OULUN AMMATTIKORKEAKOULU

Tuure Vairio

SEMI-AUTONOMOUS DRONES IN WILDFIRE DETECTION

A Proof of Concept

SEMI-AUTONOMOUS DRONES IN WILDFIRE DETECTION

A Proof of Concept

Tuure Vairio
Bachelor's Thesis
Spring 2023
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Bachelor's Degree in Information Technology

Author(s): Tuure Vairio
Title of thesis: Semi-Autonomous Drones in Wildfire Detection, A Proof of Concept
Supervisors(s): Janne Kumpuoja
Term and year when the thesis was submitted: Spring 2023
Number of pages: 42

The scope of this thesis was to create a proof of concept solution of a semi-autonomous drone swarm that can detect and monitor wildfires. This thesis was ordered by the VTT Technical Research Centre of Finland as a part of their FireMan project. This project tries to create innovative solutions to combat the ever-increasing wildfire thread caused by global warming of the climate.

The main theories used in this thesis were concepts of artificial neural networks and how image detection works, how to train an image detection algorithm, and the basics of aerial searches and patterns that the drone could utilize to detect fire from a selected area.

Used data for the flame detection model was found on the internet, mainly using Google image search and the training algorithm used was YOLOv8. Drones were simulated by using Parrot Sphinx simulator software and the command codes were written in Python with Parrot Olympe SDK. Additional libraries were also used for detection streaming and calculations. Olympe SDK allows the use of physical drones if field tests are to be conducted.

Key findings indicate that a simple image detection model could be used for detecting flames from a video stream sent by a drone. A drone can also be flown by using a computer that sends command signals to the drone. The created proof of concept could be developed into a concrete product with a drone swarm capability. However, there are a variety of subjects to be tackled for this to be a working product, such as swarm command and emergency capabilities.

Keywords: flame detection, autonomous flight, surveillance, wildfire, drone, UAV, UAS

CONTENTS

1 INTRODUCTION

As climate change progresses, it has already created more extreme weather phenomena on Earth. Already we can see more devastating floods and draughts and they bring more challenges to manage them. As draughts become more frequent, also wildfires become more common (1, 14). The fires not only create significant economic losses but also put lives in danger and speed up the climate crisis by releasing more and more greenhouse gases into the atmosphere (6). New solutions for prevention, de-escalation, and management of these fires are being researched and implemented and, for example, airplanes and helicopters have been used for quite a while to search and monitor forest fires (12).

By using semiautonomous UASs (Unmanned Aerial Systems or drones) for bushfire search and monitoring operations, it might be possible to lower carbon emissions and get the operations easier and faster to launch and handle. A video feed can be shared with the firefighting team on the ground to alert and inform and to the emergency operations center, for planning and organizing the firefighting mission. Early detection and monitoring can provide vital information for the diagnosis of the fire and help predict the propagation of the situation (12).

More view angles can be created with drone swarms and different kinds of sensor data can be utilized (RGB images, thermal, photogrammetry, LiDAR, etc.), and more areas can be covered. Drones can also be used to find secure routes to and from the fire area, to find natural water sources (12), and can be flown closer to the ground than helicopters because the prop wash is not in drones is not as powerful. They can also fly in smoke without risking pilots.

Drones have been part of implemented technologies for a while and lots of research even in forest fire utilization has been made(12) and this thesis relies mostly on the working paper *Using a Semi-Autonomous Drone Swarm to Support Wildfire Management – A Concept of Operations Development Study* (H. Karvonen, E. Honkavaara, J. Röning, V. Kramar & J. Sassi) (12) and an article *Monitoring and Cordoning Wildfires with an Autonomous Swarm of Unmanned Aerial Vehicles* (F. Saffre, H. Hildmann, H. Karvonen, T. Lind) (18).

The technology of autonomous systems has become more advanced and a part of our day-to-day lives, however, the definition of autonomy might be unclear for many of us. Unmanned aerial

systems need an operator or a pilot, even if the pilot is not in the aircraft itself. Automatic systems can perform processes from start to finish, with predefined inputs. Autonomy is often mixed with Automatic systems, but the main difference is self-governance and the capability to make decisions spontaneously. (20).

The level of autonomy can be defined from, *human-operated* to *fully autonomous* (20). For example, in commercial drones some autonomy is already implemented, such as return to home (RTH) when the signal is lost, or hover, when the drone waits for human input. When a system is fully autonomous it can perform all the given tasks, such as creating a flightpath, avoiding obstacles, and regenerating the flightpath if needed.

The scope of the thesis is only to create a proof-of-concept semi-autonomous drone solution that can detect and monitor a fire in a simulated situation in software simulation and field tests in the future, by implementing computer vision for fire detection and considering different ways to conduct monitoring missions. For this purpose, Parrot Anafi drones were chosen, because they can be operated with an open-source library and Oulu University of Applied Sciences has those drones for use.

This thesis is organized in such a way that first some methodologies and materials regarding this topic are covered briefly. After that the workflow is explained generally and the results, findings, and discussion are introduced at the end.

2 BASICS OF ARTIFICIAL NEURAL NETWORK

For fire detection to work with the use of an image detection algorithm, a simple understanding of image detection methods and artificial neural networks should be formed. An Artificial Neural Network (ANN) is an algorithm inspired by the way a brain functions. Deep learning consists of a variety of methods, but the purpose of this thesis is only for the main concept understanding.

To simplify, the ANN algorithm takes an input and passes through a function that triggers specific "neurons" to activate and generate an output. To train ANN a dataset or a task is needed and by adjusting a set of weights and repetition the wanted prediction can be achieved. Traditional machine learning or ML differs from ANN in such a way that detection is achieved by a manually programmed script that describes the wanted features. This would mean more and more complexity for the script if the recognition task is difficult (3,19).

ANN is trained with feature-labeled input data that then goes through the script and the script trains itself. The architecture of the network can be customized so that it can process structured or unstructured data, such as images. Independent variables are fed to input layers (Figure 1) and the input layer outputs are connected to hidden layers that modify the data until it is fed to the output layer. Each layer has neurons that are connected to each neuron in the next layer (3, 15).

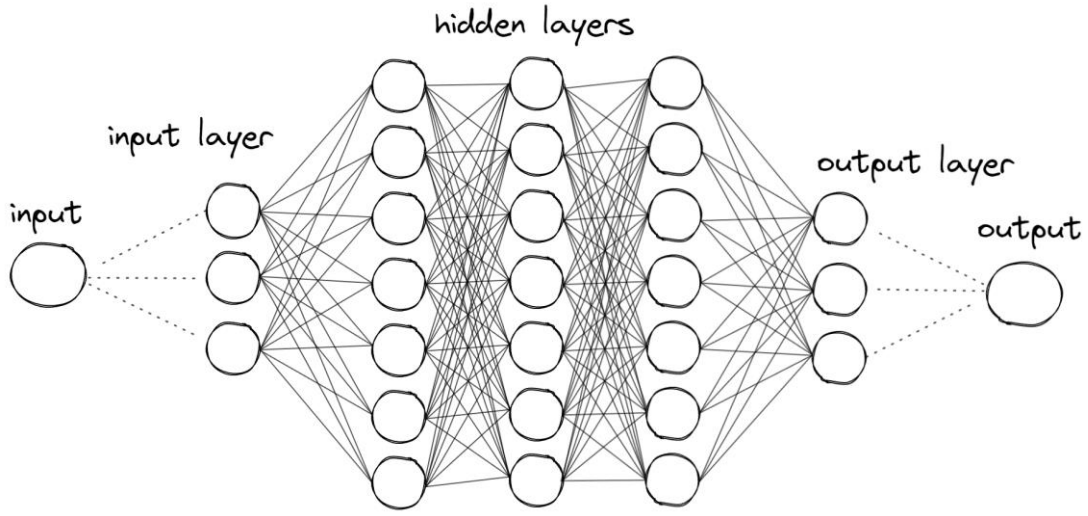

Figure 1. Neuron connections in Artificial Neural Network.

3 COMPUTER VISION AND ARTIFICIAL INTELLIGENCE

Computer vision enables devices such as computers, smartphones, cars, and drones to detect and identify different kinds of objects from digital image feeds. It is used for example in face recognition, register plate reading, image searching, and face filters in social media. After advances in machine deep learning, the scope of applications has also increased. (8).

Artificial intelligence or AI has generated new possibilities for different industries, including medicine and autonomous driving. With good input data and a sophisticated algorithm, AI can learn to complete tasks tirelessly. Already in 2015, AI has beaten human-level performance in image classification. (3).

To simplify image detection, the task can be divided into classification, object detection, and categorizing. By segmenting the image into regions by its contents, basic classification can be made (Figure 2, a). More distinctive identification of the image and its contents can be made by identifying the boundaries of different objects in Figure 2, b. Object detection has been done when different objects are detected from the image. Objects can be highlighted by creating a bounding box around them. The objects inside the bounding box can then be identified and categorized. (8).

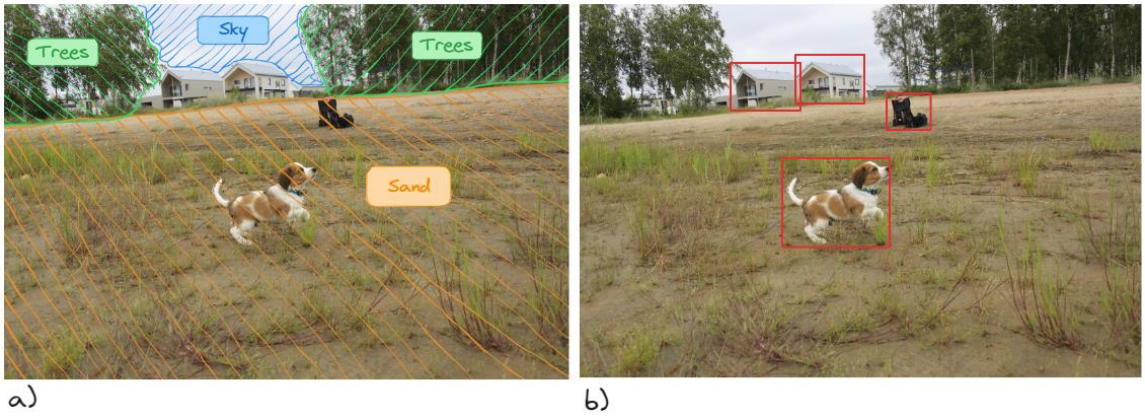

Figure 2. a) Image Classification and b) Object Detection.

3.1 Computer Vision Applications

Different applications have been created to tackle various computer vision tasks. Here are some that can be applied to fire detection and monitoring tasks.

- **Image classification** is perhaps the most used method of computer vision. Can be used for example when identifying a burning area from an untouched or burned area.
- **Object detection** can be utilized to determine different heat sources and identify uncontrolled fires from controlled such as campfires or leaf burning.
- **Object tracking** can be used when following a firefront or aiding firefighters on the ground.
- **Image geometry** can be utilized to create a 3D model of the mission area. It can provide information about the biomass in the area and some vital information about natural barriers or possible solutions to control fire.

3.2 Training AI for Fire Detection

To train AI one must understand some key concepts of AI training. The image detection model used in this thesis uses convolutional neural networks or CNNs (sometimes ConvNet) and are probably the most well-known and are proven to be very capable of detecting objects and implementing computer vision to robotics (16, 19). There are three main types of neural layers that all have different tasks in CNN architecture. The layers are **convolutional layers**, **pooling layers,** and fully **connected layers**.

In CNNs all the neurons are not connected with each neuron in the next layer, the neurons are divided into local receptive fields (Figure 3*)*. A local receptive field connects to the next layer and the data is scanned to a feature map. (15).

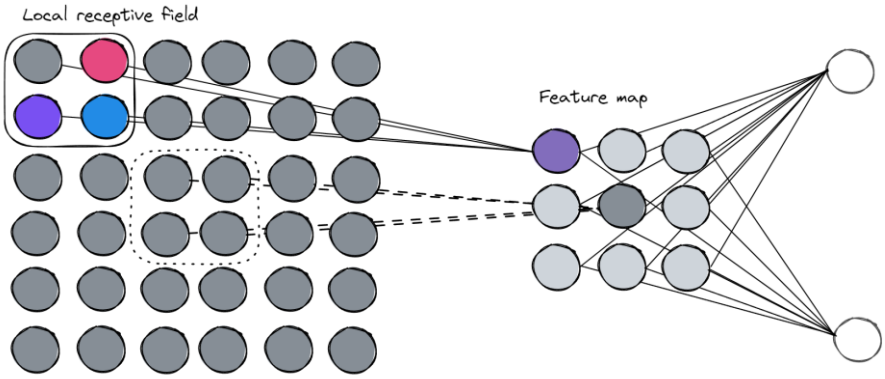

Figure 3. Convolutional Neural Network.

In the convolutional layer, the whole image is convolved with the help of filters or intermediate feature maps into new feature maps. These filters can extract for example eyes, corners hair,

people, animals, etc. The output will be sent to the next layer. The convolution process takes a defined matrix of pixels and goes through the image and will output a dot product of each pixel. (9,19).

Before the data is sent to the pooling layer the data is activated with a function such as ReLu (Rectified Linear Unit). The Function removes all the negative values from the image and replaces them with zero values. The pooling layer's main purpose is to reduce the image size or to downsample the input for the next convolutional layer (Figure 4*)*. Smaller image size implies less lost data (leaving only the most important features), but it also reduces overfitting and speeds up the process. This makes the process more efficient. Pooling can also be made in different ways, for example in max pooling, only the maximum value of each sector is selected. The minimum value is selected in min value pooling. (13, 15, 19).

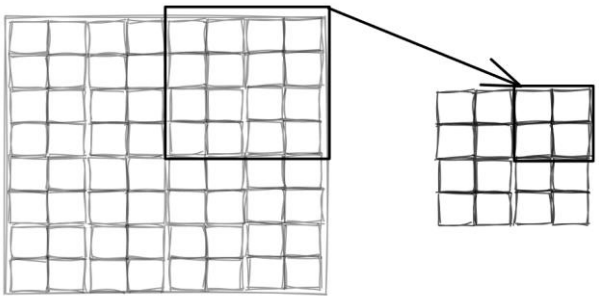

Figure 4. Pooling layer reduces the data in the image.

A fully connected layer connects all the layers in one layer to every neuron in another layer. Their main goal is to convert a 2D matrix to a 1D feature vector. The vector can be fed onward to further processing or to different categories for classification (19).

To teach an AI to detect images, the training needs input. The input images or a dataset include annotated and labeled images of the features that the AI must learn. The labeling is made by humans and the dataset (Figure 5) might consist of thousands of images. The better the annotation is done; the better CNN can learn the features. If the dataset contains a lot of duplicates, the AI is not able to do generalization as well.

A Batch size (Figure 5) is the number of images the CNN will process in one **iteration** (Figure 5). A dataset of 2 000 images with a batch size of 40 takes 2 000/40 = 50 iterations to go through all the images on the dataset. After one iteration, the **weights** are updated. The weights determine how much weight the neuron input will have on the neuron output. **Bias** is a constant value that

makes sure the neuron will activate even if the input is zero. When a complete dataset has passed through the entire neural network one **epoch** has been completed (16).

The batch size affects the needed computational power, but large batch sizes are not as good at generalizing as smaller batches. The small batches generate a lot of variation since it is not a good representation of the used dataset. By having a lot of variation, the algorithm can learn the differences between the samples better when using it in object detection. (10, 11).

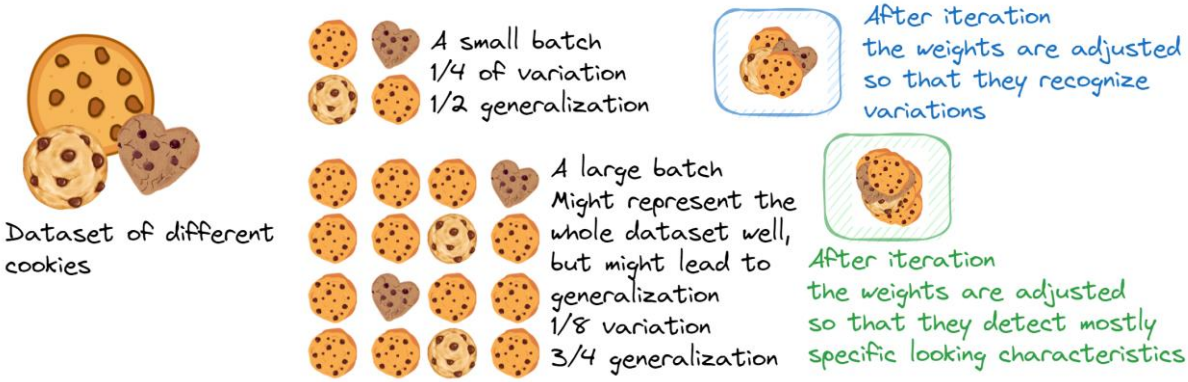

Figure 5. Dataset, batch, and iteration explained with cookies.

4 DETECTING OBJECTS FROM AIR

UAVs will continue to replace manned systems in, for example, Search and Rescue (SAR) missions, because they are easier to operate and are less costly and can be automated. UAVs require also less manpower and can pass over obstacles more efficiently than humans on the ground. (4).

To find something from a defined area in irregular terrain, the most reliable way is to have some kind of pattern. The pattern approach eliminates the chance of multiple searches of the same location. In some cases, randomness might give faster results, but it might also skip important locations altogether. Due to the limited flight time, battery life is also one thing to consider. By having a specified pattern, it is easy to resume the search from the spot the drone has gone for a battery change.

Different search patterns have been used in searches for a long time and there are different methods for creating them (2). Some sophisticated methods are also used together with drones to generate even faster and more efficient ways to deploy drones in search missions (4, 5).

A few flight patterns are introduced in this thesis, and some are implemented in the drone control code.

4.1 Some Search Patterns

Parallel Track Pattern (Figure 6, a) is widely used due to it being simple and can cover large search regions. The pattern consists of longer search legs that cover the search area and smaller search legs that lead to longer legs. The search legs are oriented to the search area, so there are longer legs and fewer turns (oriented along the major axis). (2).

The creeping Line Pattern (Figure 6, b) is a modification of the parallel line pattern, but there are more turns, and the legs are shorter (oriented along the minor axis). This pattern is used for example when a specified sector is searched from the area before others. (2).

Square Pattern (Figure 6, c) is mostly used when the search target location is somewhat known. The pattern starts from the center of the search area and starts spiraling outwards either clockwise or counterclockwise from the starting point. (2).

The sector Search Pattern (Figure 6, d) is used when the search target location is known precisely. This is used when the search target might be drifting, and the search area is not too large (21). The pattern is started from the last known location and the pattern is repeated at a specified angle.

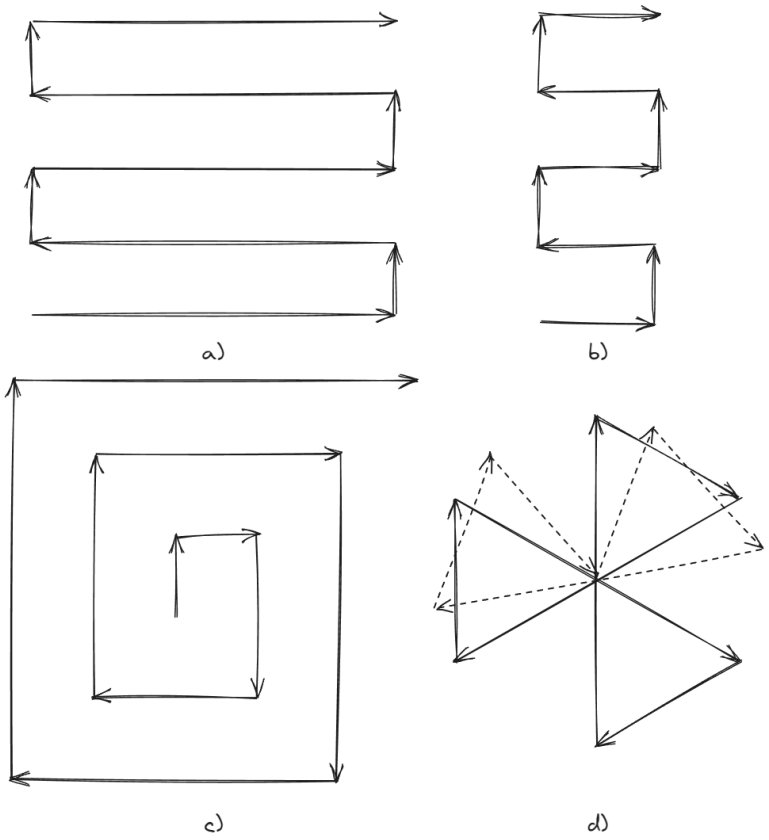

Figure 6. Different Search patterns

4.2 Aerial Forest Fire Surveillance in Finland

The Regional State Administrative Agency for Northern Finland organizes aerial forest surveillance in the whole of Finland. The surveillance flights are mostly conducted by aviation clubs or private sector companies. The flights are conducted during the wildfire season and are divided into specified areas and flight paths to cover the whole country. (22).

4.3 Determining Track Spacing and Altitude by Focal Length

The drone is required to fly at an altitude that allows the drone camera to capture the whole search area. Therefore, the focal length of the camera must be known. Focal length is the distance between the lens and the imaging point (the point where the image sensor should be). The length determines the angle of view (23). When the angle of view is fixed, the distance between the camera and the ground determines the size of the area being captured (Figure 7). The further away the camera is from the ground the larger the captured area. This also affects the image quality or ground pixel size.

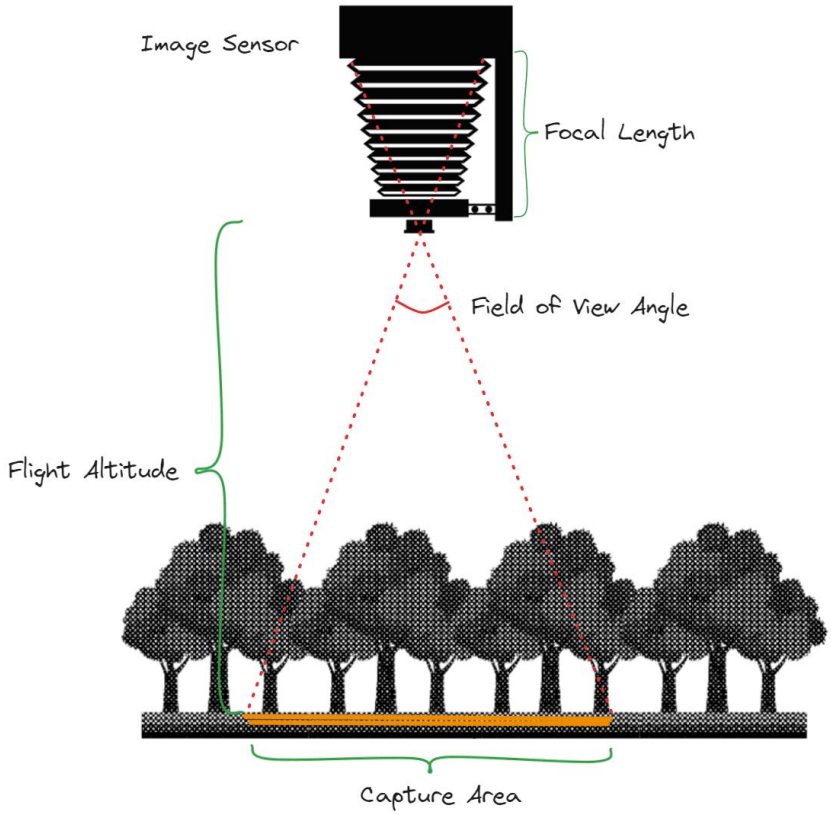

Figure 7. The correlation between flight altitude, the field of view, and the capture area.

Focal length and flight altitude also determine the flight pattern spacing distance. The downward-facing camera must cover a specified area and the next leg in the drone track must cover the area next to it (Figure 8). To calculate track spacing, the flight altitude must be determined. A form for calculating track spacing can be created. The focal length for the Parrot Anafi drone camera is 35 mm, but also HFOV 69° (Horizontal Field of View) is given by the manufacturer (24, sec. Parrot Anafi Technical Documentation), so the distance can be calculated using trigonometry:

$$W = tan\ 69° * \mathrm{h}$$

where *W* is the width of the capture area and *h* is the flight altitude. The distance between the flight legs is the width of the capture area.

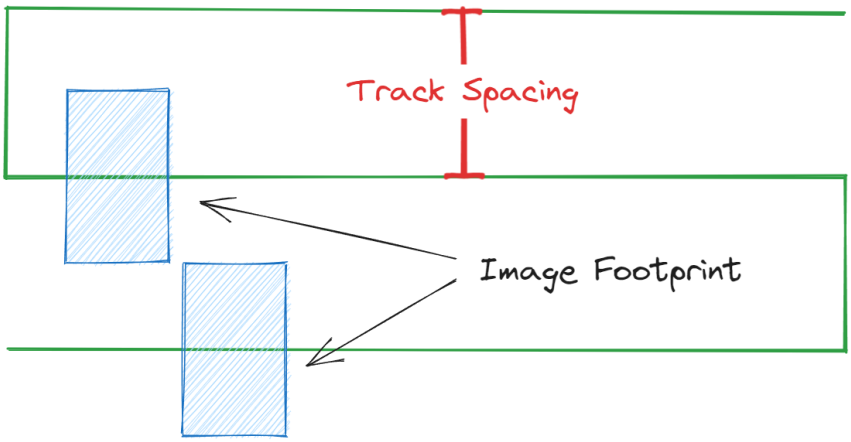

Figure 8. Pattern spacing can be determined from the image footprint.

4.4 Calculating Point of Interest (POI) Coordinates

To simplify the calculation of the point of interest (POI) location, an assumption that the location is on the ground (Figure 9), makes it easier to calculate. If drone location, altitude, attitude, and drone camera gimbal pitch are known, the location can be determined with trigonometry. The drone camera needs to be centered (the centering process is described in *5.6.3 Movement Library*) on the POI. The calculation doesn't consider terrain formations, or if the POI is above ground level.

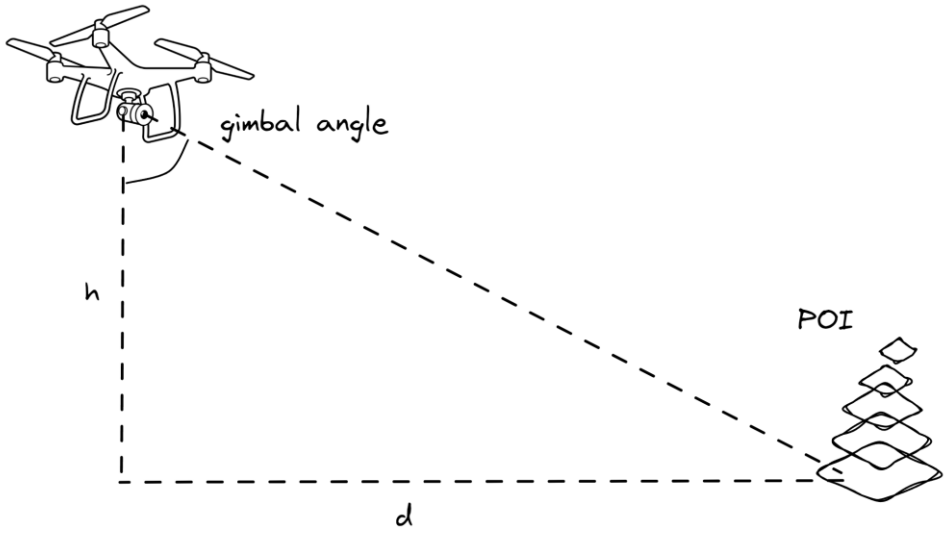

Figure 9. Ground distance d can be determined using trigonometry.

The ground distance to the point is:

$$d = \mathrm{h} * \tan \alpha$$

where d is the ground distance from the drone to the POI in meters, h is flight altitude and α is the camera pitch angle. The POI coordinate point can now be created using a Python library called GeoPy. For this function (Figure 10, line 319), only the current location, attitude (bearing), and distance are needed.

```
307 def getPOIfromCameraDirection(drone):
308     # Get current location
309     droneLocation = dronelocationToCoords(drone.drone.get_state(
310         GpsLocationChanged))
311     # Angle zero is when the camera is alligned horizontally,
312     # but when the camera faces down, the angle should be zero
313     GROUND_IS_ZERO_ANGLE = 90
314     camera_angle = GROUND_IS_ZERO_ANGLE - abs(
315         drone.drone.get_state(attitude)[0]['pitch_absolute'] )
316     drone_hdg = np.rad2deg(drone.drone.get_state(AttitudeChanged)['yaw'])
317     drone_alt = droneLocation.alt
318 
319     d_to_drone = (np.tan(np.deg2rad(camera_angle)) * drone_alt)
320     #returns a point p with latitude and longitude values
321     p = distance(meters= d_to_drone).destination(
322         (droneLocation.lat, droneLocation.lon), bearing= drone_hdg)
323 
324     obj_location = CoordPoint(lat= p[0], lon= p[1], alt=0)
325 
326     print(f"POI location: {obj_location}")
327 
328     return obj_location
```

Figure 10. Function that calculates the POI by using drone location and camera angle.

4.5 Generating a Search Pattern

One search pattern function was created for easier testing. A parallel search pattern was chosen due to its simplicity. The function parameters were two points in metric coordinates and flight altitude. The two points define an area where the first point is the starting point, and the second point is an opposite corner of a rectangle. The function creates a parallel search pattern from two points representing opposite corners of a rectangle and outputs an array of points(x, y).

First, the leg spacing is calculated with the form mentioned in *Determining Track Spacing and Altitude by Focal Length* chapter. When the leg spacing is known, the search area width is divided by

the leg spacing to get the number of legs. If the result is not an integer (it usually is not) one more leg is added to make sure the area is covered. The first point of the pattern is the first point added to the function. In a for loop the short and long leg points are generated as shown in Figure 11. The function generates a pattern that looks something like in Figure 12. However, if the width of the area is greater than the depth, the x and y parameters of the points are flipped after the for loop, to make sure the longer distance is always the longer leg.

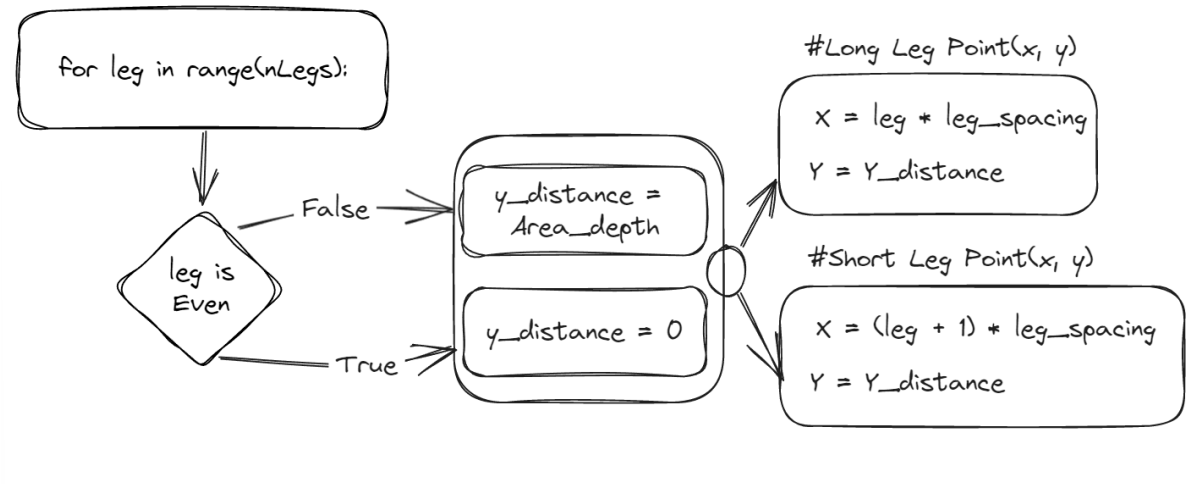

Figure 11. A for loop generating long and short leg points (x, y).

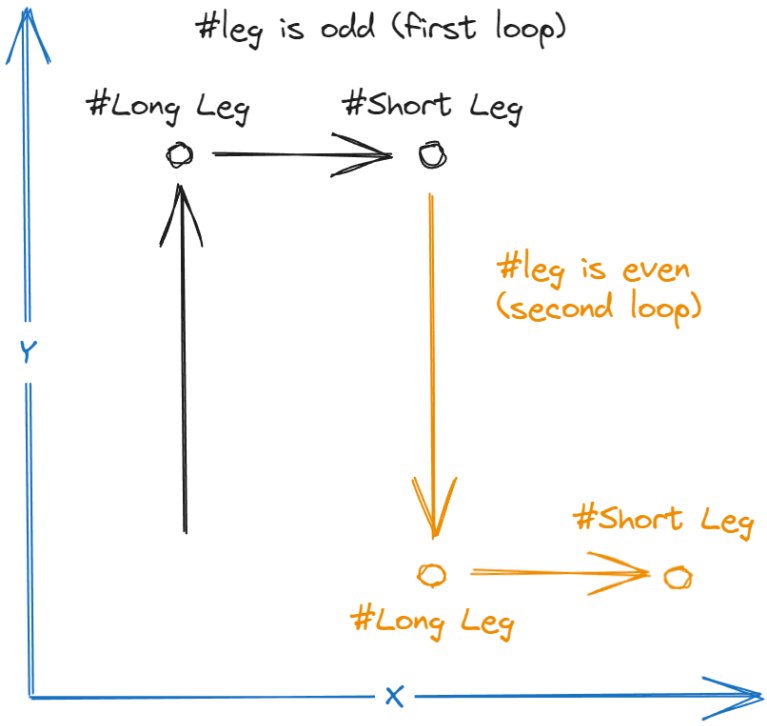

Figure 12. The pattern generation is shown in the coordination system.

5 SOFTWARE AND HARDWARE

In this section, the used hardware and software are introduced. Chapters 7 and 8 implement the previously introduced theory, software, and hardware. The drone receives commands from a controller linked to a computer that sends control messages to the drone and receives data back from the drone.

5.1 Image Detection Platform

For image detection, YOLO (You Only Look Once) version 8 was chosen due to its ease of use and fast detection time. The training results are introduced in Chapter *7 Creating and Evaluating Flamed Detection Model*. YOLO is an open-source platform with well-documented instructions. The first version of YOLO was introduced in 2015 and since then it has been developed to become one of the most used image detection platforms. (25).

YOLO training includes image labeling done by hand (creating a bounding box for the object) and after the labeling has been done, the images (x_train) are fed to the model together with a labeling file containing the bounding box coordinates as a vector (y_train). To tackle the problem with multiple objects in the same image, the image is divided into a grid and the bounding box center determines in which vector it belongs to (Figure 13*)*.

Each training image contains as many vectors as there are grid cells in the image. Vector parameters are P_c = a number of identified objects, B_x = x location of bounding box center, B_y = y location of bounding box center, B_w = width of the bounding box, B_h = height of the bounding box, C_1 = detected object class (in this case fire) and C_2 = detected object class (in this case a lantern in Figure 13). The number of classes can vary and, in this case, the Z axis of the whole image vector will increase or decrease. If the same grid vector contains two objects, the size of the z-axis will double. (7).

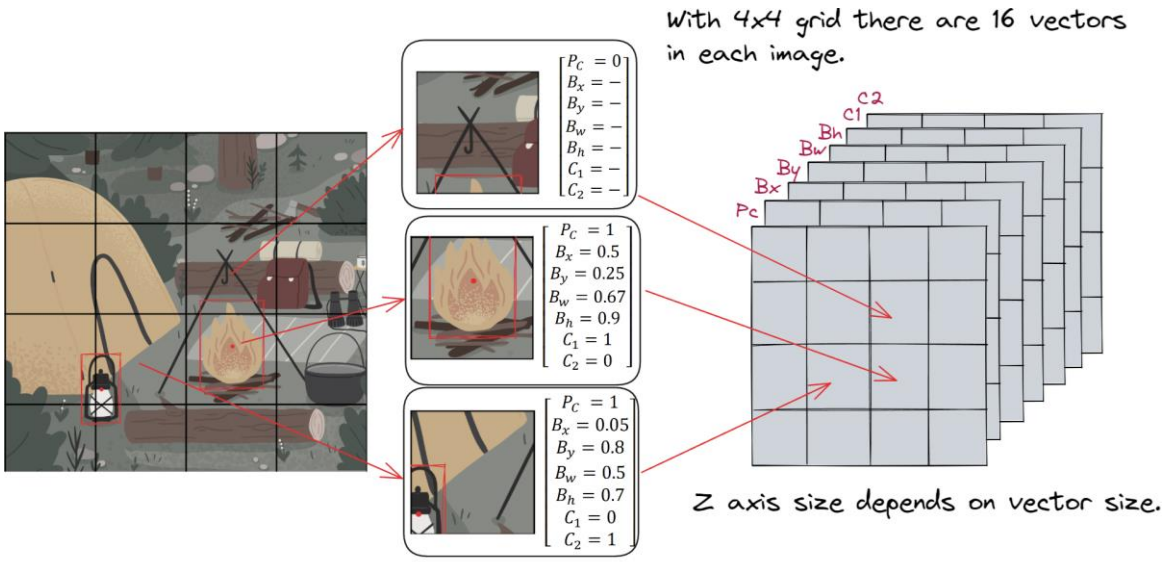

Figure 13. detected objects are stored in 3D vector.

5.2 Training Platform and Version Management

For easier workflow with AI training, the ClearML platform was chosen. It enables easy queuing and runs different tests on a local machine or server, dataset version management, and a good way for comparing tests with each other. GitHub was used to store code and keep up with different versions.

5.3 Simulation Software

Parrot Sphinx simulation software was used to test the code and the drone's performance. The Simulation software is made from Unreal Engine. Different terrains can be used to perform simulations and the drone can be operated the same way as in real life. Parrot Sphinx can also simulate different scenarios but lacks a ready-made scenario for a fire event. (26).

The simulator is created so that it can be used with a Parrot SkyController or with programs made by others. GPS coordinate system can also be used, and it can be changed if needed to simulate the area of operation. The drone can also be viewed from a map view from a web browser.

5.4 Unreal Engine 4.26

Parrot also provides their own version of Unreal Engine 4 software to create custom simulation environments and terrains. This is very useful if the drone needs to be tested in special conditions, such as forest fires. The terrain can be created from scratch and additional elements can be imported as Autodesk Filmbox (fbx) format or as Wavefront 3D Object (obj). (26).

The Epic Games account needs to be linked with the GitHub account to be able to download the source code. After that, the software can be installed. The installation procedure might take several hours. Some settings need to be done for the simulation to work properly but these procedures are documented in the Parrot Sphinx documentation. (26).

5.4.1 Test Level 1

The first level that was created with Unreal Engine includes only a ground and a flame in front of the drone spawn location. The aim was only to test how the drone stream would detect flames and then try and develop basic drone movement.

5.4.2 Test Level 2

Test level 2 was created to test different functions for drone control. The level contains only a flame, wall, and ground. The wall was added to prevent the drone from detecting the flame on takeoff. The flame is situated in the middle (Figure 14) and the spawn point of the drone is located on the right bottom.

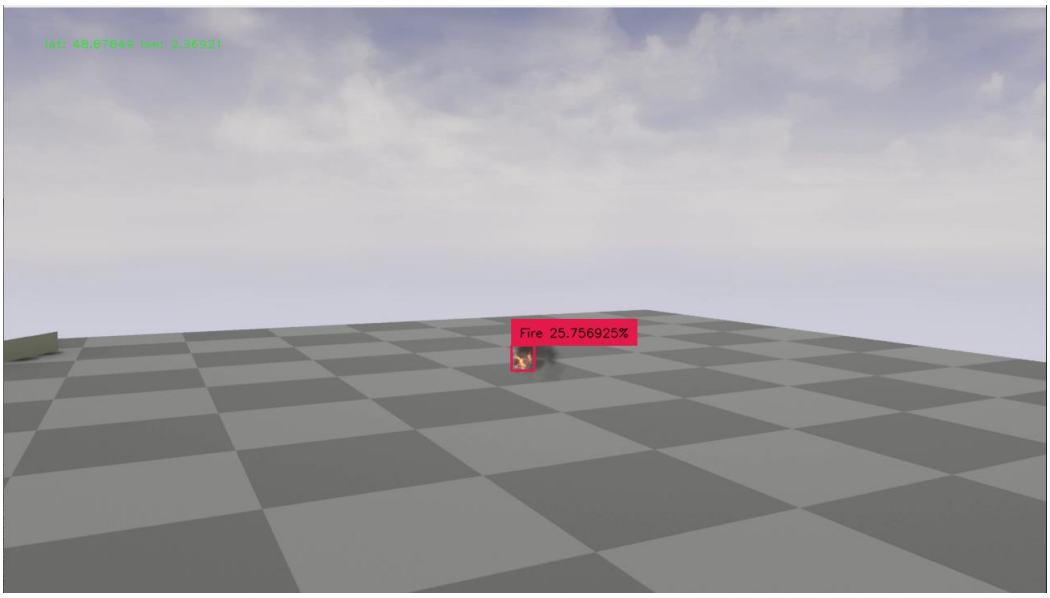

Figure 13. A screenshot taken from a drone stream from test level 2.

5.4.3 Test Level 3

Test Level 3 was created to test the drone with foliage included (Figure 15). However, due to a lack of knowledge in Unreal Engine development and the scope of the thesis, the lights and shadows were not created properly and the video stream from the camera was over-exposed.

Figure 14. Fire and foliage in Unreal Engine development software.

5.5 Drones

4 rotor drones are used in this proof of concept. Olympe SDK can be used to control three different drones, Parrot Anafi, Parrot Ai, and Parrot Anafi Thermal. Anafi Ai and Anafi are also supported in

the Sphinx simulator, so the testing is quite easy to conduct. The whole Anafi family is open source so everyone can make their programs to control the drones.

5.5.1 Parrot Anafi

Parrot Anafi (Figure 16*)* is a 4-rotor drone equipped with a 4K camera designed for consumer use. The drone can fly for 25 minutes with one battery. It can stream video for 720p resolution. The maximum transmission distance is 4 kilometers when the controller is used. The drone weighs only 320 grams and uses GPS and GLONASS for positioning. It can also fly in windy conditions (50km/h) (24).

Figure 16. Parrot Anafi (24, Anafi)*.*

5.5.2 Parrot Anafi Ai

Parrot Anafi Ai (Figure 17) can be used on the Sphinx simulator and the drone has some sophisticated capabilities on its own. To name a few, inbuilt object detection, obstacle avoidance, 4G connection, and stereo vision(24).

Figure 15. Parrot Anafi Ai (24, Anafi Ai)*.*

5.5.3 Parrot Anafi Thermal

Anafi Thermal cannot be used on the Sphinx simulator but could be controlled with Olympe SDK with Python programming language. It would be suitable for fire detection because of its inbuilt thermal camera. The drone itself is built on the same-looking frame as Parrot Anafi (24).

5.5.4 Parrot Anafi USA

Parrot Anafi USA (Figure 18) is designed mainly for use by authorities such as law enforcement and emergency services. There is also a military version of the drone with additional communication capabilities. The drone has an integrated thermal camera and RGB camera capable of 32 times zoom (24).

Figure 16. Parrot Anafi USA (24, Anafi USA).

6 CODE

Parrot drones can be programmed using open-source Olympe SDK in Python 3.9 programming language. Code done by using Olympe SDK can be run in the Sphinx simulator. This provides a good opportunity to test the program in a simulated environment before using real drones. To simplify the usage of the code, some modularity was added, and new classes and libraries were made.

6.1 Drone Class

The drone class includes the drone initialization function, and the main parameters, such as IP address and functions for connecting to the drone.

6.2 Detection Class

The detection class includes all the functions that are related to seeing the stream and object detection. The flame detection model was implemented and used by creating a detection object from this class. The YOLOv8 detection script provides box labeling, but a supervision library was used, so that it was possible to manipulate the labels. The supervision library also provided a method to retrieve the center point of the detected object. With the anchor point, it was possible to command the drone to turn towards the detected object (this is explained in the *6.3 Movement Library*).

6.3 Movement Library

The movement library includes all the functions that control drone movement and behavior. It also includes Point3D class and CoordPoint class that can be used when creating point objects. The library also includes functions such as a function for creating a parallel search pattern and a function for moving the drone to a coordination point.

The drone can receive commands that include expectations. For example, the drone can be commanded to hover and then fly to a specific point and after that hover again. With a *successful ()*-callback the completion can be checked.

A function was created to command the drone to center the camera toward the fire. Before being able to turn, some assist functions were. These were *panTo* and *isCentered* shown in Figure 19 describing the process. The panTo function takes in a Python dictionary class object that includes x and y components representing the detected object center point in the stream frame as shown in Figure 20. The turn angle (x-axis) is calculated as shown in Figure 21 and for the pitch angle (y-axis) the same method is used, only the HFOV is changed to VFOV (Vertical Field of View) and x-axis parameters with the y-axis.

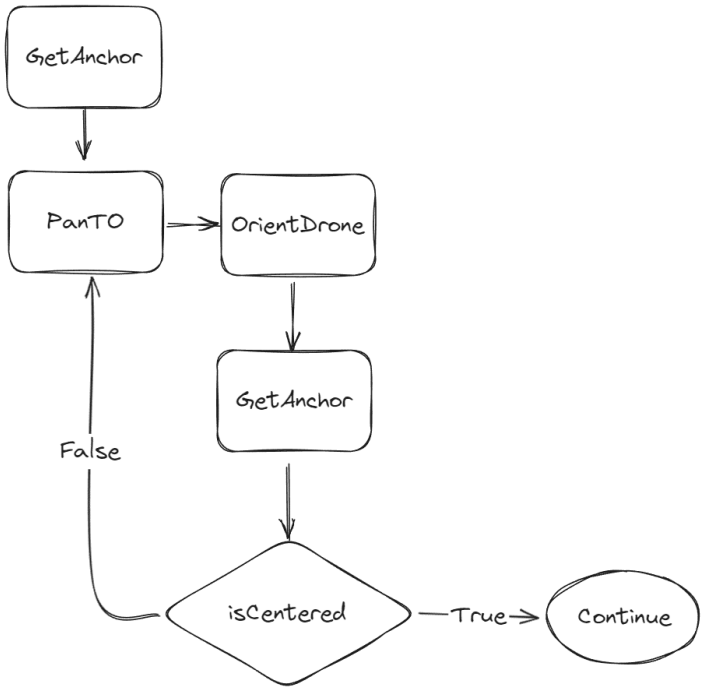

Figure 19. Flow chart of the drone frame orientation towards the detected object.

The calculated yaw (turn) and pitch are fed into a function that gives these parameters to the drone and after that, a new object center parameter, and this in turn is fed into a function that checks if the values are within desired area shown in Figure 20.

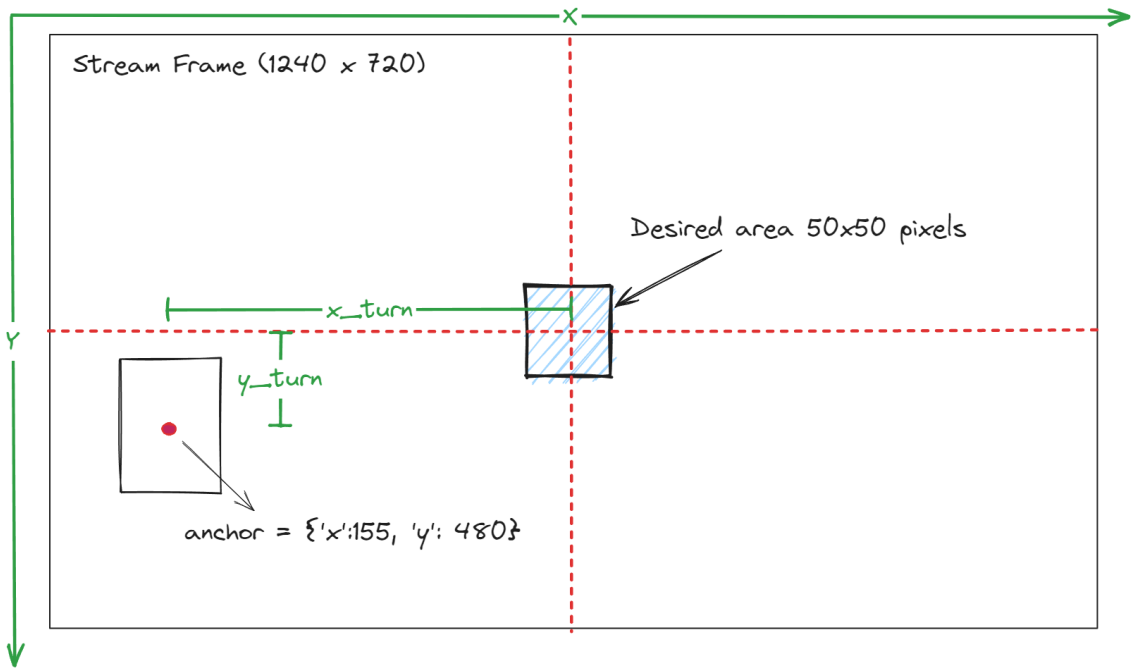

Figure 20. Detection center represented in pixel location in the frame.

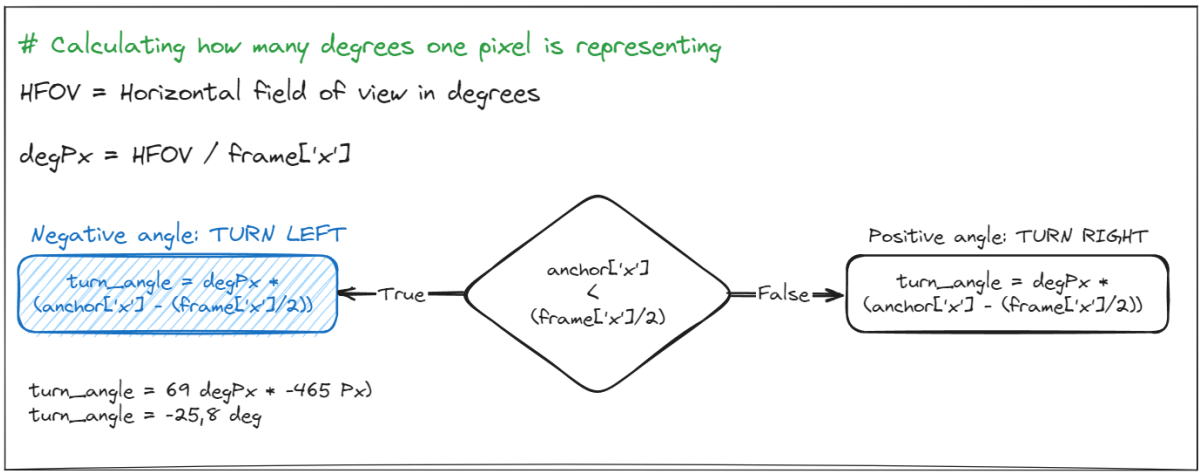

Figure 17. A Flow chart representing the turn direction calculated with the values in Figure 21.

7 CREATING AND EVALUATING A FLAME DETECTION MODEL

To simplify the process, smoke detection was discarded from the project. Smoke can be difficult to separate from fog and clouds. After the algorithm can detect flames, a new dataset including smoke can be added to teach new features in the future.

A dataset is needed for the creation of the YOLOv8 image detection model. For this purpose, three different datasets were used and a total of 18 models were created. The models were then evaluated using charts that were automatically generated with the training algorithm. The dataset creation was a time-consuming task, since the wanted objects needed to be labeled (Figure 18*).*

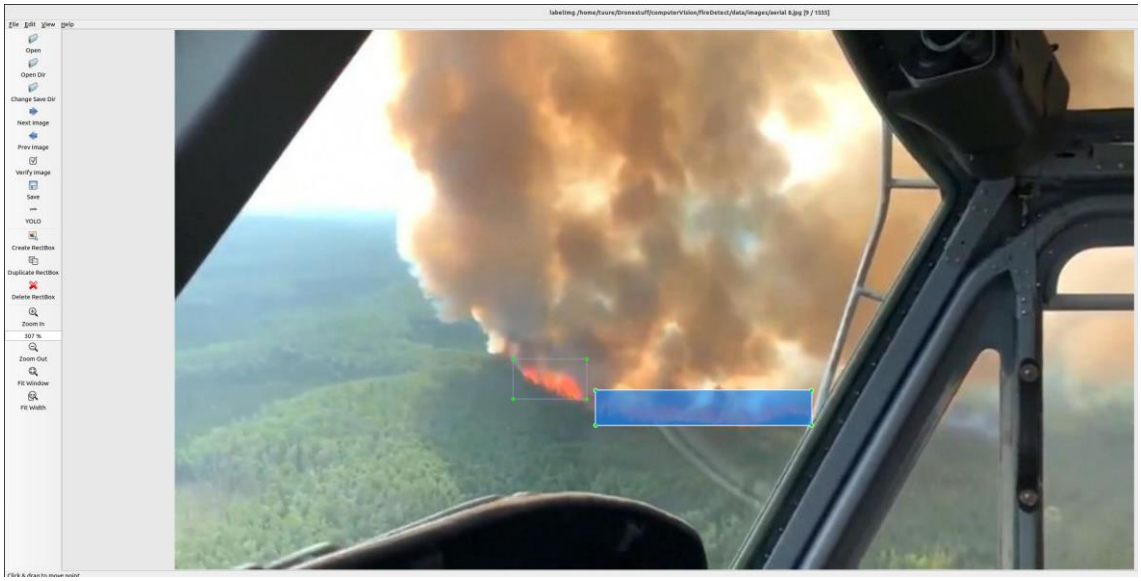

Figure 18. Labeling images for image detection.

7.1 Dataset 1

For the first dataset, a simple JavaScript code was used to download Google image search URLs to a text file. After creating text files from searches a python script was used to download the images (27). Used image search terms were *aerial photos of wildfire* (823 results), *campfire* (679 results), *fire* (469 results), in Finnish *nuotio metsästys (*translation, fireplace hunting*)* (664 results), and *wildfire* (646 results). After the script discarded unusable links and after a visual look through the dataset, the ready dataset consisted of 1518 images with 4215 instances. 3862 instances of fire, 332 for firefighters, and 21 for helicopters.

7.2 Dataset 2

The first dataset was mostly created for testing the algorithms, but it lacked good practices that should be used for model training. The new dataset consisted of 2433 images with 7640 instances. 6587 instances of fire, 1053 for firefighters. Helicopters were left off from the dataset, due to lack of images containing them. Also, 219 background images without any objects were included in the dataset. The dataset was divided into a training set containing 80 % of the images and a validation set containing 20% of the images.

The added datasets were found on the internet (17, 28), and they have been used in the same kind of experiments. Some of the new images were already labeled, but some needed labeling.

7.3 Dataset 3

The third dataset used the same data as dataset 2, but the data was not divided. The same data was used in training and validation. Only two experiments were made with this dataset, tests 9 and 10.

7.4 Generating Different Versions from Detection Models

Each of these datasets was experimented with shown in TABLE 1, except for dataset 3. For this dataset, only experiments 9 and 10 were done. Each experiment created a detection model that was evaluated. The batch size of 39 images was chosen, because the used GPU (Nvidia-based MSI GeForce 1080 ti, with 11GB VRAM) was limiting higher batches.

TABLE 1. Experiments for Flame Recognition Model

Test	1	2	3	4	5	6	7	8	9	10
Epochs	10	50	100	500	10	50	100	500	500	500
Batch Size	10	10	10	10	39	39	39	39	10	39

7.5 Evaluating the Model

The script that trains the model provides some good graphs to evaluate the created model. The most obvious way is to check some of the preview images where labeled and predicted images can be analyzed. When comparing Figure 19 which shows labeled images, with images in Figure 20 we can see that with only 10 epochs the model can predict fire quite well and the certainty is quite high in some cases.

Figure 19. Test 1 done with dataset 1. Labeled images.

Figure 20. Test 1 done with dataset 1. Predictions.

For further analysis, the provided graphs can give more information. Some basic terms to understand the graphs are **Precision** and **Recall**. Precision tells the precision of the predictions; of all the positive prediction (fire) are actually true (true positive or TP). Recall on the other hand tells how well the model can identify the positive cases from all the cases; cases are TP and FP (false positive). When trying to increase precision, the model might leave some uncertain predictions marked and this will decrease recall. On the other hand, increasing the recall will result in more false positives (FP).

F1-score will take both the precision and the recall into account and give an estimate of the overall prediction of the model. The value is given between 0 and 1, and the higher the score the better the model is.

The experiments with 500 epochs show the best results, however, the model might have been overfitted. Also, when looking at all confusion matrixes, for example, one shown in Figure 21, the created models confuse the background from fire very often. This can be because of poor labeling. In some cases, not all the flames were marked and for the model, it might be especially hard to detect an actual flame from orange and red-colored surroundings.

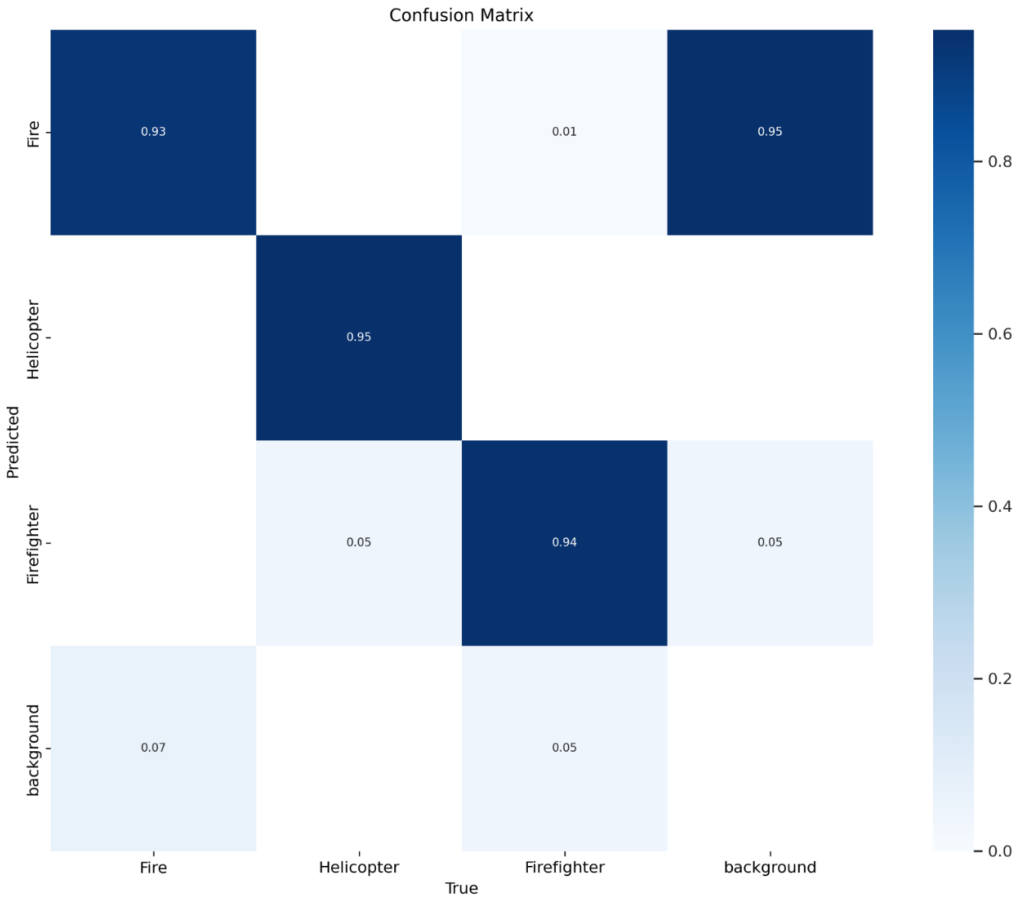

Figure 21. The model detects flames from the background better than actual fire in this confusion matrix. Dataset 1, batch 10, epochs 500.

To ease the evaluation of the charts, a table was created to get an overall picture of the experiments (TABLE 2). From the results shown in TABLE 2, an estimation can be made of which model was the best and which model succeeded poorly. From the simple comparison, the best overall model was made with experiment 18 (TABLE 2), and experiment 16 (TABLE 2) gained the lowest overall scores.

However, considering the limited understanding of image detection, a more comprehensive evaluation should be made to define the best model and how the results can be further improved. Nevertheless, the scope of this thesis does not require in-depth evaluation. Some noteworthy observations can still be made. The results indicate that the model began overfitting, where the model memorizes the training data rather than learning general patterns. One indication of overfitting is that tests 4 and 16 are identical except for the datasets used, the precision-recall values show a significant disparity. In experiment 4, the same data was used for validation and training, and this might lead to learning the data over time.

TABLE 2. Results from Flame Recognition Model Training

Experiment	Dataset	Batch	Epochs	Confusion matrix, fire [fire]	Confusion matrix, background [fire]	Recall-Confidence [fire]	Precision-Confidence [fire]	Precision-Recall [fire]	F1-Curve [fire]
1	1	10	10	0,53	0,95	0,9	0,951	0,573	0,58
2	1	10	50	0,72	0,91	0,89	0,966	0,773	0,75
3	1	10	100	0,82	0,92	0,95	0,947	0,857	0,82
4	1	10	500	0,93	0,95	0,97	0,882	0,946	0,9
5	1	39	10	N/A	N/A	N/A	N/A	N/A	N/A
6	1	39	50	0,72	0,94	0,9	0,914	0,791	0,78
7	1	39	100	0,85	0,94	0,96	0,923	0,88	0,85
8	1	39	500	0,94	0,94	0,98	0,953	0,848	0,92
9	2	10	10	0,29	0,89	0,82	0,952	0,292	0,38
10	2	10	50	0,46	0,95	0,8	0,987	0,412	0,42
11	2	10	100	0,45	0,94	0,77	1	0,398	0,42
12	2	10	500	0,42	0,93	0,79	0,885	0,404	0,42
13	2	39	10	0,34	0,96	0,8	0,963	0,353	0,4
14	2	39	50	0,41	0,9	0,79	0,938	0,412	0,42
15	2	39	100	0,43	0,95	0,74	0,956	0,389	0,4
16	2	39	500	0,41	0,94	0,69	0,932	0,345	0,4
17	3	10	500	0,88	0,94	0,94	0,914	0,901	0,88
18	3	39	500	0,89	0,86	0,95	0,892	0,92	0,9

8 TESTING THE CODE IN THE SIMULATOR

Prior to conducting tests in the simulator some essential components had to be developed, the flame detection model, a test level, and some code for instructing the drone to operate as wanted. These “components” were crucial for the evaluation of the autonomous flight and flame detection capabilities even in simulated environments.

8.1 Movement

The initial objective of the simulator testing involved developing a code capable of controlling and moving the drone. Once the connection to the drone was established it was commanded to take off and hover at a selected altitude. A queue parameter created by the search pattern function is fed to a function that takes the next waypoint from the queue after completion in the designated blue area in Figure 25.

8.2 Finding Fire

The drone is commanded to stream the front camera video and every frame is fed to a detection class function that checks if the frame contained fire. After checking if it contains fire, it will add a box and labels around the detected fire object, then the frame is shown in a window by using the cv2 library. If there are no detections, the stream will just show without any detection boxes.

The detection runs on its own thread and moves on its own (Figure 25). Another thread also runs to stop the drone’s movement thread if a fire is recognized. The drone then tries to center the camera toward the detected fire.

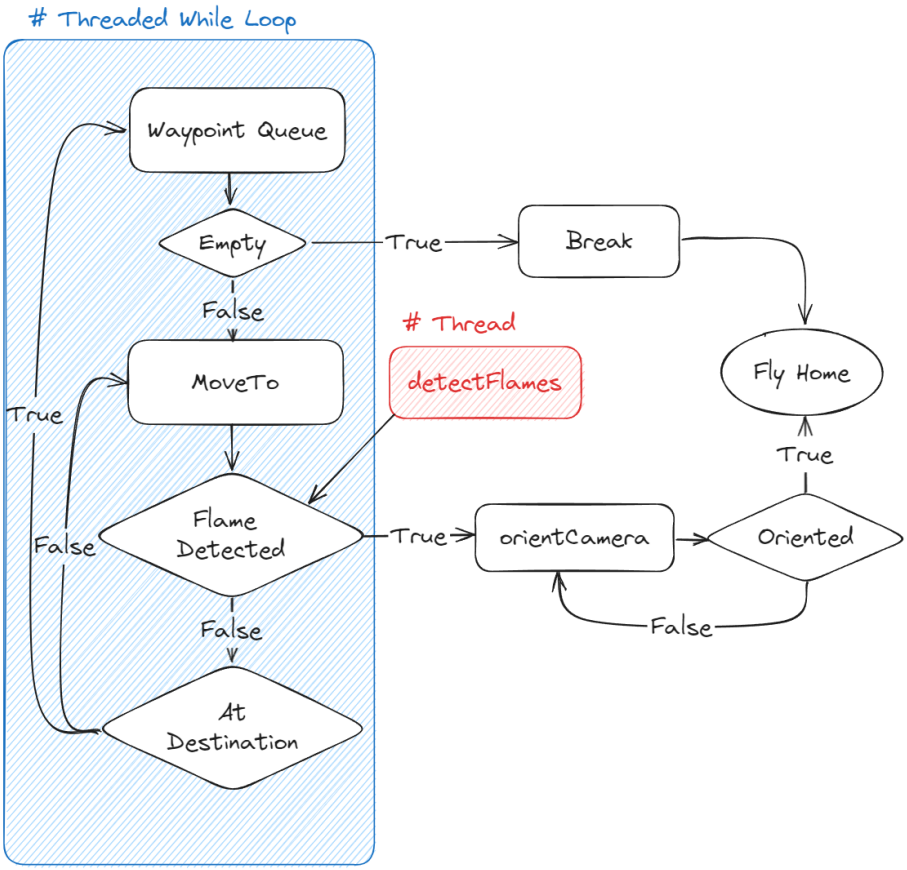

Figure 25. Script that handles the drone behavior in autonomous flight.

When the drone has centered the camera toward the detected object, the drone calculates the distance to the object and moves to a desired distance. The location of the fire was confirmed in the first tests, so the drone was commanded to fly to the fire after the distance was calculated.

8.3 Test Results

The code seemed to provide the promising potential for future work and confirms that a drone equipped with a sufficiently capable camera and a flame detection algorithm can be used to detect and monitor flames. Nonetheless, to work seamlessly, substantial efforts must be undertaken. For example, the drone lacks safety measures to force the drone to land or in case of emergency shut off the rotors to drop itself immediately.

The drone is not capable of detecting things on its own and the detection is done with a computer that controls the drone. Furthermore, the detection models tested in the simulator might have false recognitions from time to time. This can be prevented by increasing the certainty level so that the

script will only pass a detection if it is certain enough. Also, filters or buffers could be added to make sure the detection can be seen in more than one frame. The most obvious way is to improve the detection model.

Test level 3 needs more work with the lighting settings and foliage. It is essential to get at least some kind of understanding of how the drone behaves in an environment where there are obstacles, foliage, and light sources that might trick it to behave unexpectedly. For this, the simulation environment suits well and does not break expensive drones.

9 DISCUSSION, CONCLUSIONS, AND FURTHER IDEAS

Throughout the process of creating a proof of concept from the idea of using drone swarms to detect and monitor forest fires, some noteworthy considerations emerged. There are no sources added to provide verification for the given arguments and they have only arisen ideas from the author. This chapter aims to explore and address these ideas and observations in detail. The scale of the thesis limited the execution of some experiments, such as the use of drone swarms. Nevertheless, the created proof of concept enables future development.

9.1 About the Concept

To achieve autonomous or even semiautonomous flight of drone swarms with the approach mentioned in this thesis, a variety of improvements should be made. Some of the issues were already raised in chapter *8.3 Test Results*. Safety measures play a crucial role in drone operations and operating a swarm of drones raises a variety of questions; How an individual drone behaves in case of emergency? How about the whole swarm? Are the drones aware of each other or is the controller handling everything? How about other air traffic? What are the current rules for operating a drone swarm? Is it possible to fly outside the line of sight?

Before commencing field tests, it is essential to address some of these questions, although not all aspects need to be fully prepared in a controlled environment. If a drone, or drones, can be directly controlled by the operator, the automatic safety measures do not need to be developed completely.

One of the most obvious additions to the drone is a thermal camera. A thermal camera can detect heat sources and give readings from them. The drones could also be equipped with temperature sensors to make sure the monitored fire does not cause a malfunction to the drone. The monitor window should also some additional information, such as the location of the fire, size of the fire, wind direction, and the bearing and location of the drone. For longer periods of monitoring, the drone should automatically be sent to a battery swap or charging station, and maybe some staggering should be added to enable constant monitoring.

9.2 Improvements to the Code

There is a plethora of improvements for the code to work better, but the scale of the thesis set some limitations on what could be achieved with limited time resources. Some of the most important lacking things were the use of multiple drones and orbiting behavior. However, both are already thought concepts and they “only” lack the implementation. To make future development possible for this code, ideas should be introduced.

9.2.1 Orbiting the Drone

The Parrot Olympe SDK includes some orbiting behavior, However, the documentation about it was a bit unclear. This issue has been raised in the parrot development forum and hopefully, an answer is received by the developers.

The drone can be commanded to look towards a point of interest represented in latitude and longitude form, but when commands for movements are given, the drone does not keep the focus on the POI. It can hold its orientation but does not lock towards the center. To tackle this problem a function that will create an array of coordination points in a circle every ten degrees could be made. This array could be passed to a function that commands the drone to move to the points in an orderly fashion. After each move, the drone could be turned towards the center.

9.2.2 Using Multiple Drones

To be able to operate with multiple drones, a server should be used for communication between computers that control the drones. This server could work similarly as an ATC (air traffic control), in a way that it knows the location and speed of every drone used in the swarm. Additionally, it would also receive data from the detected fire and alarm the other drones of that swarm to move to the detection location. It could also assign different drones a different cycle from the orbiting function to avoid a collision if the drones are commanded to orbit the same object.

The server could also stream the video stream to other locations, for example, to the command center or the firefighters on the ground.

REFERENCES

1. Saffre, Fabrice, Hildmann, Hannu, Karvonen, Hannu and Lind, Timo. 2022. Monitoring and Cordoning Wildfires with an Autonomous Swarm of Unmanned Aerial Vehicles. MDPI. DOI: 10.3390/drones6100301

2. Carnicer, Jofre, Alegria, Andrés, Giannakopoulos, Christos, Giuseppe, Francesca Di, Karali, Anna, Koutsias, Nikos, Lionello, Piero, Parrington, Mark and Vitolo, Claudia. 2022. Global warming is shifting the relationships between fire weather and realized fire-induced CO 2 emissions in Europe. Scientific Reports. Online. Vol. 12, p. 10365. Cited 26 March 2023. DOI: 10.1038/s41598-022-14480-8

3. Devansh-. 2022. Machine Learning Made Simple. Why Small Batch sizes lead to greater generalization in Deep Learning | by Devansh- Machine Learning Made Simple | Geek Culture | Medium. Medium. Online. Cited 22 April 2023. Available from: https://medium.com/geekculture/why-small-batch-sizes-lead-to-greater-generalization-in-deep-learning-a00a32251a4f. DOI: 10.37266/ISER.2022V10I2.PP159-165

4. Blakenship, Rory, Bluman, James and Steckenrider, Josiah. An Investigation of Search Algorithms for Aerial Reconnaissance of an Area Target. Industrial and Systems Engineering Review. Online. 25 December 2022. Vol. 10, no. 2, p. 159–165. Cited 23 April 2023.

5. Alizadeh, Mohammad Reza, Abatzoglou, John T., Luce, Charles H., Adamowski, Jan F., Farid, Arvin and Sadegh, Mojtaba. 2021. Warming enabled upslope advance in western US forest fires. Proceedings of the National Academy of Sciences of the United States of America. Online. Vol. 118, no. 22, p. e2009717118. Cited 26 March 2023. DOI: 10.1073/pnas.2009717118

6. Ayyadevara, V Kishore and Reddy, Yeshwanth. 2020. Modern Computer Vision with PyTorch. Packt Publishing Ltd. Online. Cited 22 April 2023. Available from: https://learning.oreilly.com/library/view/modern-computer-vision/9781839213472/71a7d3fa-dd2c-42d4-bdab-5fbd8e8f4bdd.xhtml

7. Codebasics. 2020. What is the YOLO algorithm? | Deep Learning Tutorial 31 (Tensorflow, Keras & Python) - YouTube. *YouTube*. Online. Cited 23.4.2023. Available from: https://www.youtube.com/watch?v=ag3DLKsl2vk&ab_channel=codebasics

8. Dadhich Abhinav. 2018. Practical Computer Vision. *Packt Publishing*. Online. Cited 26.3.2023. Available from: https://learning.oreilly.com/library/view/practical-computer-vision/9781788297684/

9. Deeplizard. 2017. Convolutional Neural Networks (CNNs) explained - YouTube. *YouTube*. Online. Cited 18.4.2023. Available from: https://www.youtube.com/watch?v=YRhxdVk_sIs

10. Devansh-. 2022. Machine Learning Made Simple. Why Small Batch sizes lead to greater generalization in Deep Learning | by Devansh- Machine Learning Made Simple | Geek Culture | Medium. *Medium*. Online. Cited 22.4.2023. Available from: https://medium.com/geekculture/why-small-batch-sizes-lead-to-greater-generalization-in-deep-learning-a00a32251a4f

11. Kandel, Ibrahem and Castelli, Mauro. 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*. Vol. 6, no. 4, p. 312–315. DOI 10.1016/J.ICTE.2020.04.010.

12. Karvonen, Hannu, Honkavaara, Eija, Röning, Juha, Kramar, Vadim and Sassi, Jukka. *Using a Semi-Autonomous Drone Swarm to Support Wildfire Management-A Concept of Operations Development. Working paper. Not yet published.*

13. Michelucci, Umberto. 2019. Advanced applied deep learning: Convolutional neural networks and object detection. *Advanced Applied Deep Learning: Convolutional Neural Networks and Object Detection*. Online. Cited 23.4.2023.

14. Moritz, Max A., Parisien, Marc-André, Batllori, Enric, Krawchuk, Meg A., Van Dorn, Jeff, Ganz, David J. and Hayhoe, Katharine. 2012. Climate change and disruptions to global fire activity. *Ecosphere*. Vol. 3, no. 6, p. art49. DOI 10.1890/ES11-00345.1.

15. Nelson, Abhilash. 2020. Computer Vision: You Only Look Once (YOLO) Custom Object Detection with Colab GPU. *O'Reilly*. Online. Cited 23.4.2023. Available from: https://learning.oreilly.com/videos/computer-vision-you/9781800563865/9781800563865-video6_1/

16. Paperspace. Artificial Intelligence Wiki. *Paperspace*. Online. Cited 23.4.2023. Available from: https://machine-learning.paperspace.com/

17. Rosebrock, Adrian. 2018. Fire and smoke detection with Keras and Deep Learning. *Fire and smoke detection with Keras and Deep Learning*. Online. Cited 27.3.2023. Available from: https://pyimagesearch.com/2019/11/18/fire-and-smoke-detection-with-keras-and-deep-learning/

18. Saffre, Fabrice, Hildmann, Hanno, Karvonen, Hannu and Lind, Timo. 2022. *Monitoring and Cordoning Wildfires with an Autonomous Swarm of Unmanned Aerial Vehicles*. MDPI.

19. Voulodimos, Athanasios, Doulamis, Nikolaos, Doulamis, Anastasios and Protopapadakis, Eftychios. 2018. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*. Vol. 2018. DOI 10.1155/2018/7068349.

20. Williams, Andrew P, and Scharre, Paul D. 2015. *Autonomous Systems Issues for Defence Policymakers* Online. Hague: NATO Communications and Information Agency. Cited 29.4.2023. Available from: https://apps.dtic.mil/sti/citations/AD1010077

21. IAMSAR Search Patterns Explained with Sketches - Oways Online. Online. Cited 16.4.2023. Available from: https://owaysonline.com/iamsar-search-patterns/

22. Aerial forest fire surveillance - Guidance and advice - Government agencies - Aluehallintovirasto. *Aluehallintovirasto*. Online. Cited 16.4.2023. Available from: https://avi.fi/en/services/government-agencies/guidance-and-advice/aerial-forest-fire-surveillance

23. What Is Focal Length in Photography? *Photography Life*. Online. Cited 23.4.2023. Available from: https://photographylife.com/what-is-focal-length-in-photography

24. Parrot | European leader in professional drones. *Parrot Drone SAS*. Online. Cited 22.5.2023. Available from: https://www.parrot.com/en

25. Index - Ultralytics YOLOv8 Docs. *Ultralytics*. Online. Cited 23.4.2023. Available from: https://docs.ultralytics.com/

26. What is Parrot Sphinx - 2.11. *Parrot Drone SAS*. Online. Cited 14.5.2023. Available from: https://developer.parrot.com/docs/sphinx/index.html

27. How to create a deep learning dataset using Google Images - PyImageSearch. *PyImageSearch*. Online. Cited 23.4.2023. Available from: https://pyimagesearch.com/2017/12/04/how-to-create-a-deep-learning-dataset-using-google-images/

28. OlafenwaMoses/FireNET: A deep learning model for detecting fire in video and camera streams. *GitHub*. Online. Cited 22.5.2023. Available from: https://github.com/OlafenwaMoses/FireNET