algorithms

MDPI

Article

Testing a Vision-Based Autonomous Drone Navigation Model in a Forest Environment

Alvin Lee [1,†], Suet-Peng Yong [2,†], Witold Pedrycz [3] and Junzo Watada [4,*]

[1] Mercedes-Benz Malaysia, Puchong 47180, Selangor, Malaysia; alvinxiii3@gmail.com
[2] International Society of Management Engineers (ISME), 2-10-8-407 Kobai, Yahatanishi, Kitakyushu 806-0011, Japan; vspyong@gmail.com
[3] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6R 2V4, Canada; wpedrycz@ualberta.ca
[4] Faculty of Data Science, Shimonoseki City University, 2-1-1 Daigaku-cho, Shimonoseki, Yamaguchi 751-8510, Japan
* Correspondence: watada@waseda.jp; Tel.: +81-90-3464-4929
† These authors contributed equally to this work.

Abstract: Drones play a pivotal role in various industries of Industry 4.0. For achieving the application of drones in a dynamic environment, finding a clear path for their autonomous flight requires more research. This paper addresses the problem of finding a navigation path for an autonomous drone based on visual scene information. A deep learning-based object detection approach can localize obstacles detected in a scene. Considering this approach, we propose a solution framework that includes masking with a color-based segmentation method to identify an empty area where the drone can fly. The scene is described using segmented regions and localization points. The proposed approach can be used to remotely guide drones in dynamic environments that have poor coverage from global positioning systems. The simulation results show that the proposed framework with object detection and the proposed masking technique support drone navigation in a dynamic environment based only on the visual input from the front field of view.

Keywords: deep learning; masking vision; autonomous navigation; drone navigation

Citation: Lee, A.; Yong, S-P.; Pedrycz, W.; Watada, J. Testing a Vision-Based Autonomous Drone Navigation Model in a Forest Environment. *Algorithms* **2024**, *17*, 139. https://doi.org/10.3390/a17040139

Academic Editor: Xujiong Ye

Received: 30 January 2024
Revised: 1 March 2024
Accepted: 13 March 2024
Published: 27 March 2024

1. Introduction

An unmanned aerial vehicle (UAV), commonly known as a drone, is an aircraft without any human pilot on board. It can be operated either remotely by a human operator or autonomously by onboard computers. The use of drones has increased rapidly in various industrial applications, including agriculture, mining, search and rescue operations, surveillance, and recreation. The recent technological advancements in Industry 4.0 have led to several breakthroughs in the areas of computer vision, deep learning, and UAV control management operations. These advancements create more possibilities for drones to perform tasks in dynamic environments by analyzing their surroundings via computer vision technologies.

Generally, most commercial drones use global positioning systems (GPSs) to determine their location and continuously capture their 3D position. Hence, these drones can take advantage of differential GPS [1] and localization techniques [2] to improve their navigational accuracy. However, the absence of reliable GPSs in drones degrades their navigation functions because they cannot capture location information or navigate in a remote environment, for example, in a forest or jungle. Overall, the autonomous navigation of airborne vehicles remotely in complex environments is challenging because of the unreliability of infrastructure-based positioning systems. Locations with weak GPS signals due to poor coverage by the ground provider's satellite [3] severely degrade the performance of drones' vital functions, such as autonomous navigation [4].

Currently, forest inspection is conducted mainly via human patrol, which has safety, accuracy, and cost issues [5]. The interruptions and errors in flight control systems can cause severe safety problems, especially in autonomous [5] drone navigation systems.

The ability to navigate autonomously is an essential aspect of UAV autonomy. It allows UAVs to autonomously compute the best path from a start point to an end point. UAVs have to constantly change their trajectories depending on the particular terrains and conditions prevailing during their flight [6]. Therefore, a reliable control system is required for a drone to perform complex tasks like autonomous navigation in a dynamic environment without human intervention, which remains challenging. For this purpose, certain variables, such as the environment, objects, hardware, and algorithm, must be considered in developing a robust autonomous navigation system. Among these variables, the autonomous navigation of drones based on scene-understanding algorithms has received much attention in recent years [7]. Even though algorithms that can develop object detection technology have been explored for drones [8], complex tasks, such as acquiring contextual information from surrounding visual objects to determine a clear navigation path in the scene, have been neglected.

Obtaining a high-level semantic description for acquired scenes using only object detection and image processing algorithms is one of the difficulties for autonomous drone flight. Localizing a visual object from a moving camera is always a problem due to the camera movement, which results in a lack of reference points in the scene, affecting object detection and high-level scene modeling [7,9]. Furthermore, prior knowledge based only on the static context is unsuitable for understanding a scene in a dynamic environment. For example, for autonomous flight among trees in a forest, determining methods to detect drones and avoid trees while flying autonomously along a clear path is the research problem of this paper. For this purpose, object detection and tracking become difficult when distinguishing between moving objects and environmental elements [10,11].

This paper addresses the problem of building a vision-based navigation model that would be virtually the only option for an autonomous drone to maintain its functionality in the absence of reliable positioning systems and sensors other than a 2D visual camera. We consider a scenario where a drone flies through a forest for an inspection mission. The optical positioning module takes the input from the front-view video camera of the drone, extracts the visual object information of each frame, and feeds this information to the proposed empty space-masking algorithm to guide the drone position in the visual scene's context, as shown in Figure 1.

Figure 1. Sample frame from the drone vision system depicting the detected objects and the guided localization point in the frame.

In this study, a deep learning-based model is adopted for object identification, and the model is pre-trained offline with a set of tree objects that appear in the context of the particular application. The dynamics of high-level semantic representations are learned from video streams via state-of-the-art object detection networks. The spatial relationships among objects are extracted to form segments in the scene. A field of view is used to pass through the segments as a masking process to identify empty space for the drone path.

Our approach enables drones to self-navigate, especially in remote environments. The scenario used in this study is based on a front-view drone camera, which is different from those in other studies that are more focused on the aerial or top view.

In summary, the proposed study contributes the following:

- The localization of visual objects is performed via object detection to give the algorithms we parse in Section 3 contextual information of the scene for further processing. In Section 3, scenes are described using color-based segmentation with a masking approach to provide a comprehensive semantic representation of their relative environments.
- The vision-based navigation model can autonomously guide a drone in a dynamic and remote environment.
- The masking model is innovatively employed on the basis of vision to autonomously navigate the drone in a remote environment and without any connection with the center.
- The masking model is innovatively used based on the vision to autonomously navigate.
- A drone can navigate in remote environments without any connection with the center.

This paper is structured as follows: Section 2 provides an overview of the technical challenges encountered in vision-based navigation systems and examines related research efforts. The algorithm and data pertaining to the proposed vision-based navigation model are detailed in Section 3. Subsequently, Section 4 showcases the simulation results of drone pathing based on the proposed model, while Section 5 offers concluding remarks for the paper.

2. Related Works

In unfamiliar environments, achieving vision-based autonomous navigation poses significant challenges due to limited knowledge of the surroundings. Consequently, vehicles can only devise paths that optimize their existing knowledge. However, leveraging scene information obtained from a vision sensor can provide the necessary data for navigating through unknown terrains [12]. In UAV applications, real-time navigation heavily relies on the data collected by vision systems. Object detection stands as a crucial element of autonomous systems, particularly in route and path planning. By identifying obstacles in the vehicle's vicinity, a safe navigation route can be determined. However, detecting small objects within the environment presents a formidable challenge. Despite the availability of various sensors like Radio Detection and Ranging (Radar) and Light Detection and Ranging (LiDAR) for navigation support, their performance often falters due to environmental factors [13]. As a viable alternative, vision-based sensors such as cameras find widespread use in autonomous vehicle navigation. Recent advancements and the evolution of deep learning-based solutions [14] have rendered the navigation of autonomous vehicles achievable.

Certain related methods, such as moving object detection and tracking [12,13], have been applied to solve the problems of video-based tracking. Some drone research [14,15] uses low-level vision-based solutions to solve object recognition problems. They focus on finding the thresholded color to track visual objects, but this approach has limited application for autonomous drones. Many factors, such as the surrounding objects, obstacles, remote environments, and the absence of GPS signals, should be considered to obtain a precise understanding of a scene. Some variables, such as the environment, objects, hardware, and algorithm, must be considered in the development of a robust autonomous navigation system. Among these variables, the autonomous navigation of drones based on scene-understanding algorithms has received much attention in recent years [7].

Several works have proposed ways of using deep networks for predicting object bounding boxes [16,17]. Deep networks for object detection, tracking, and scene understanding with drones have the potential to widen the application scope of autonomous drones. However, many existing deep learning-based solutions may not be generalized to challenging scenarios, such as those with occlusions and complex backgrounds. Object detection algorithms are often combined with other image processing techniques to improve their robustness and efficiency in many applications [18]. A sequence of dual background difference methods was proposed to identify the unattended objects in a scene [19]. The algorithm is based on finding the intensity between the current and reference background models within a time period. Similarly, the authors use an object detector as part of the implementation to identify unattended objects in the scene. Hence, combining object detection algorithms with other image processing techniques can be seen as a novel way to improve scene understanding.

There are some studies focusing on surveillance and inspection missions using UAVs [20–22]. Though they consider different problems, enabling autonomous drone navigation still remains the main challenge for the application of UAVs. Autonomous drone navigation has become an essential component of future industrial UAVs. It is essential to design a robust vision-based navigation model for the considered problems in autonomous drone navigation.

To acquire a better understanding of the environment, a drone model that can navigate through larger areas was presented in [23]. The reliability of autonomous drone navigation can only be improved by using GPS information which is equipped with a real-time landmark detection system. The 3D position of a UAV should be acquired based on GPS, as mentioned in [24]. Consequently, it is easy to localize a target based on the UAV camera information, such as the position and orientation [25]. However, GPS information is not always available due to certain obstacles for GPS signals, such as trees, buildings and adverse atmospheric conditions.

Semantic segmentation has been widely used in automated driving [26]. However, implementing semantic segmentation in a safety system environment that requires high levels of accuracy and robustness is a challenge. Nevertheless, semantic segmentation is useful for dividing an image or scene into different parts [27], and can be applied to an image consisting of unknown objects to find the overall region of interest in the scene.

The recent advances in object detection have been driven by the success of region proposal methods [28] and region-based convolutional neural networks (R-CNNs) [29]. In brief, an R-CNN is a combination of a region proposal network (RPN) and a convolutional neural network (CNN). A CNN is mainly used for image classification, while an RPN is used for generating region proposals. Ross et al. [30] proposed an object detection system with an R-CNN architecture. The R-CNN architecture consists of three modules. The first module generates region proposals to determine the corresponding features in the region, the second module uses a CNN to extract features from the proposals, and the third module adopts a class-specific linear support vector machine for object classification.

Various works demonstrate that deep network-based region proposal methods outperform certain previous works [28,29]. The state-of-the-art object proposal method, namely, Faster R-CNN [31], requires 300 proposals compared with the 2000 proposals produced by [32]. For certain reasons, R-CNN requires a forward pass of the CNN for every region proposal for every image. Additionally, R-CNN requires separate training for three different models, which makes it rather slow for object detection. Hence, Fast R-CNN [33] and Faster R-CNN [34] were proposed to solve these problems. In the RPN of Faster R-CNN, the time cost of generating region proposals is much shorter than that of Fast R-CNN [31]. Real-time and cloud-based object detection for UAV- and drone-based object counting by a spatially regularized regional proposal network has shown impressive results in detecting different types of objects in real time. However, these region proposal drone-based object detection methods use a top-view perspective, which has no obstacles in its field of view. In a scenario where the drone operates in a remote environment with

many background objects in the front view of the drone camera, the spatial relationships between the detected objects should be described to enable precise navigation of the drone.

The primary goal of our approach is to build an object detection algorithm by applying a state-of-the-art detector [31], combined with our proposed masking technique, to semantically interpret the scene. The semantic technologies facilitate building a high-level description of the environment and its elements based on the heterogeneous information gathered from different sources, unlike the drone-based solutions mentioned above, which use low-level image processing techniques and heavily rely on a GPS for detecting environmental features.

3. The Proposed Method

The proposed model for drone navigation based on visual information is shown in Figure 2. Principally, there are four main components in the model: object detection to find the trees that are obstacles and masking to identify the area that can be navigated, and waypoint guiding to navigate the drone.

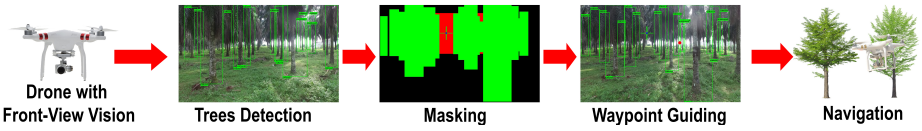


Figure 2. The complete flow for the proposed method.

The vision sensor or camera located on the drone is used in this study to identify the surrounding objects and sense the information in the environment. In short, the scene is captured as a video sequence, and is processed to extract the contextual features of the visual objects, while our proposed masking method is executed to understand the scene and provide a waypoint for guiding the navigation.

In an environment with dynamic brightness, flying a drone is a challenging task, especially for scene understanding. In our work, the masking method is applied to perform semantic segmentation to identify a safe route to navigate in the scene.

The first algorithm, represented in Algorithm 1, focuses on frame processing for drone navigation. It includes steps such as object detection, background adjustment, obstacle detection, spatial relationship analysis, and path determination. These steps collectively contribute to identifying clear paths for drone navigation based on the analysis of video frames. On the other hand, Algorithm 2 presents a different approach to drone navigation, incorporating object detection, semantic segmentation, thresholding for dynamic light conditions, and navigation based on contextual information derived from these processes. This algorithm emphasizes scene understanding through semantic segmentation and the identification of safe routes and bright spots for navigation.

Algorithm 1 Clear Path Determination Algorithm

```
function PROCESS_FRAME(frame)
    objects ← DETECT_OBJECTS(frame)
    frame ← ADJUST_BACKGROUND(frame)
    fov_frame ← ADD_FOV_REGION(frame)
    obstacle_boxes ← DETECT_OBSTACLES(objects, frame)
    clear_path ← ANALYZE_SPATIAL_RELATIONSHIP(fov_frame, obstacle_boxes)
    path ← DETERMINE_CLEAR_PATH(clear_path)
    return path
end function
function DETECT_OBJECTS(frame)
    objects ← OBJECT_DETECTION_ALGORITHM(frame)  ▷ Implementation depends on the detection model
    return objects
end function
function ADJUST_BACKGROUND(frame)
    return np.zeros_like(frame)
end function
function ADD_FOV_REGION(frame)
    fov_frame ← frame.copy()
    fov_frame[FOV_REGION] ← [255, 0, 0]  ▷ Red color
    return fov_frame
end function
function DETECT_OBSTACLES(objects, frame)
    obstacle_boxes ← [obj['bounding_box'] for obj in objects if obj['type'] == 'tree']
    return obstacle_boxes
end function
function ANALYZE_SPATIAL_RELATIONSHIP(fov_frame, obstacle_boxes)
    clear_path ← SPATIAL_ANALYSIS_ALGORITHM(fov_frame, obstacle_boxes)
    return clear_path
end function
function DETERMINE_CLEAR_PATH(clear_path)
    drone_path ← PATH_DETERMINATION_ALGORITHM(clear_path)
    return drone_path
end function
```

3.1. Object Detection

Every object in the scene is important to autonomous drone navigation, as the drones interpret and perform tasks based on object locations. As is currently common practice, deep learning is used for object detection in this project. However, the lack of standardized datasets that contain different types of trees in images acquired from drones makes it challenging for the deep learning models described here to create a drone-based solution.

The v2 Around NAS dataset used in this paper comprises images captured by a drone vision system. All the video data used here were recorded by a drone in the campus area of Petronas University of Technology and the nearby oil palm plantation. Ten videos of the investigation areas were recorded. The image sequences were extracted from the videos acquired and all the frames were manually annotated to train the object detection model with a deep learning method. Furthermore, to boost the performance of the model, the datasets were augmented through different processing techniques such as flips, grains, and shifts. A deep learning method with a convolution neural network was implemented using Tensor-Flow running on an Nvidia Telsa K80.

Algorithm 2 Drone Navigation Algorithm with Object Detection, Semantic Segmentation, Thresholding, and Navigation

```
function AUTONOMOUS_NAVIGATION(video_frames)
                                                                          ▷ Object Detection Module
    detected_objects ← OBJECT_DETECTION(video_frames)

                                          ▷ Semantic Segmentation and Thresholding Modules (Parallel Execution)
    safe_route ← SEMANTIC_SEGMENTATION(detected_objects)
    bright_spots ← THRESHOLDING(video_frames)

                                                                          ▷ Navigation Module
    NAVIGATE_AUTONOMOUSLY(safe_route, bright_spots)
end function
function OBJECT_DETECTION(video_frames)
    detected_objects ← []          ▷ Placeholder, actual implementation depends on the detection model
    for frame in video_frames do
                                  ▷ Process each frame to detect objects and enclose them with bounding boxes
        objects ← DETECT_OBJECTS(frame)
        detected_objects.append(objects)
    end for
    return detected_objects
end function
function SEMANTIC_SEGMENTATION(detected_objects)
                          ▷ Extract contextual features and apply image segmentation to identify a safe route
                                  ▷ Describe relations between objects with colored bounding boxes
    safe_route ← SCENE_UNDERSTANDING_ALGORITHM(detected_objects)
    return safe_route
end function
function THRESHOLDING(video_frames)
    bright_spots ← []          ▷ Placeholder, actual implementation depends on the thresholding algorithm
    for frame in video_frames do
                           ▷ Identify bright spots using thresholding and draw contours, label with red dots
        bright_spots_frame ← THRESHOLDING_ALGORITHM(frame)
        bright_spots.append(bright_spots_frame)
    end for
    return bright_spots
end function
function NAVIGATE_AUTONOMOUSLY(safe_route, bright_spots)
                                  ▷ Use contextual information from semantic segmentation and thresholding
                          ▷ Implement velocity control to guide the drone for precise autonomous navigation
    NAVIGATION_ALGORITHM(safe_route, bright_spots)
end function
```

These models, available as open-source, offer convenient access to their architectures and pre-trained weights, expediting development and experimentation. Several key features of these models are outlined below:

- They are trained on large-scale datasets, providing a robust foundation for transfer learning with minimal additional training data.
- They are designed to output bounding boxes alongside class predictions and they facilitate object localization, a critical aspect of object detection.
- They are selected for feature extraction based on their high mAP (mean average precision) scores on benchmark datasets like COCO, indicating superior performance in object detection tasks.
- We evaluated the performance of these feature extraction models on datasets with varying compositions. While trained on real trees, they were also tested on 3D model trees [link to dataset], exhibiting effective performance in both scenarios.
- The green region corresponds to the bounding box of the detected tree, while the red region represents the fixed field of view (FOV) of the drone.

In this project, the video data were acquired by a DJI Phantom 3 Professional quadcopter drone. The output of the proposed model was simulated for planning the drone flight missions. The drone was equipped with a single camera, including a three-axis gimbal. The camera has a resolution of 30 frames per second and is able to capture 12-megapixel photos with a field of view (FOV) of 94. The 3 axes can maintain a perfect

camera level in any flight conditions, which results in stable footage throughout every flight. In this study, an optimum resolution of 1920 × 1080 was used to record the videos. All the video frames were downsized to 1024 × 600 pixels according to the parameter settings of our model. Since our videos were recorded in high resolution, downsizing the dataset resolution did not substantially affect the image quality. In most common drone applications, the camera is mounted at a 90 degree top-view angle to capture the scene at a higher altitude. However, we consider a scenario where the drone is flying in a remote environment and is surrounded by many trees. In our experiment, the camera of the drone is fixed at the front view for object detection with a broader field of view.

A frame is first processed by detecting and localizing the objects of interest. Once an object is detected, its location relative to other objects in the scene is determined. The detection is performed by training an object classification algorithm that is expected to learn the useful features and produce highly accurate classification in different classes. In this project, a real-time object detection algorithm called Faster R-CNN ResNet-101 [31] was used. Faster R-CNN is a unified model that uses a region proposal network to generate regions of interest. By unifying three independent models, the network is able to replace the selective search method entirely. Faster R-CNN computes all the feature maps by using a single forward-pass approach over the entire image. Then, the region proposals share the feature matrices that are later used for the bounding box regressors and learning the object classifier. The workflow of Faster R-CNN for tree object detection is shown in Figure 3.

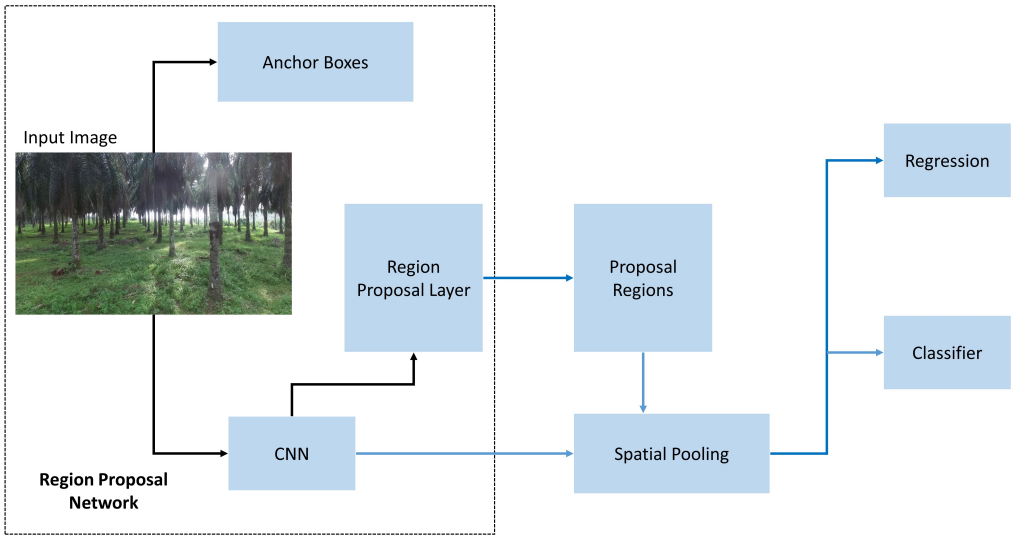


Figure 3. Overview of the Faster R-CNN pipeline.

As shown in Figure 4, the Faster RCNN detection architecture uses ResNet-101 as the ConvNet backbone to extract features. The Resnet101 Conv block consists of five fully convolution layers (*conv1_x, conv2_x, conv3_x, conv 4_x, conv5_x*) [35]. There are denoted as *C1, C2, C3, C4, and C5* for *conv1, conv2, conv3, conv4, and conv5*. Each conv layer consists of a 3 × 3 convolution size with a fixed feature map dimension. The *conv1* layer has a size of 3 × 3 filters. The *conv2, conv3, and conv4* layers have 3 × 3 filters with 128, 256, and 512 feature map dimensions, respectively. The final layer *conv5* is connected to the RPN to generate the regions proposals, and then, undergoes ROI pooling.

First, the input image is fed into the ConvNe backbone of the RPN network. In the process, the ResNet-101 CNN extracts each feature vector with a corresponding fixed feature map dimension. The network has to learn at every point of the output feature map to estimate its size at the corresponding location and also to decide whether an object is located in the image. Then, a sliding window runs spatially on the convolutional feature maps of the image at the spatial pooling stage. For each sliding window, we need to predict multiple regions of various scales and ratios simultaneously. Hence, anchor boxes are

introduced in Faster R-CNN to resolve the variations in the ratios and scales of the objects. In total, a set of nine anchors are generated that all have the same center (cx,cy), as shown in Figure 5.

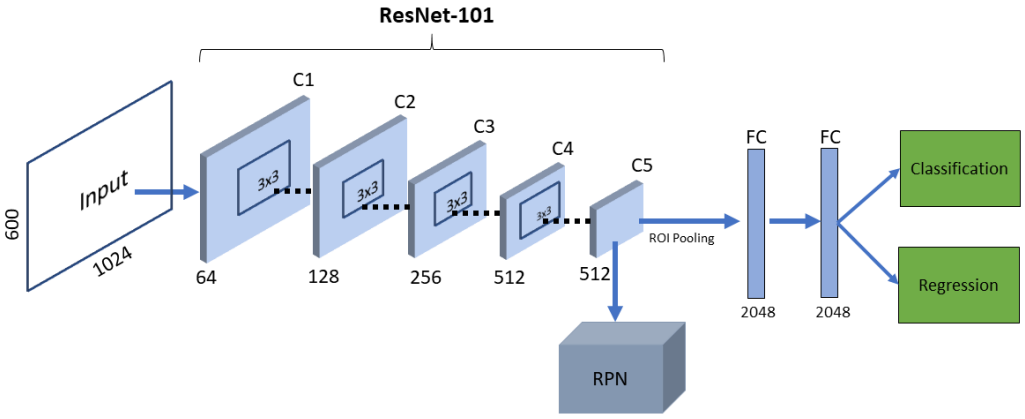


Figure 4. The Faster R-CNN ReseNet-101 architecture for object detection.

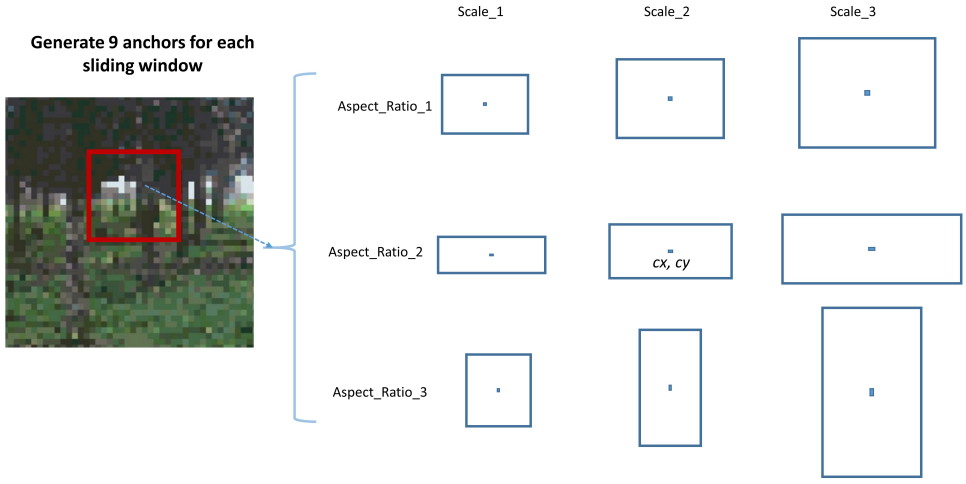


Figure 5. Generate anchor boxes.

All these anchors have different aspect ratios and different scales. At each location, these anchor boxes predict the probability of the corresponding object in the image. Each of the anchors calculates how much it overlaps with the ground-truth bounding boxes. In the final step, all the bounding boxes are processed by applying spatial pooling. Then, the feature maps are flattened and fed into two fully connected neural networks with 2048 dimensions. The spatial features extracted from the convolutional feature maps are computed in a smaller network for classification and regression. For this reason, bounding box regression is applied to improve the anchor boxes at each location. The output regressors determine a final predicted bounding box with (x, y, w, h) coordinates. The output of the classification determines the probability of whether the predicted box contains an object or not, as shown in Figure 6.

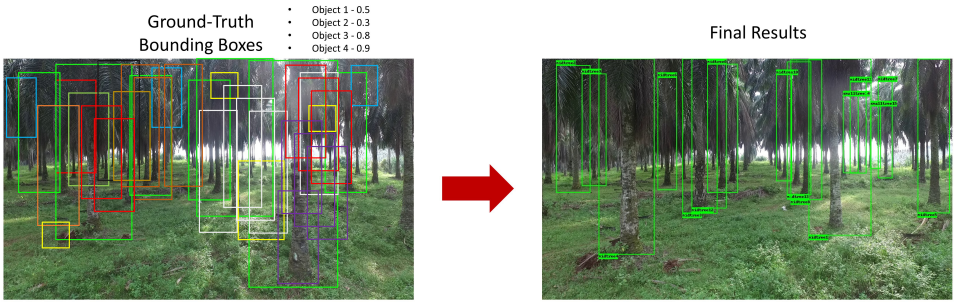


Figure 6. The classification output selects the highest probability to determine if the predicted box contains an object.

3.2. Masking

The masking operation was performed on a background layer that overlapped the actual frame to determine the navigation zone. A sample of the scene combined with the background masking layer is shown in Figure 7 . As shown in the figure, the image is 1024 pixels in width and 600 pixels in height. The center of the image is the region of interest, and we consider the center of the image to be the navigation zone. The navigation zone has an area of 341 $\times$ 250. The field of view (FOV) of the drone is always focused in the navigation zone. The other areas in black are ignored. For our scenario of forest inspection, the drone is always flying in the forward direction using the vision of the drone. Hence, the region of interest is set in the upper center area. In addition, there is a large empty space below the navigation area to prevent the drone from crashing on the ground or colliding with lower ground obstacles.

Figure 7. Sample region of interest.

The navigation zone is slightly higher than the center, and there is a free space below the navigation zone. All the images were captured at the same altitude, which was approximately 1.5 m to 2 m from the ground. Our model was designed to enable a drone to fly in a forest or oil palm plantation surrounded by mid-sized trees. For these conditions, when the drone flies among obstacles, especially in a forest, the drone will not collide with the ground. The drone is able to navigate swiftly around the obstacles in the forest without adjusting its altitude. Furthermore, constantly changing the altitude in a forest might cause collisions, as the drone would not be able to see obstacles above or below it. Since our drone setup is only equipped with a fixed camera, forest inspection is conducted in the forward direction. Based on the visual input, the drone is expected to navigate through the obstacles, perform object detection, and find its way out of the forest.

By partitioning a scene into multiple regions, the visual objects in the scene can be represented by different colors to make the scene more meaningful and easier to analyze. In this study, the visual objects in the scene such as the trees are represented by colored regions. The primary goal is to divide the scene into multiple regions for further analysis. In this scenario, the areas between the tree objects are described with red-colored regions so that the drone knows which areas can be navigated through. Simultaneously, the rela-

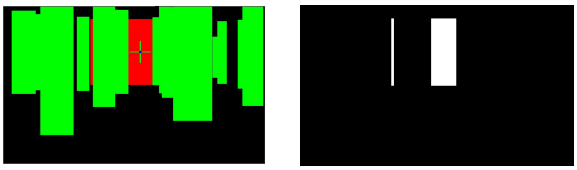
tionship between the color-based regions and the green foreground regions is described so that the drone can autonomously avoid obstacles.

To illustrate this process, we take a frame as an example, as shown in Figure 8a. The tree objects are detected with bounding boxes to enclose the objects. First, the background frame is filled with black, as shown in Figure 8b. A layer depicting a fixed FOV navigation region is added to the frame, as shown in Figure 8c. Since we are using only one camera attached to the drone and the drone is always flying forward, the FOV is set at the center of the frame as the initial view. This region is represented by red as the region of interest and navigation route in the scene. To be precise, the red region is the safe zone in front of the drone with a short distance that the drone can fly through. The tree objects detected in the scene with relative bounding boxes are shown in Figure 8d. The bounding boxes are colored in green, which indicates obstacles. The red regions overlapping the green boxes form the red spaces among the green boxes. By comparing the spatial relationships between the regions, the most significant red area is considered a potential clear path for the drone to navigate.

(**a**) Original Image (**b**) Blank Layer (**c**) Drone FOV (**d**) Image Masking

Figure 8. Image masking process

We must ascertain how to determine the most significant red safe area. Since our goal is to find the largest clear path in the scene that the drone can fly through, the corresponding image areas are masked based on the specific range of the pixel values in the HSV color space. In this case, the HSV color space for the obscured red areas shown in Figure 9a are lower = [0,100,100] and upper = [18,255,255]. The particular weighted average of the image intensities shown in Figure 9b are obtained by using masking operations. In the drone FOV, when a detected screen is filled with many obstacles, to find a clear path for the drone to navigate, the red area can be specified through a binary mask. There are two bitmaps in each scene: the actual scene and an additional mask. The unused areas from the actual scene are assigned a pixel value with all bits set to '0'. For the additional mask, the red areas are assigned a pixel value with the bits set to '1'.



(**a**) Segmentation (**b**) HSV Masking

Figure 9. Segmented regions based on the HSV color space.

For instance, the image in Figure 9b is a binary image that has two possible intensity values for each pixel, i.e., '0' for black and '1' for white. When an obstacle, i.e., a detected tree object, appears over the background, masking is performed on the overlapping scene pixels in the red area in Figure 9a. Hence, the free and open areas of the scene are preserved. The masked areas of a clear path with pixel intensities of '1' are obscured by the overlapped unused '0' areas, as in Figure 9b. Then, the system combines the original image pixels with the background pixels using the bitwise OR operation. The original image pixels are appropriately placed while keeping the masked region.

To be precise, the largest red area is calculated using the image moments. The properties of the image determined via the image moments include the area (or total intensity),

centroid, and information about the orientation. The n area of the nonzero pixels is calculated based on the image moments.

We consider the first moment as follows:

$$sum_x = \sum_{x=0}^{w} \sum_{y=0}^{h} xf(x,y) \tag{1}$$

The x coordinates of all the white pixels (where $f(x,y) = 1$) are summed up. In addition, the sum of the y coordinates of all the white pixels is calculated as follows:

$$sum_y = \sum_{x=0}^{w} \sum_{y=0}^{h} yf(x,y) \tag{2}$$

The area of the binary image is calculated as follows:

$$M_{0,0} = \sum_{x=0}^{w} \sum_{y=0}^{h} f(x,y) \tag{3}$$

When there is more than one area of white pixels in the image, the centroid is somewhere in between. Hence, we need to extract each white pixel area separately to obtain the centroid. Given that A is an array used to store all the white-pixel areas found in the image as follows:

$$A = [n_i, n_{i+1}, ...] \tag{4}$$

to find the largest (b) area, we calculate the following:

$$if \quad A = [n_{i+1}] > A = [b_i]; \quad b_i = n_{i+1} \tag{5}$$

Technically, the largest white-pixel area is selected as the final clear path because this is the largest empty area between the obstacles. When we have the sums of all the x and y pixel coordinates, to find the average in the binary image, the values obtained are divided by the number of pixels in the respective area in the image. Thus,

$$M_{1,0} = \frac{sum_x}{M_{0,0}}, \quad and \quad M_{0,1} = \frac{sum_y}{M_{0,0}} \tag{6}$$

Finally,

$$centroid = (\frac{M_{1,0}}{M_{0,0}}, \frac{M_{0,1}}{M_{0,0}}) \tag{7}$$

Therefore, the centroid obtained serves as the guided waypoint for the drone. From the drone FOV, the drone navigates to the largest red region. By combining all the layers, given that the layers do not have the same colors, the semantic color-based segmentation forms a comprehensive description of the scene. Essentially, the contextualized information is used to solve the problem of self-navigation in the scene.

3.3. Drone Navigation with Simulator

The navigation of the drone is simulated by integrating the output of the algorithm with an open-source ground station application called the APM Planner 2.0 [36]. This application is a lightweight messaging protocol for communicating with drones, especially among the onboard drone components. Thus, this application is used for controlling the drone's movements by manipulating the vehicle velocity vectors. The software in the loop (SITL) of APM Planner 2.0 is a simulator that allows the user to run any aerial copter, drone, or plane without any hardware. This simulator allows the user to test the behavior of the code without any hardware. Therefore, users are able to build customized functions for a drone that suits the scenario, conditions, and environment. As shown in Figure 10,

the simulator comes with multiple panels such as a map, a mission planner, vehicle data, and flight data. All these individual panels enable users to simulate the vehicle movements in a real environment.

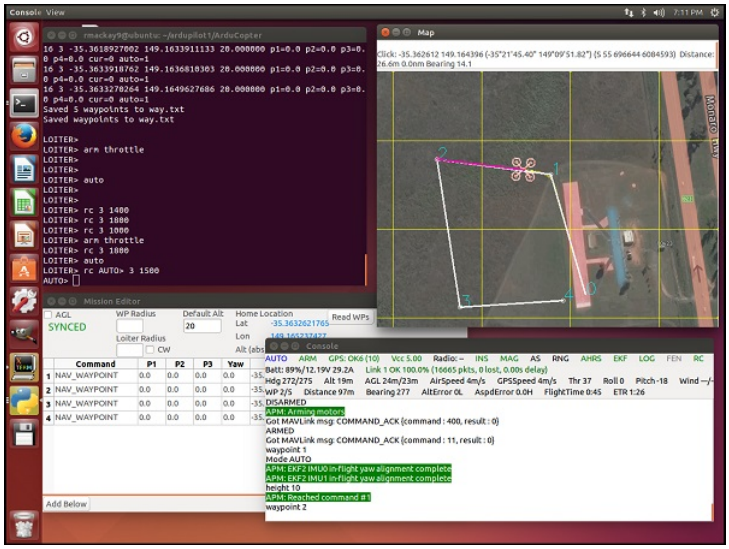


Figure 10. SITL (software in the loop) simulator.

In our scenario of vision-based drone navigation in a forest, the goal is to perform autonomous navigation without relying on any GPS devices. To minimize the cost and prove that our work is successful, all the navigation experiments are conducted with a simulator. Based on the vision sensor, all the image-based data and the coordinates of the obstacles are collected from the modules of the drone and sent to the simulator for further processing. The simulator runs in parallel with a drone-kit [33] application to enable two-way communication between the simulator and the controller application. The drone-kit allows users to create custom apps that run on an onboard companion computer and communicate with the ArduPilot flight controller using a low-latency link. Thus, there are no latency issues in simulating vision-based navigation via the simulator. The simulator displays the precise navigation path, movements, and flight data on the map in real time.

The features obtained from the framework allow the largest red region in the scene to be selected as the navigation path, which is also considered the region of interest. A waypoint is drawn on the centroid of the region of interest. The coordinates of the guided waypoint, c_x and c_y, obtained from Equation (7), are sent to the drone navigation command to control its movement without human intervention. Hence, the waypoint serves as the guided navigation point for autonomous drone flight without a predefined goal. Similarly, the centroid of the brighter region in the scene is denoted by red points. However, these red points are only useful for finding the brighter spots between trees. In the proposed method, navigation is based on the drone kit [33] velocity control command to react to new scenarios in the scene as they occur. The path always follows the guided waypoints obtained from the front view of the camera.

In this study, the camera FOV of the drone faces forward, as depicted in Figure 11. All the objects that fall in this FOV are detected for autonomous navigation control purposes. To calculate the velocity control value, the horizontal FOV is fixed at 118.2, and the vertical FOV is fixed at 69.5. Between these two FOVs is the region of interest in the visual scene, and the velocity control of the drone is determined by the coordinates in the free path.

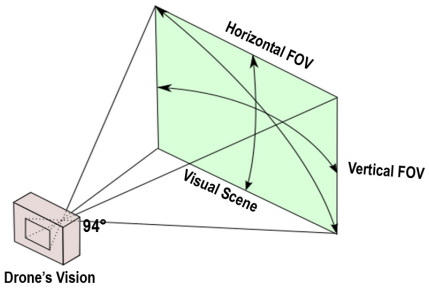


Figure 11. FOV of the drone camera.

The c_x and c_y coordinates are used to calculate the vehicle velocity control. Due to the nature of the scenario, coordinate z is not used in this study. Moreover, the speed and altitude of the drone are fixed. Referring to Equations (8) and (9), the coordinates c_x and c_y from the blue point are applied in the equations to obtain the velocity control output.

$$(c_x - \frac{horizontal\_resolution}{2}) * \frac{horizontal\_fov}{horizontal\_resolution} \tag{8}$$

$$(c_y - \frac{vertical\_resolution}{2}) * \frac{vertical\_fov}{vertical\_resolution} \tag{9}$$

If the output coordinates, c_x and c_y, fulfill the velocity threshold values, as shown in Table 1, the direction of the drone changes accordingly. We assume that there is no additional information regarding the height of the drone including a GPS signal. We obtain the output coordinates and perform waypoint guiding in the region of interest (ROI). The size of the ROI will be slightly adjusted from the center position of the frame to prevent the drone from flying toward the ground or a lower area, which is based on the velocity control command. The reason we chose these FOV and ROI resolution settings is to ensure that the obstacles and navigation marker appear inside the selected ROI. Any coordinates that fall in other areas in the visual scene are ignored.

Table 1. Velocity control.

Velocity Operation	Direction
$velocity\_x > 0$	Fly NORTH
$velocity\_x < 0$	Fly SOUTH
$velocity\_y > 0$	Fly EAST
$velocity\_y < 0$	Fly WEST
$velocity\_z > 0$	Ascend
$velocity\_z < 0$	Descend

3.4. Model Training

Object detection using data-driven approaches can be seen as an extension of an image classification problem. The detection part of the problem is solved by training the ResNet101 model, which is expected to learn useful features and be able to demonstrate high classification accuracy in object detection. A Tesla K80 GPU card was set up in the server to train the model. In the training process, 20,000 images were used to train the ResNet101 model to detect trees as the objects of interest.

The whole training process took about 5 days on the trees dataset. The training was executed in several sessions. Every 5k steps, the ResNet-101 model was validated and we measured its mAP scores concurrently in the training process. Additionally, the training

routine periodically saved checkpoints every five minutes. The checkpoint at the highest number of steps was used to generate the frozen inference graph for fine-tuning and for next training. An initial learning rate of 0.001 was used for the first 20k steps, and 0.0001 for the next 20k steps. Finally, the training was stopped at 40k steps when the loss value did not have a decreasing tendency.

4. Results and Discussion

Several experiments were conducted to evaluate the performance of the proposed model. All the experiments were conducted based on videos recorded by a drone under various conditions, including different lighting levels and random obstacles along the path of the drone. In this study, there was no predefined target or path for the drone flight in the scene. These videos are the ground truth of the scenario where the drone was seamlessly flying directly toward a clear path. We assumed that the drone was flying autonomously in an unknown environment to find a clear path between the obstacles. The video data collected are based on the front-view aerial images taken by the drone. Thus, our model took the video sequences as the input to develop scene understanding and reasoning. At first, several tests were conducted to select a good model of Faster R-CNN for object detection. Once the model was confirmed, simulation tests were conducted based on the waypoint generated by the model from the video data to verify our approach in a scenario of a drone flying through trees in a remote environment. This experiment simulated autonomous drone navigation in a jungle without relying on a GPS.

4.1. Object Detection Results

Faster R-CNN achieved impressive accuracy in object labeling and detection tasks [31]. For this purpose, three feature extraction models, namely, the *Inception-Resnet-v2 Atrous*, *NAS*, and *Resnet101* models, which were all pre-trained from this detector, were evaluated.

These models were trained with our data composed of 20,000 images captured from the drone and through online collection. The object class in this study was trees. All the images were annotated, and 10% of the dataset was randomly selected for testing. Based on the mean average precision (mAP) for each model in our dataset, the Faster-RCNN Inception-Resnet-v2 Atrous model achieved a mAP value of 92.0%, Faster-RCNN NAS achieved a mAP value of 85.0%, and *Faster-RCNN Resnet101* achieved a mAP value of 95.0% (the highest value). Hence, *Faster-RCNN Resnet101* was selected as the ideal model to be implemented in our proposed framework.

4.2. Flight Direction Results

To evaluate the flight directions of the drone, the coordiantes x and y that appeared in the video were captured to calculate the navigation waypoint. The calculation was performed by using the formula in Equations (8) and (9) to calculate the waypoint. Two separate experiments were conducted to test whether the drone was able to fly and navigate in different directions, like north, south, east, and west, according to the obtained coordinates from the masking module. As shown in Table 2, 10 tests were conducted in the experiment by applying different positive and negative coordinates to the simulator. When the output results are more than 0, the drone will fly toward the north direction. If the output results are less than 0 , the drone will fly toward the south direction. Similarly to the results in Table 3, the drone will fly in the west direction if the output results are less than 0, and it will fly in the east direction if the results are more than 0.

Table 2. Velocity control for coordinate x.

Coordinate x	Flight Direction
5.23	NORTH
4.11	NORTH
5.68	NORTH
3.19	NORTH
0.22	NORTH
−1.15	SOUTH
−0.19	SOUTH
−1.18	SOUTH
−4.23	SOUTH
−5.87	SOUTH

Table 3. Velocity control for coordinate y.

Coordinate y	Flight Direction
−2.23	WEST
−0.11	WEST
−2.68	WEST
−4.19	WEST
−0.92	WEST
2.15	EAST
0.33	EAST
1.18	EAST
3.12	EAST
5.56	EAST

4.3. Autonomous Navigation Results

To evaluate the performance of our framework in terms of waypoint generation as a path for the drone to navigate, we calculated the blue waypoint obtained through our masking model that was drawn on the red region in every single frame. To be precise, if the blue point is drawn on the red region, this condition is a positive result. However, if the blue point is drawn in the object bounding box or green region, this condition is a negative result. Figures 12 and 13 show correct and incorrect results, respectively, in frames from the video sequences.

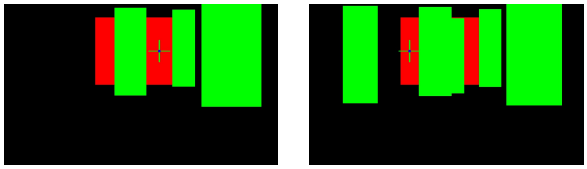


Figure 12. Frames with correct waypoints.

Figure 13. Frames with incorrect waypoints.

In the frames shown in Figure 12, the guided waypoint is located correctly on the free path. However,the frames in Figure 13 show that the guided waypoint is located on an obstacle, which is considered a false result. The movements of the drone are always based on the guided waypoint navigation command. When the guided waypoint falls on the free path, the navigation command directs the drone to fly toward that area.

Additionally, we collected several videos to test the proposed method. For each video, a total of 500 frames from the video sequence were processed to find the blue waypoint. The total number of positive and negative points was calculated for each video. Then, the accuracy rate of the algorithm was obtained by calculating the ratio of the number of frames, where the frame of the blue waypoint is correctly overlapped in the red region (n), to the total number of frames (N). The formula is as follows:

$$Accuracy = \frac{n}{N} * 100 \tag{10}$$

There are six videos with 500 frames each, and the accuracy results are shown in Table 3. Overall, the experiments show higher accuracy in localizing the navigation point, with an average of 97.7%.

To further demonstrate the performance of our proposed method, we conducted several simultaneous simulations on the APM drone simulator with the output from our system. In this experiment, we evaluated our proposed method by simulating autonomous drone navigation in a forest. To prove that our method can successfully develop scene understanding and provide precise navigation command to the drone, the trails of the drone were captured and plotted on an offline satellite map. In the navigation module, the ultimate goal of the blue waypoint is to determine a clear path for navigation. Essentially, the coordinates acquired from the proposed system are processed using the velocity-guiding function in the drone kit framework [33]. Hence, the drone is automatically guided by the navigation command and reacts to the surrounding obstacles in the scene.

The simulations are graphically shown in Figures 14 and 15. All the simulations were set at the same starting point which is from Location A to Location B. Additionally, all of the simulations were evaluated on different videos consisting of 500 frames each, as mentioned in Table 4. The flying direction was different in each simulation.

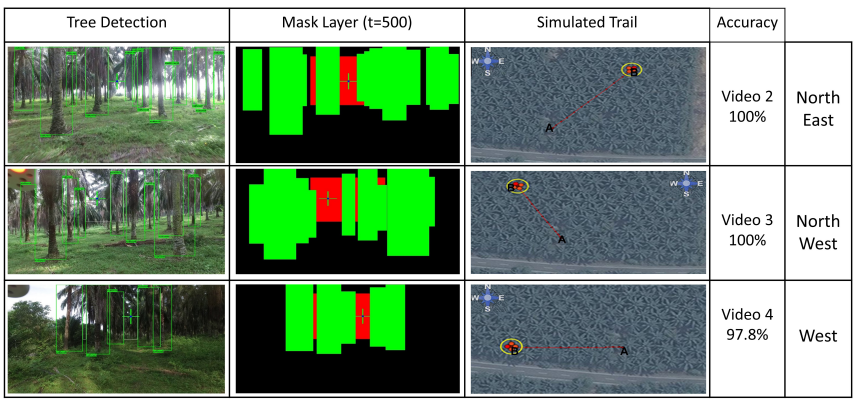


Figure 14. Correct simulated trails of autonomous drone navigation.

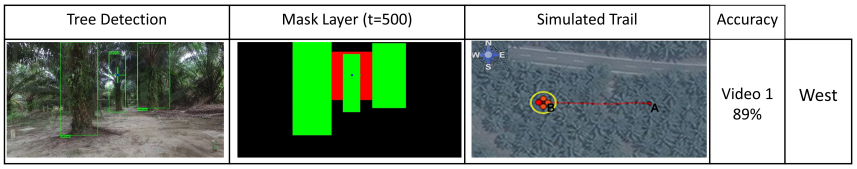


Figure 15. Incorrect simulated trails of autonomous drone navigation.

Table 4. Accuracy rate in localizing the navigation waypoint of each video.

Video	No. of Frames	No. of Errors	Accuracy
1	500	55	89.0%
2	500	0	100.0%
3	500	0	100.0%
4	500	11	97.8%
5	500	3	99.4%
6	500	0	100.0%
	Average		97.7%

As shown in the first row of Figure 14, the drone can interpret the scene and navigate autonomously toward the north-east direction (Location A to B) in a straight path correctly. The input was taken from the drone camera based on the detected trees, and the masking module processed the image frames for waypoint guiding. Based on the simulated results, the masking module was able to process all 500 frames with 100% localized waypoint accuracy in Video 2. The second and third rows of Figure 14 depict the correct simulated trail of autonomous drone navigation in different directions. As shown in the figure taken from Video 3, the drone was able to navigate autonomously toward the north-west direction along a straight path correctly with 10% accuracy. The guided waypoint consistently appeared on the left side of the region of interest, providing a north-west navigation command to the drone in 500 consecutive frames. Meanwhile, the simulation illustrated in the third row of the figure was based on the results from Video 4, with 97.8% accuracy. The drone was flying autonomously toward the west direction.

A false simulated trail is shown in Figure 15. The trail was based on Video 1, which achieved 89% accuracy with minor false localized waypoints. As shown in the figure, at frame T = 353, the guided waypoint landed in the middle of the tree or green region. The simulated trail shows that the drone was flying in a straight line toward the west for the rest of the frames. Hence, the masking module was providing false waypoint guiding information to the drone navigation command, which could result in the drone crashing into the trees. This problem was due to the model being unable to segment the distant objects correctly with only a few known frames. However, when the tree appears closer and bigger in the drone's FOV in the next frames, if the sizes of the trees that appear in the scene are segmented correctly, the masking module will be able to estimate the distance between the trees correctly. Due to the time and budget constraints of this project, this implementation would be considered in future work.

Finally, these simulations revealed the effectiveness of the proposed model because the drone could develop real-time scene understanding for autonomous navigation purposes.

4.4. Discussion

Our results indicate that the proposed method performs well in guiding a drone through scene understanding without using extensive computational resources or complex algorithms. The object detector was pre-trained offline on a typical object class that appears in a forest inspection scenario. In this approach, all the features obtained during object detection are sent to the proposed masking modules for further processing. A detailed

description of the scene was achieved by segmenting the region of interest. The results are particularly interesting in terms of the recent developments in autonomous agent implementations that are suitable for low-power and low-cost UAVs. Thus, the implementation of autonomous drone navigation in real-life applications is feasible.

5. Conclusions

Today, various industries use drones in their business operations. We proposed an innovative deep learning masking model for realizing vision-based autonomous navigation for drones. In this paper, we considered the problem of drone visual navigation using a region proposal network-based detector for generating features. In our proposed masking process, the features are used with waypoint localization for autonomous navigation. Such a localization technique is beneficial for the correct functioning of vital controls and navigation functions in autonomous vehicles that operate in complex environments without or with an unreliable infrastructure-based positioning system.

Specifically, we evaluated the performance of our proposed method. The results of the experiments based on purely the 2D-view visual camera inputs showed the effectiveness of the object detection algorithm combined with an HSV (Hue, Saturation, and Value) color-based segmentation approach for the drone. The final test was conducted using a drone simulation, and the simulated trials proved that the drone can autonomously navigate in a dynamic environment. For future developments, we recommend adding more cameras to the drone. If the drone is equipped with multiple cameras, it can capture more data and perform object detection with a 360° view to enhance navigation in a dynamic environment. Furthermore, implementing stereo cameras in the framework could provide comprehensive visual detection, in which the approximate distance to the trajectory could be measured to provide a more accurate navigation path for the drone. Using a multimedia system, we will focus on developing a panoramic view for navigation.

All these issues will be considered in our future research.

Author Contributions: Conceptualization, A.L. and S.-P.Y.; methodology, A.L. and S.-P.Y.; software, A.L.; validation, A.L., S.-P.Y. and J.W.; formal analysis, A.L., S.-P.Y. and J.W.; investigation, A.L.; resources, W.P. and J.W.; data curation, A.L.; writing—original draft preparation, A.L., S.-P.Y. and J.W.; writing—review and editing, A.L., S.-P.Y. and J.W.; visualization, A.L., S.-P.Y. and J.W.; supervision, A.L., S.-P.Y. and J.W.; project administration, W.P. and J.W.; funding acquisition, W.P. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Due to privacy or ethical restrictions, as well as university policies, the data supporting the reported results are unavailable.

Conflicts of Interest: Author Alvin Lee was employed by the company Mercedes-Benz Malaysia. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. The National Coordination Office (NCO). National Coordination Office for Space-Based Positioning, Navigation, and Timing. 2013. Available online: https://www.gps.gov/governance/excom/nco/ (accessed on 18 May 2021).
2. Karpenko, S.; Konovalenko, I.; Miller, A.; Miller, B.; Nikolaev, D. UAV control on the basis of 3D landmark bearing-only observations. *Sensors* **2015**, *15*, 29802–29820. [CrossRef] [PubMed]
3. Kato, A.; Obanawa, H.; Hayakawa, Y.; Watanabe, M.; Yamaguchi, Y.; Enoki, T. Fusion between UAV-SFM and terrestrial laser scanner for field validation of satellite remote sensing. In Proceedings of the 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 26–31 July 2015 ; IEEE: Piscataway, NJ, USA, 2015; pp. 2642–2645.
4. H.X. Pham and H.M. La and D. Feil-Seifer and L.V. Nguyen. Autonomous UAV Navigation Using Reinforcement Learning. *arXiv* **2018**, arXiv:1801.05086.
5. Xiao, B.; Yin, S. A new disturbance attenuation control scheme for quadrotor unmanned aerial vehicles. *IEEE Trans. Ind. Informatics* **2017**, *13*, 2922–2932. [CrossRef]
6. Roberge, V.; Tarbouchi, M.; Labonté, G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Trans. Ind. Inform.* **2012**, *9*, 132–141. [CrossRef]

7. Cavaliere, D.; Loia, V.; Saggese, A.; Senatore, S.; Vento, M. Semantically Enhanced UAVs to Increase the Aerial Scene Understanding. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 555–567. [CrossRef]
8. Yong, S.; Yeong, Y. Human Object Detection in Forest with Deep Learning based on Drone's Vision. In Proceedings of the 2018 4th International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 13–14 August 2018; pp. 1–5. [CrossRef]
9. Yong, S.; Deng, J.D.; Purvis, M.K. Modelling semantic context for novelty detection in wildlife scenes. In Proceedings of the 2010 IEEE International Conference on Multimedia and Expo, Singapore, 19–23 July 2010; pp. 1254–1259. [CrossRef]
10. Cai, J.; Huang, P.; Zhang, B.; Wang, D. A TSR visual servoing system based on a novel dynamic template matching method. *Sensors* **2015**, *15*, 32152–32167. [CrossRef] [PubMed]
11. Chen, T.; Lu, S. Object-level motion detection from moving cameras. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *27*, 2333–2343. [CrossRef]
12. Saripalli, S.; Montgomery, J.F.; Sukhatme, G.S. Visually guided landing of an unmanned aerial vehicle. *IEEE Trans. Robot. Autom.* **2003**, *19*, 371–380. [CrossRef]
13. Bruni, V.; Vitulano, D. An improvement of kernel-based object tracking based on human perception. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 1474–1485. [CrossRef]
14. Dang, C.T.; Pham, H.T.; Pham, T.B.; Truong, N.V. Vision based ground object tracking using AR. Drone quadrotor. In Proceedings of the 2013 International Conference on Control, Automation and Information Sciences (ICCAIS), Nha Trang, Vietnam, 25–28 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 146–151.
15. Thiang, I.N.; LuMaw, H.M.T. Vision-based object tracking algorithm with ar. drone. *Red* **2016**, *160*, 179.
16. Szegedy, C.; Toshev, A.; Erhan, D. Deep neural networks for object detection. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 2553–2561.
17. Erhan, D.; Szegedy, C.; Toshev, A.; Anguelov, D. Scalable object detection using deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 2147–2154.
18. Yong, S.P.; Deng, J.D.; Purvis, M.K. Novelty detection in wildlife scenes through semantic context modelling. *Pattern Recognit.* **2012**, *45*, 3439–3450. [CrossRef]
19. Filonenko, A.; Jo, K.H. Unattended object identification for intelligent surveillance systems using sequence of dual background difference. *IEEE Trans. Ind. Inform.* **2016**, *12*, 2247–2255.
20. Zhou, Z.; Zhang, C.; Xu, C.; Xiong, F.; Zhang, Y.; Umer, T. Energy-efficient industrial internet of uavs for power line inspection in smart grid. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2705–2714. [CrossRef]
21. Huang, H.; Savkin, A.V. An algorithm of reactive collision free 3D deployment of networked unmanned aerial vehicles for surveillance and monitoring. *IEEE Trans. Ind. Inform.* **2019**, *16*, 132–140. [CrossRef]
22. Yang, L.; Sun, Q.; Ye, Z.S. Designing Mission Abort Strategies Based on Early-Warning Information: Application to UAV. *IEEE Trans. Ind. Inform.* **2019**, *16*, 277–287. [CrossRef]
23. Huang, Y.P.; Sithole, L.; Lee, T.T. Structure From Motion Technique for Scene Detection Using Autonomous Drone Navigation. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *49*, 2559–2570. [CrossRef]
24. Minaeian, S.; Liu, J.; Son, Y.J. Vision-based target detection and localization via a team of cooperative UAV and UGVs. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 1005–1016. [CrossRef]
25. Dementhon, D.F.; Davis, L.S. Model-based object pose in 25 lines of code. *Int. J. Comput. Vis.* **1995**, *15*, 123–141. [CrossRef]
26. Zhang, H.; Geiger, A.; Urtasun, R. Understanding high-level semantics by modeling traffic patterns. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 3056–3063.
27. Guo, Y.; Liu, Y.; Georgiou, T.; Lew, M.S. A review of semantic segmentation using deep neural networks. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 87–93. [CrossRef]
28. Arbeláez, P.; Pont-Tuset, J.; Barron, J.T.; Marques, F.; Malik, J. Multiscale combinatorial grouping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 328–335.
29. Zitnick, C.L.; Dollár, P. Edge boxes: Locating object proposals from edges. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Switzerland, 2014, pp. 391–405.
30. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
31. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
32. Uijlings, J.R.; Van De Sande, K.E.; Gevers, T.; Smeulders, A.W. Selective search for object recognition. *Int. J. Comput. Vis.* **2013**, *104*, 154–171. [CrossRef]
33. 3D Robotics. Guiding and Controlling Copter. Available online: https://dronekit-python.readthedocs.io/en/latest/guide/copter/guided_mode.html (accessed on 26 August 2020).
34. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.

35. 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, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
36. ArduPilot Dev Team. APM Planner 2 Home. Available online: https://ardupilot.org/planner2/ (accessed on 18 Apr 2021).