk detector for the next step. Based on the outcome of the results, the model with two convolutional layers is suggested to be trained with more image data to further improve its detection performance.

### 5.2 Comparison between the Scratch Model and Pre-trained Models

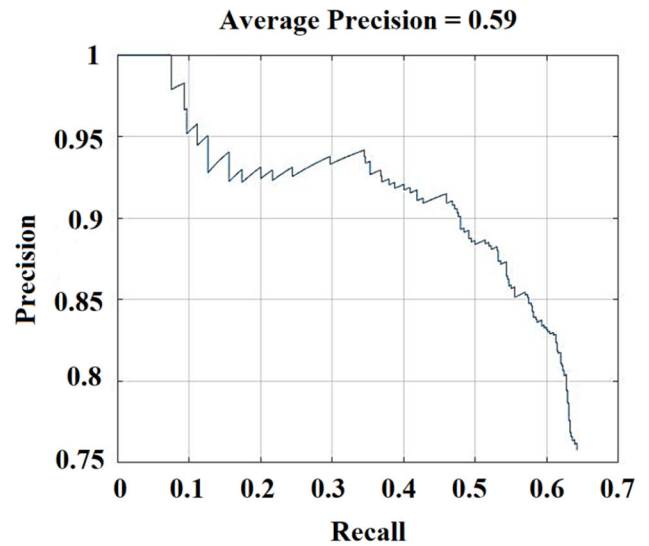
The scratch model selected in Section 5.1 and two pre-trained models, i.e., Resnet-50 and Inception v2, were trained with the same dataset. The dataset consists of images in different lighting conditions. The training dataset is a 12-minute flight video, which consists of 4689 images. These images contain a total of 23866 tree trunk annotations. After training, the models were tested with a 2-minute flight video, composed of 514 images in sequence recorded in the same field.

Figure 8 shows some random examples of the detection results on the test images. In this figure, the output images show that both pre-trained models are able to detect almost all the tree trunks, even with dense shadows or different lighting conditions. The majority of the output boxes cover the full size of the tree trunks in the test images shown in Fig. 8. There is also no obvious false positive in the result of both pre-trained models.

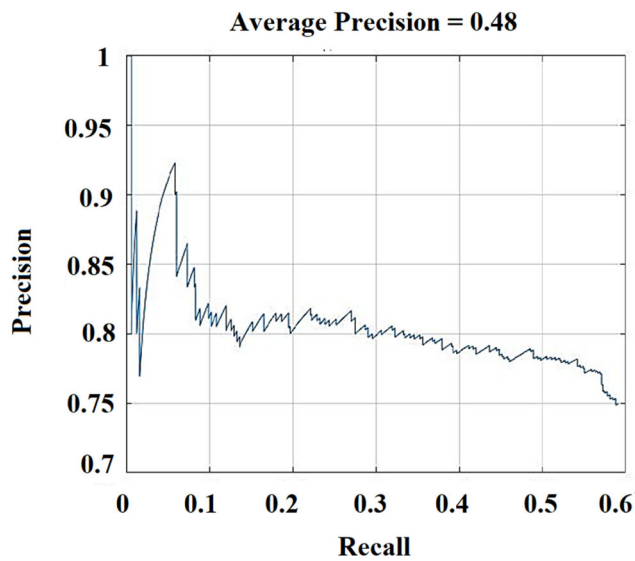




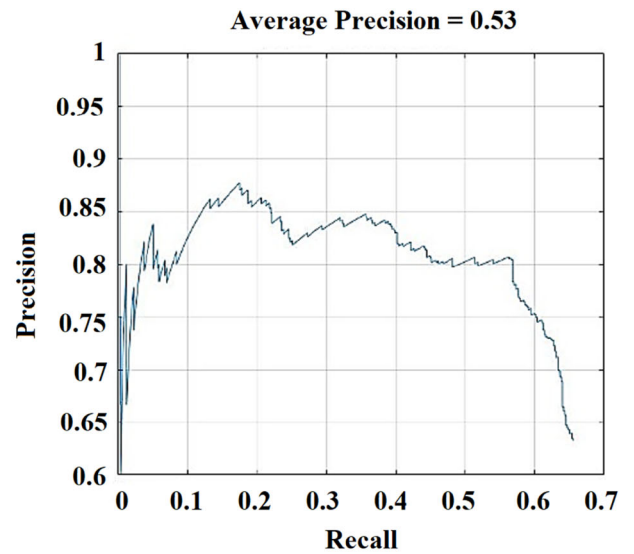
(a) Detector A



(b) Detector B



(c) Detector C



(d) Detector D

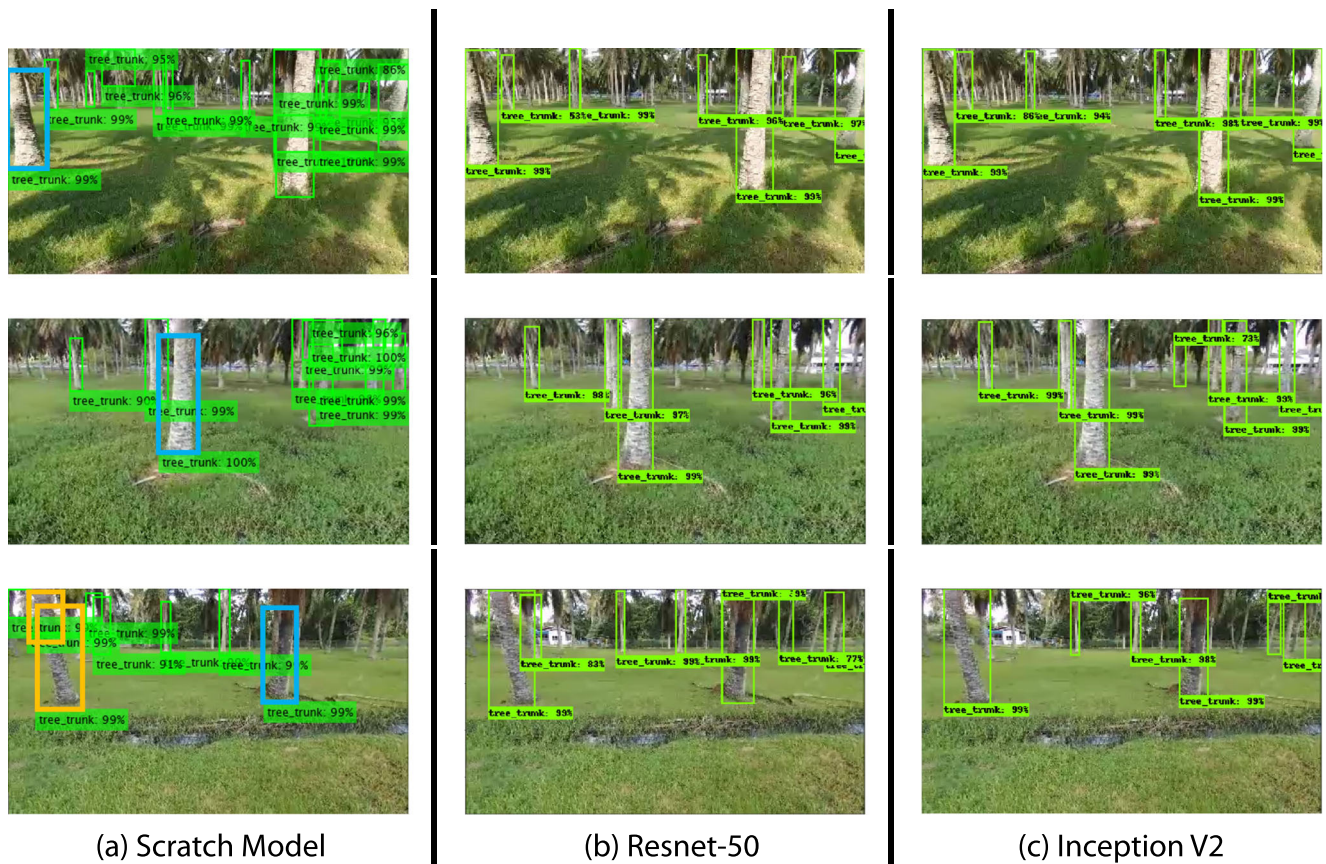
Fig. 7 Precision-recall curves of the test results

On the other hand, the scratch model outputs false-positive boxes with high confidence scores around the background in the test images. Some of the bounding boxes from the scratch model did not cover the full size of a tree trunk as visualized with the blue boxes in Fig. 8. Furthermore, most of the large obstacles in the images were bounded with two separate bounding boxes marked with orange boxes.

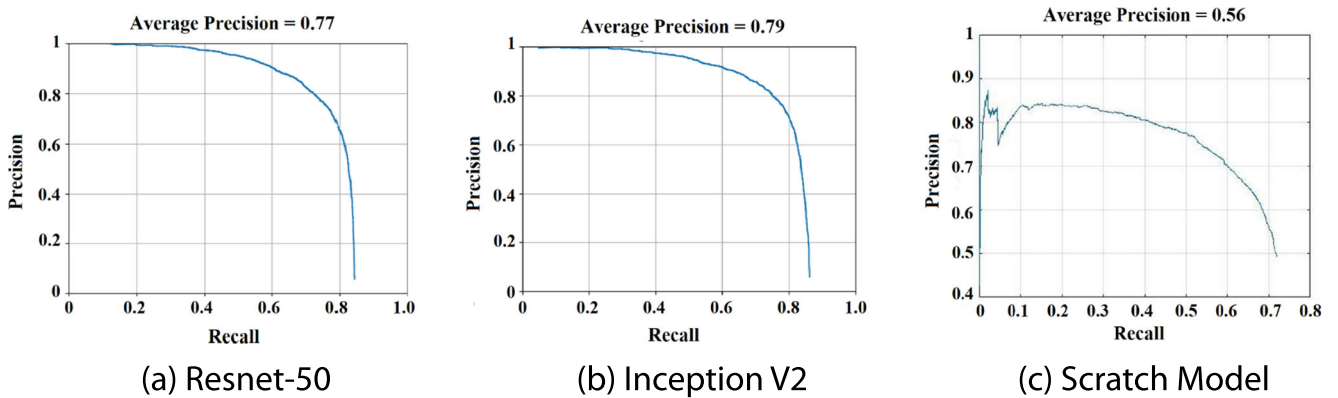
The bounding accuracy is crucial as the obstacle avoidance strategy uses its height as the depth clue of the obstacle from the UAV. If an approaching obstacle is bounded with two separate boxes or not fully covered, the strategy will not

identify it as a critical obstacle. As a result, the UAV will collide with the obstacle. Hence, it may not be practical to adopt this scratch model in the proposed strategy due to its output quality.

Figure 9 presents the precision-recall curve of the scratch model and pre-trained models. Both pre-trained models achieved a higher maximum recall compared to the scratch model. In other words, the pre-trained models are able to predict bounding boxes around the tree trunks more accurately and will not miss any approaching obstacles. Table 3 shows the average precision and detection speed of the scratch model and pre-trained models. The average



**Fig. 8** Random examples of the tree trunk detection results on the test images. Each output box is indicated by a green box with the category label and confidence score. The blue boxes represent the bounding boxes that did not fully cover the obstacle, whereas the orange boxes indicate that the model detects an obstacle as two separate ones



**Fig. 9** Precision- Recall Curve plotted to evaluate the performance of the scratch model and pre-trained models

**Table 3** Differences in parameters among four detectors trained

Parameters	Pre-trained Models		Scratch model
	Resnet-50	Inception V2	
Average Precision	0.77	0.79	0.56
Detection Speed per Frame (s)	2.02	0.36	1.3





**Fig. 10** Examples of safe condition that the UAV is allowed to move forward. The white line represents the direction of the optimum heading and remains in the center of the image as there is no critical obstacle ahead and the UAV is safe to continue its forward motion

precision values of Resnet-50 and Inception V2 are 27% and 29% higher than the scratch model, respectively.

Although Resnet-50 performed better than Inception v2 on ImageNet validation, the average precision of the Faster R-CNN with Inception v2 is slightly higher than that of Resnet-50 on the tree trunk detection. Furthermore, by using the same hardware resources, due to the wide architecture of Inception v2, the detection speed of Inception is 9.5 times faster than Resnet-50.

Therefore, Faster R-CNN with Inception v2 is selected to be implemented as the detector in our obstacle avoidance system because the model can output detection boxes accurately and rapidly. In addition, this setup outputs bounding boxes, which fully cover the obstacles, especially the critical obstacles along the UAV path.

## 6 Obstacle Avoidance Result

### 6.1 Flight Test Setup

To evaluate the system performance, the proposed method is implemented to control a Parrot Bebop 2. The Bebop 2 is equipped with an Inertial Measurement Unit (IMU), a sonar, an altimeter, a magnetometer, a Global Positioning System

(GPS), and two cameras facing forward and downward. The forward-looking camera is of particular interest to us for the obstacle avoidance purpose. It is activated during the flight tests to capture images at 60 FPS. The sonar is used to maintain a constant height of 1.5 m throughout the trajectory. The attitude angles and GPS coordinates of the Bebop 2 are logged during the flights for validation purposes.

Bebop autonomy driver<sup>1</sup> in the Robot Operating System (ROS) is used to communicate with the Software Development Kit (SDK)<sup>2</sup> of the Bebop 2. The communication of the UAV is based on a 2.4 GHz or 5 GHz Wi-Fi signal. This system is run within the ROS Kinetic environment on the Ubuntu 16.04. The ground control station uses an Intel Core i7-7500U MB CPU with a 32GB RAM and an Nvidia 940MX to perform extensive mathematical computations in the detection algorithm.

The exact locations of the two tested fields are shown in Fig. 2. The first location is the field where the training data was collected previously. After obtaining the success in Location 1, flight tests will be performed by the same UAV in another tree field. This scene is different from the

<sup>1</sup>ROS Bebop Driver: <https://bebop-autonomy.readthedocs.io/en/latest/>

<sup>2</sup>Parrot Bebop SDK: <https://developer.parrot.com/docs/SDK3/>



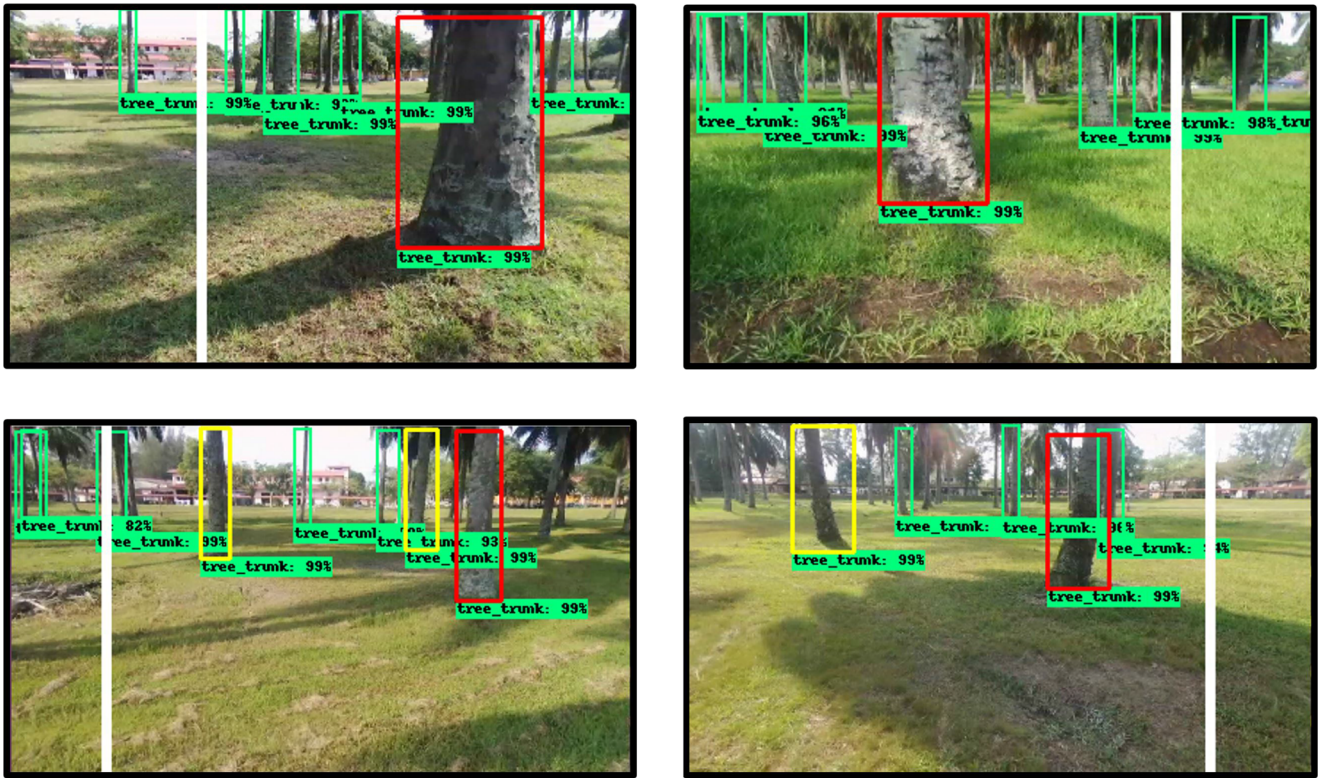


Fig. 11 Examples of the detection result when a critical obstacle was detected. The white line shows the desired heading direction for the UAV with the widest free space in the image

training dataset to validate the robustness of the obstacle avoidance system. The background of the area is different from the training images, which is surrounded by corridors and buildings. In both locations, the distances among the trees are sparse enough to provide an obstacle-free space for the UAV.

### 6.2 Detection Results during Flight Tests

The height ratio of each bounding box is used to indicate whether the detected obstacles are identified as critical obstacles. In the flight tests, three different colors, i.e., green, yellow, and red, represent whether the obstacles

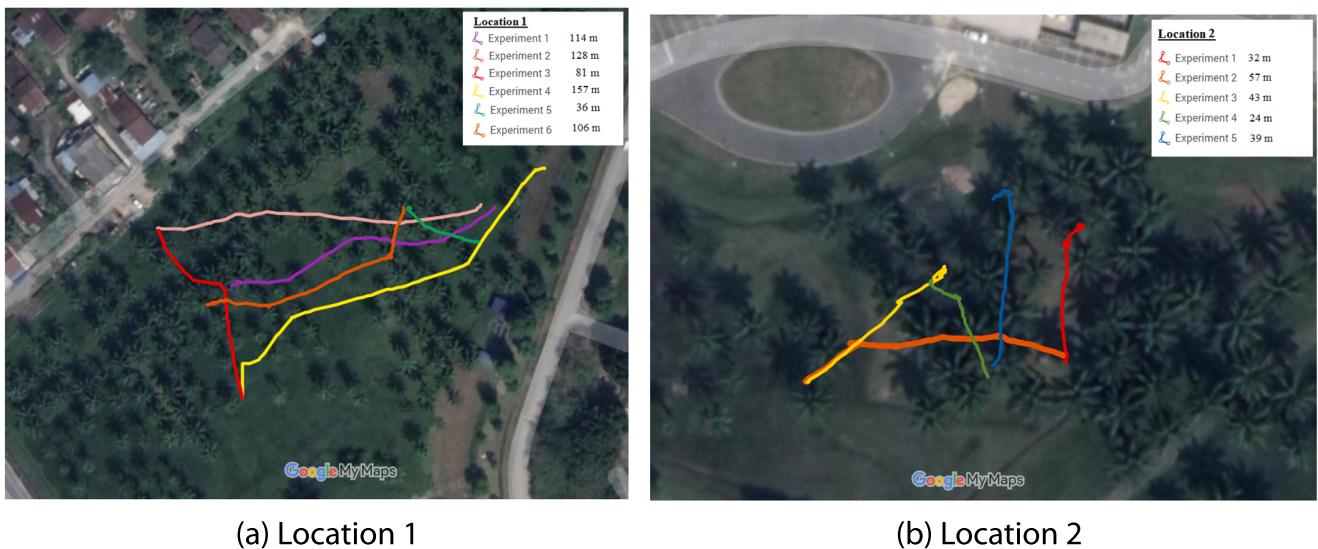
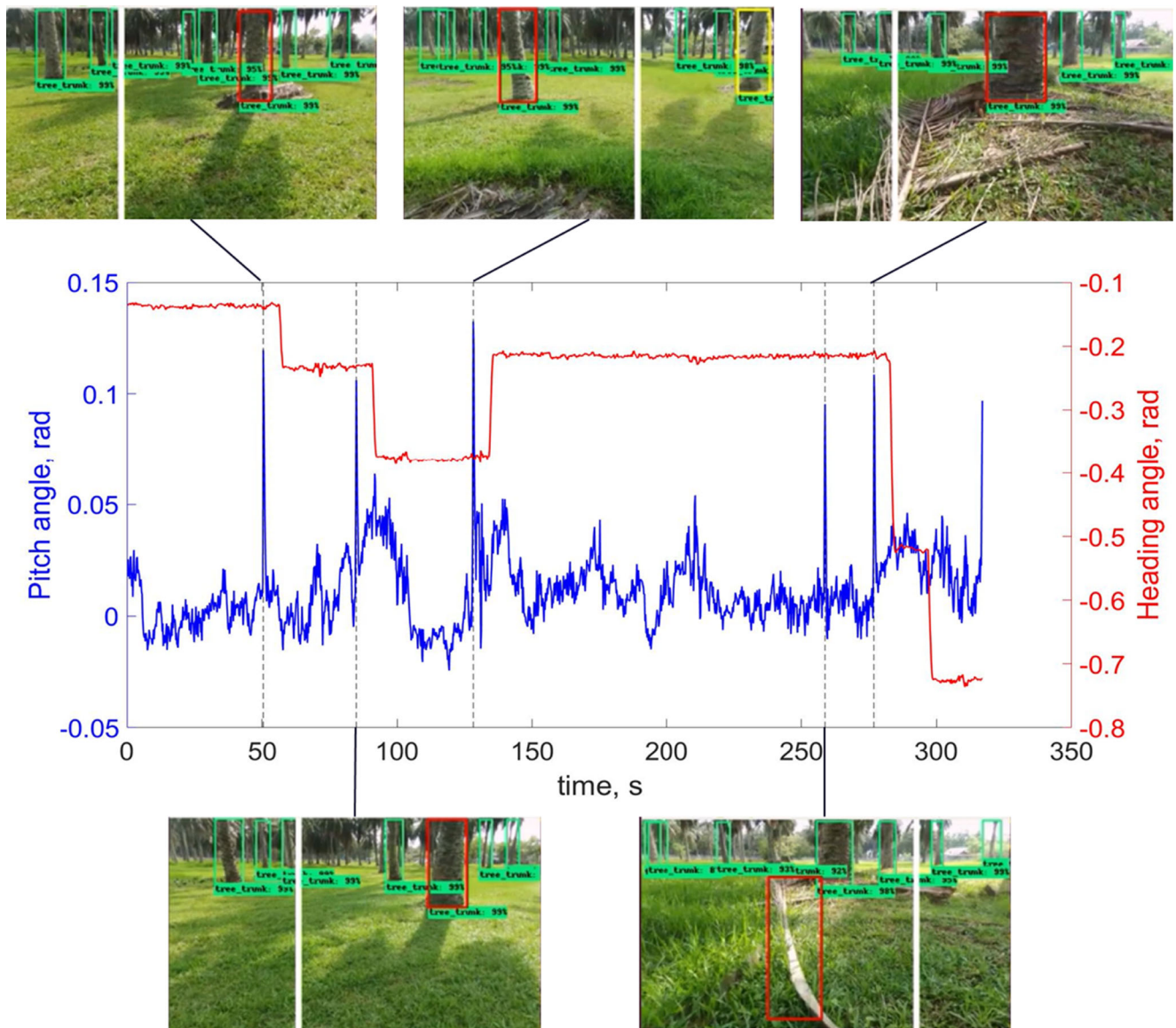
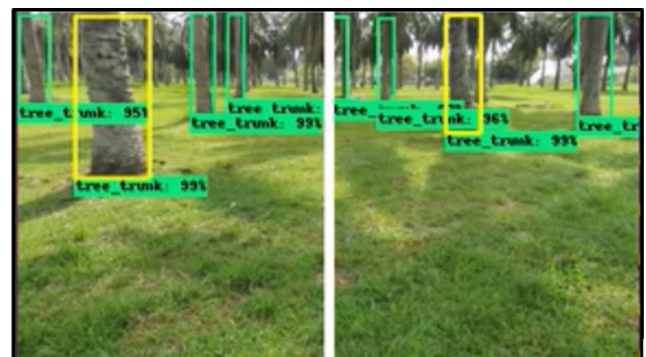


Fig. 12 The GPS routes of the UAV travelled in each autonomous navigation experiment

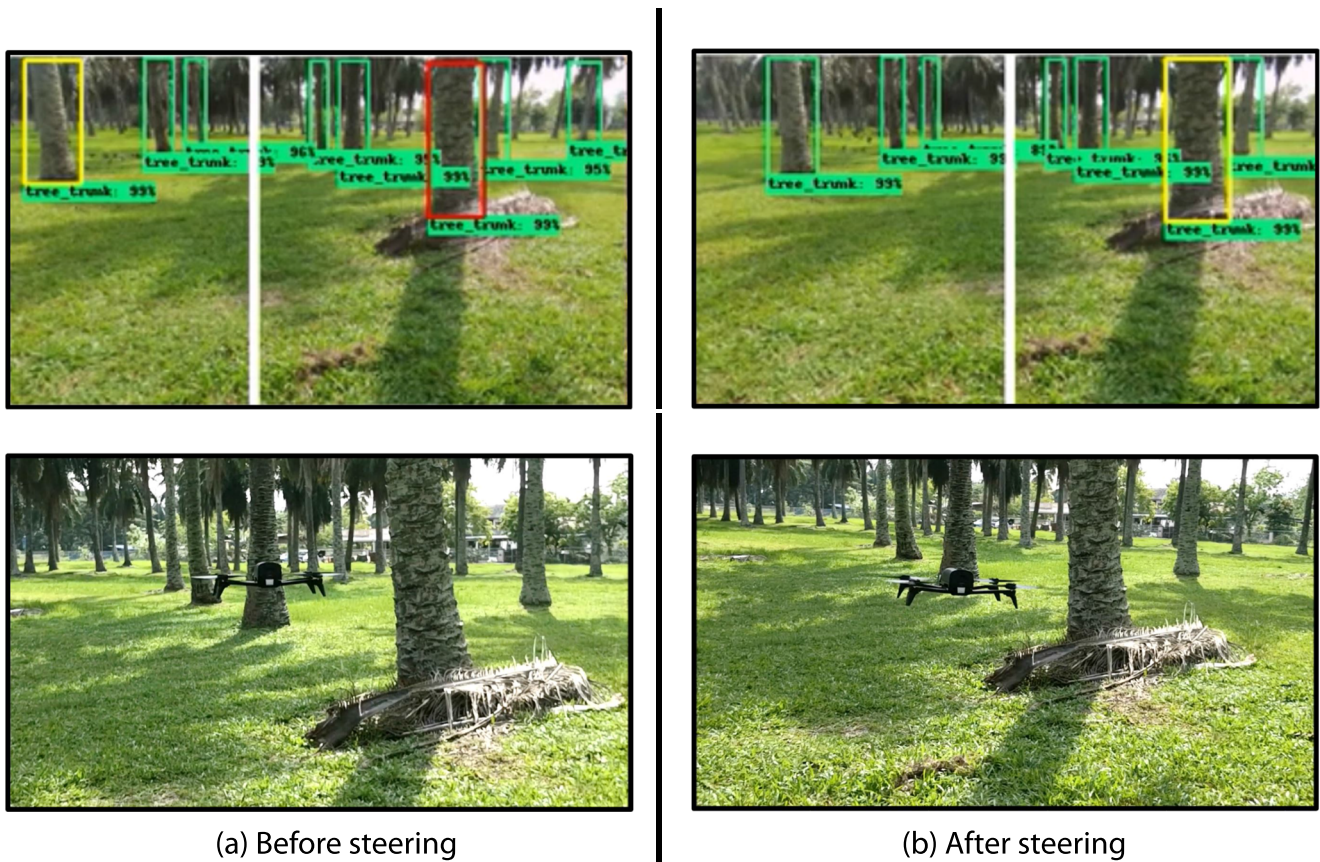


**Fig. 13** The pitch and heading angles of the UAV in one of the experiments, indicated by the blue and red lines, respectively. The images that triggered the system to stop in each case are also attached. The red bounding boxes in the images represent the detected critical obstacles

**Fig. 14** The detection result at the starting point. The desired heading reference indicated with the white line is at the center of the image allowing the UAV to navigate forward. The yellow boxes represent the warning obstacles that are approaching but not critical to stop the UAV







(a) Before steering

(b) After steering

**Fig. 15** The images on the top show the detection result, and the images on the bottom show the UAV from an observer's view **a** before and **b** after the UAV steered when the system detected a critical obstacle at 50.5 s

are far, near, or approaching to the UAV, respectively. For example, the green bounding boxes means the obstacles are far away. The yellow bounding boxes appears when the UAV is approaching trees but are still outside the 1 m safety boundary. They are so-called warning obstacles.

The purpose of introducing warning obstacles is to serve as an alert that an obstacle is near to the UAV. This factor is considered when the system calculates the heading direction after a critical obstacle is detected. However, it is not used to trigger the system to stop the UAV or change its heading. If it is free from critical obstacles, the heading remains, indicated by a white line at the center of the image. This shows that the UAV is safe for the forward motion in those scenarios, as shown in Fig. 10.

When the critical obstacle is detected, the obstacle is bounded with a red box in the image. Then, the system determines a new desired heading direction by calculating the widest free space. Figure 11 shows the desired heading determined in each frame where both red and yellow boxes are detected. The updated position of the white line in this figure shows that the system successfully identifies the critical obstacle and determines a new direction that is free from obstacles. This strategy not only steers the UAV

away from the critical obstacle but also guides the UAV navigating to the direction with the least number of frontal obstacles.

### 6.3 Avoidance Control during Flight Tests

Figure 12 shows the trajectory routes of the UAV plotted on the map with the GPS coordinates collected in 11 experiments at two different locations. The distances travelled in each experiment are also shown in this figure.

In all the experiments, the UAV successfully avoided all the frontal critical obstacles without any accident during the flight.<sup>3</sup> Although Location 2 is a new environment for the detector, the UAV could still safely avoid all the critical obstacles encountered during the flights. This shows that the system is robust and able to be adopted in the autonomous navigation of the UAV in other oil palm fields with different backgrounds and characteristics. The UAV was able to safely navigate in all the flight experiments until the program was stopped manually by the user.

<sup>3</sup>An introductory video with flight tests: <https://youtu.be/-3LuxDCJ5jk>

Besides, its heading and pitch angles were logged to investigate the UAV motion during the experiments, as shown in Fig. 13. In this graph, the sudden peaks of the pitch angle indicate that the UAV was stopped at that particular moment. This is because the UAV tends to pitch up at the instant when it is stopped. The corresponding images are also attached in Fig. 13. After the sudden peaks, the heading angles of the UAV changes in the figure as the UAV was steered to a new desired heading to avoid the critical obstacle. It can be clearly seen that the detection model can identify the critical obstacles and determine the new desired headings, even in the presence of dense shadows.

In this experiment, the UAV travelled 113 m autonomously in 5 minutes and 46 seconds. During the flight, the UAV made a total of 5 steering actions to avoid obstacles. After each stop, the system determined the new heading direction based on the space containing the least obstacle. This made the UAV navigation smoother without spending excessive time to stop at the position before making the steering decision.

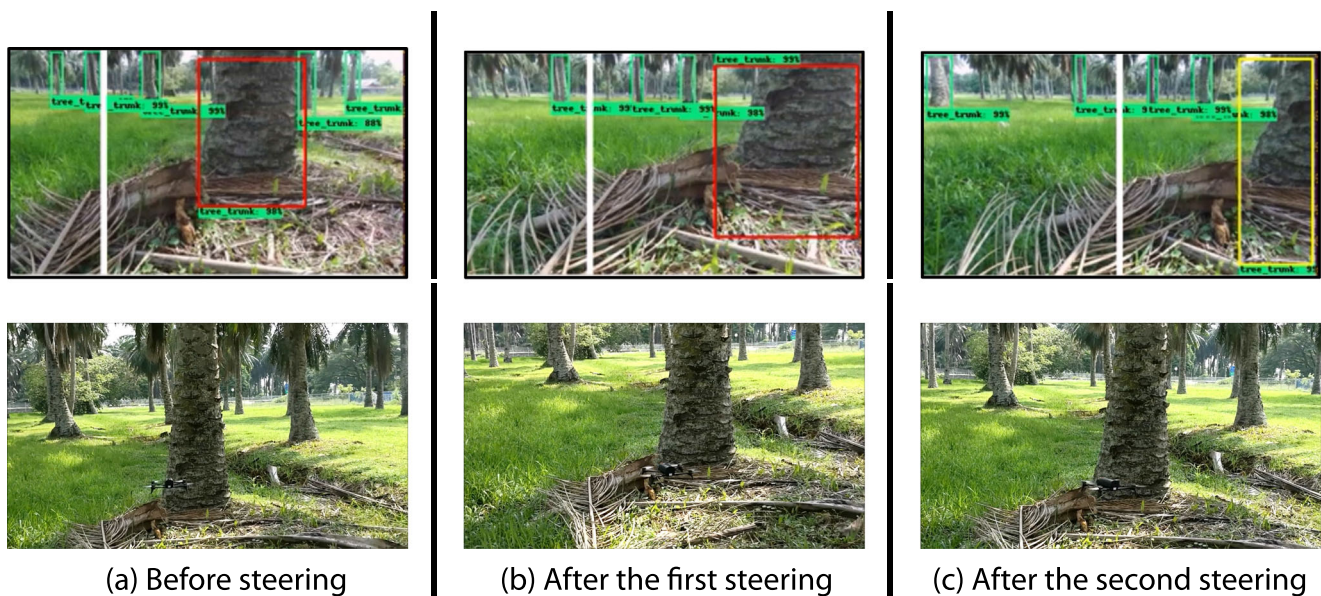
Figure 14 shows the detection result at the starting position. There were two warning obstacles identified and marked with yellow boxes but they did not trigger the UAV to stop. The desired heading reference remained at the center, which indicates that the forward motion was allowed. The UAV was navigated forward to begin the autonomous operation.

At time 50.5 s, a critical obstacle was detected, as shown in Fig. 15. The system stopped the UAV and decided to turn

7 deg counterclockwise based on the desired heading angle, corresponding to the direction with the widest obstacle-free space. Similarly, in the case at time 84.89 s and 128.2 s, the UAV was triggered to stop and turned to a new direction as the critical obstacle was detected in front. The steering angle was calculated by finding the maximum obstacle-free space.

At 276.8 s, the UAV navigated at a much lower height, even though its height was controlled at a constant desired value using sonar measurement feedback. This is due to the fact that the height measurement was affected by the soft mud area, causing the sensor to measure a distance larger than its actual value. Thus, the controller descended the UAV to a lower height. Figure 16 shows the obstacle detection and avoidance results at that time during the flight. The tree trunk appeared to be short in the image due to the current UAV height. This caused the UAV to stop when the tree trunk was very close to the UAV. However, the UAV was able to stop in front of the obstacle at around 0.7 m, which is still outside of the safety boundary.

Additionally, the UAV also adjusted its heading angle twice, due to its limited field of view, to successfully avoid obstacle collisions, as can be seen in Fig. 16. After the first yawing of -19 deg, the obstacle was still identified as the critical obstacle blocking its forward motion. Hence, the UAV was commended for steering another -13 deg. After the second yawing, the detection result showed that the obstacle was away from the UAV heading direction. Hence, the UAV was allowed to continue its forward motion.



**Fig. 16** The images on the top show the detection result after the UAV stopped at 276.8 s, and the images on the bottom show the UAV from an observer's view **a** before steering, **b** after the first and **c** the second

steering of the UAV, respectively. In the image **c**, there is no critical obstacle in the image. Thus, the desired heading direction is at the center of the image



## 7 Conclusion

In this paper, a novel approach for autonomous navigation of UAV in the tree plantation environment was presented. This approach eliminates the dependency of the external sensors for detecting the presence of obstacles. The system utilizes the on-board camera to obtain the frontal images and predicts the location of the obstacles on a frame-to-frame basis. The model outputs the coordinates of the detected obstacles in the images to control the UAV.

A pre-trained model, Faster R-CNN with Inception v2, as the convolutional base is selected. This is because the model is the best detection model for this application in terms of average precision and processing period, compared to Resnet-50 and a scratch detection model under the limited training data. The height of the bounding box is used to indicate the distance of the UAV from the detected obstacle. In addition to the sense-and-avoid approach, the system determines the desired heading direction with the widest obstacle-free space to continue the flight.

The ability and performance of the autonomous navigation system for the UAV in tree plantations were verified in the real flight tests in two different locations. In all 11 flight tests, the UAV successfully detected all the tree trunks that were dangerous to its navigation and performed the avoidance manoeuvre autonomously, even when the flying height did not remain the same during flights.

**Acknowledgements** The corresponding author would like to thank Universiti Sains Malaysia (USM) for providing the Short Term Research Grant Scheme (304/PAERO/6315113).

## Compliance with Ethical Standards

**Conflict of interests** The authors declare that they have no conflict of interest.

## References

- Agrawal, P., Ratnoo, A., Ghose, D.: Inverse optical flow based guidance for UAV navigation through urban canyons. *Aerosp. Sci. Technol.* **68**, 163–178 (2017)
- Aguilar, W.G., Casaliglla, V.P., Pólit, J.L.: Obstacle avoidance based-visual navigation for micro aerial vehicles. *Electronics* **6**(1), 10 (2017)
- Alvarez, H., Paz, L.M., Sturm, J., Cremers, D.: Collision avoidance for quadrotors with a monocular camera. In: *Experimental Robotics*, pp. 195–209. Springer (2016)
- Bachrach, A., He, R., Roy, N.: Autonomous flight in unknown indoor environments. *Int. J. Micro Air Veh.* **1**(4), 217–228 (2009)
- Bauer, P., Hiba, A., Bokor, J., Zarandy, A.: Three dimensional intruder closest point of approach estimation based-on monocular image parameters in aircraft sense and avoid. *J. Intell. Robot. Syst.* **93**(1–2), 261–276 (2019)
- Bills, C., Chen, J., Saxena, A.: Autonomous MAV flight in indoor environments using single image perspective cues. In: *Robotics and automation (ICRA)*, 2011 IEEE international conference, pp. 5776–5783. IEEE (2011)
- Canziani, A., Paszke, A., Culurciello, E.: An analysis of deep neural network models for practical applications. arXiv:1605.07678 (2016)
- Chebroly, N., Läbe, T., Stachniss, C.: Robust long-term registration of UAV images of crop fields for precision agriculture. *IEEE Robot. Autom. Lett.* **3**(4), 3097–3104 (2018)
- Chong, K.L., Kanniah, K.D., Pohl, C., Tan, K.P.: A review of remote sensing applications for oil palm studies. *Geo. Spat. Inf. Sci.* **20**(2), 184–200 (2017)
- Cui, J.Q., Lai, S., Dong, X., Chen, B.M.: Autonomous navigation of UAV in foliage environment. *J. Intell. Robot. Syst.* **84**(1–4), 259–276 (2016)
- Daftry, S., Zeng, S., Khan, A., Dey, D., Melik-Barkhudarov, N., Bagnell, J.A., Hebert, M.: Robust monocular flight in cluttered outdoor environments. arXiv:1604.04779 (2016)
- Eresen, A., Mamolu, N., Efe, M.N.: Autonomous quadrotor flight with vision-based obstacle avoidance in virtual environment. *Expert Sys. Appl.* **39**(1), 894–905 (2012)
- Esrafilian, O., Taghirad, H.D.: Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM. In: *Robotics and Mechatronics (ICROM)*, 2016 4th International Conference, pp. 240–245. IEEE (2016)
- Gageik, N., Benz, P., Montenegro, S.: Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors. *IEEE Access* **3**, 599–609 (2015)
- Girshick, R.: Fast r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448 (2015)
- Giusti, A., Guzzi, J., Ciresan, D.C., He, F.L., Rodríguez, J.P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G.: A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot. Autom. Lett.* **1**(2), 661–667 (2016)
- Gosiewski, Z., Ciesluk, J., Ambroziak, L.: Vision-based obstacle avoidance for unmanned aerial vehicles. In: *2011 4th International Congress on Image and Signal Processing (CISP)*, vol. 4, pp. 2020–2025. IEEE (2011)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
- Herwitz, S., Johnson, L., Dunagan, S., Higgins, R., Sullivan, D., Zheng, J., Lobitz, B., Leung, J., Gallmeyer, B., Aoyagi, M.: Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. *Comput. Electron. Agric.* **44**(1), 49–61 (2004)
- Ho, H., De Wagter, C., Remes, B., De Croon, G.: Optical-flow based self-supervised learning of obstacle appearance applied to MAV landing. *Robot. Auton. Syst.* **100**, 78–94 (2018)
- Hoo-Chang, S., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R.M.: Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Trans. Med. Imaging* **35**(5), 1285 (2016)
- Hui, X., Bian, J., Yu, Y., Zhao, X., Tan, M.: A novel autonomous navigation approach for UAV power line inspection. In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 634–639. IEEE (2017)
- Huuskonen, J., Oksanen, T.: Soil sampling with drones and augmented reality in precision agriculture. *Comput. Electron. Agric.* **154**, 25–35 (2018)
- Iacono, M., Sgorbissa, A.: Path following and obstacle avoidance for an autonomous UAV using a depth camera. *Robot. Auton. Syst.* **106**, 38–46 (2018)

25. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 (2015)
26. Kanellakis, C., Nikolakopoulos, G.: Survey on computer vision for UAVs: Current developments and trends. *J. Intell. Robot. Syst.* **87**(1), 141–168 (2017)
27. Kim, D.K., Chen, T.: Deep neural network for real-time autonomous indoor navigation. arXiv:1511.04668 (2015)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105 (2012)
29. Lee, J.O., Lee, K.H., Park, S.H., Im, S.G., Park, J.: Obstacle avoidance for small UAVs using monocular vision. *Aircr. Eng. Aerosp. Technol.* **83**(6), 397–406 (2011)
30. Lelong, C.C., Burger, P., Jubelin, G., Roux, B., Labbé, S., Baret, F.: Assessment of unmanned aerial vehicles imagery for quantitative monitoring of wheat crop in small plots. *Sensors* **8**(5), 3557–3585 (2008)
31. Liu, P., ElGeneidy, K., Pearson, S., Huda, M.N., Neumann, G., et al.: Towards real-time robotic motion planning for grasping in cluttered and uncertain environments. In: *Towards Autonomous Robotic Systems: 19th Annual Conference, TAROS 2018, Bristol, UK July 25–27, 2018, Proceedings*, vol. 10965, p. 481. Springer (2018)
32. Liu, P., Yu, H., Cang, S.: Adaptive neural network tracking control for underactuated systems with matched and mismatched disturbances. *Nonlinear Dyn.* **98**(2), 1447–1464 (2019)
33. Liu, Z., Zhang, Y., Yuan, C., Ciarletta, L., Theilliol, D.: Collision avoidance and path following control of unmanned aerial vehicle in hazardous environment. *J. Intell. Robot. Syst.*, 1–18 (2018)
34. Ma, Z., Wang, C., Niu, Y., Wang, X., Shen, L.: A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles. *Robot. Auton. Syst.* **100**, 108–118 (2018)
35. Mancini, M., Costante, G., Valigi, P., Ciarfuglia, T.A.: Fast robust monocular depth estimation for obstacle detection with fully convolutional networks. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference*, pp. 4296–4303. IEEE (2016)
36. Mancini, M., Costante, G., Valigi, P., Ciarfuglia, T.A.: J-mod 2: Joint monocular obstacle detection and depth estimation. *IEEE Robot. Autom. Lett.* **3**(3), 1490–1497 (2018)
37. Matthew, D., Fergus, R.: Visualizing and understanding convolutional neural networks. In: *Proceedings of the 13th European Conference Computer Vision and Pattern Recognition, Zurich, Switzerland*, pp. 6–12 (2014)
38. Nieuwenhuisen, M., Droschel, D., Beul, M., Behnke, S.: Autonomous navigation for micro aerial vehicles in complex gnss-denied environments. *J. Intell. Robot. Syst.* **84**(1–4), 199–216 (2016)
39. Qu, T., Zhang, Q., Sun, S.: Vehicle detection from high-resolution aerial images using spatial pyramid pooling-based deep convolutional neural networks. *Multimed. Tools Appl.* **76**(20), 21651–21663 (2017)
40. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in Neural Information Processing Systems*, pp. 91–99 (2015)
41. Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive UAV control in cluttered natural environments. arXiv:1211.1690 (2012)
42. Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C.: Deepfruits: A fruit detection system using deep neural networks. *Sensors* **16**(8), 1222 (2016)
43. Schauwecker, K., Zell, A.: On-board dual-stereo-vision for the navigation of an autonomous MAV. *J. Intell. Robot. Syst.* **74**(1–2), 1–16 (2014)
44. Serres, J.R., Ruffier, F.: Optic flow-based collision-free strategies: From insects to robots. *Arthropod Struct. Dev.* **46**(5), 703–717 (2017)
45. Shyam, R.A., Lightbody, P., Das, G., Liu, P., Gomez-Gonzalez, S., Neumann, G.: Improving local trajectory optimisation using probabilistic movement primitives. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2666–2671 (2019)
46. Wang, X., Cheng, P., Liu, X., Uzochukwu, B.: Fast and accurate, convolutional neural network based approach for object detection from UAV. In: *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 3171–3175. IEEE (2018)
47. Xie, L., Wang, S., Markham, A., Trigoni, N.: Towards monocular vision based obstacle avoidance through deep reinforcement learning. arXiv:1706.09829 (2017)
48. Yao, H., Yu, Q., Xing, X., He, F., Ma, J.: Deep-learning-based moving target detection for unmanned air vehicles. In: *2017 36th Chinese Control Conference (CCC)*, pp. 11459–11463. IEEE (2017)
49. Zufferey, J.C., Floreano, D.: Fly-inspired visual steering of an ultralight indoor aircraft. *IEEE Trans. Robot.* **22**(1), 137–146 (2006)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**H. Y. Lee** received her B.Eng. degree in Aerospace Engineering from Universiti Sains Malaysia, Malaysia in 2019. She has obtained Dean's List awards for all semesters during her Bachelor studies. Her research interests are machine learning, robot control, and computer vision. She is currently an industrial engineer at Globalfoundries in Singapore.

**H. W. Ho** received his B.Eng. degree in Aerospace Engineering from Universiti Sains Malaysia, Malaysia in 2009 and M.Sc. degree in Aerospace Engineering (cum laude), specialized in Control and Simulation from Delft University of Technology, the Netherlands in 2012. He obtained his Ph.D. degree in Aerospace Engineering at the Micro Air Vehicle lab (MAV-lab) of Delft University of Technology in 2017, with the topic of autonomous landing of Micro Air Vehicles through bio-inspired monocular vision. His research interests include control of MAVs, vision-based control strategies of MAVs, machine learning, computer vision, and state estimation. Currently, he is senior lecturer and UAV lab manager at the School of Aerospace Engineering of Universiti Sains Malaysia, Malaysia, and guest researcher at the Faculty of Aerospace Engineering of Delft University of Technology, the Netherlands.

**Y. Zhou** obtained her B.S. degree and M.S. degree in School of Mechanical & Electrical Engineering, Northwestern Polytechnical University, Xi'an, China, in 2010 and 2013, respectively, and a Ph.D. degree in Control and Simulation, Aerospace Engineering, Delft University of Technology in 2018. Her Ph.D. research topic was online reinforcement learning control for aerospace systems. Her research interests lie in nonlinear control, adaptive control, Reinforcement Learning, intelligent control, guidance, and navigation. She was a lecturer at Faculty of Aerospace Engineering, Delft University of Technology from 2017 to 2018. She is currently a senior lecturer at School of Aerospace Engineering of Universiti Sains Malaysia, and a guest researcher at the Faculty of Aerospace Engineering of Delft University of Technology.