NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Early Forest Fire Detection using UAV and Computer Vision

Dimitrios A. Foteinos

Supervisor: **Stathes Hadjiefthymiades,** Professor NKUA

ATHENS

OCTOBER 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έγκαιρη ανίχνευση δασικών πυρκαγιών με χρήση μη επανδρωμένου εναέριου οχήματος και μηχανικής όρασης

Δημήτριος Α. Φωτεινός

Επιβλέπων: **Ευστάθιος Χατζηευθυμιάδης,** Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΟΚΤΏΒΡΙΟΣ 2022

BSc THESIS

Early Forest Fire Detection using UAV and Computer Vision

Dimitrios A. Foteinos
S.N.: 1115001700181

SUPERVISOR: **Stathes Hadjiefthymiades,** Professor NKUA

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έγκαιρη ανίχνευση δασικών πυρκαγιών με χρήση μη επανδρωμένου εναέριου οχήματος και μηχανικής όρασης

Δημήτριος Α. Φωτεινός
Α.Μ.: 1115001700181

ΕΠΙΒΛΕΠΩΝ: **Ευστάθιος Χατζηευθυμιάδης,** Καθηγητής ΕΚΠΑ

ABSTRACT

In recent years, a major environmental problem with huge economic consequences that concern most European countries are forest fires. Classical identification and treatment techniques are found to be insufficient with large losses occurring each year due to them. There have been many proposals in the field of fire detection, with most of them being very costly and technologically advanced, requiring large technological infrastructure to maintain them.

The solution we propose in this research concerns fire identification using UAVs combined with computer vision. More specifically, we have trained an object recognition model (Yolov5) through a custom fire dataset (images). The data collected by the drone are sent through the computer to smart devices and through an application that the fire authorities will have installed on their mobile phones, they can immediately see the place, the time, the date and also the photo of the fire, in order to intervene immediately and control it effectively.

SUBJECT AREA: Computer Vision, Drone Programming, Mobile App Development

KEYWORDS: Unmanned Aerial Vehicles, Object Recognition, Fire Identification, Smart Devices, Forest Fires

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, ένα σημαντικό περιβαλλοντικό πρόβλημα με τεράστιες οικονομικές συνέπειες που απασχολεί τις περισσότερες ευρωπαϊκές χώρες είναι οι δασικές πυρκαγιές. Οι κλασικές τεχνικές αναγνώρισης και αντιμετώπισης είναι ανεπαρκείς με μεγάλες απώλειες να σημειώνονται κάθε χρόνο εξαιτίας τους. Τα τελευταία χρόνια έχουν γίνει πολλές προτάσεις στον τομέα της πυρανίχνευσης, με τις περισσότερες από αυτές να είναι πολύ δαπανηρές και τεχνολογικά προηγμένες, με αποτέλεσμα απαιτούν αρκετά μεγάλες τεχνολογικές υποδομές για τη συντήρησή τους.

Η λύση που προτείνουμε σε αυτή την έρευνα αφορά την αναγνώριση πυρκαγιάς με χρήση μη επανδρωμένου αεροσκάφους σε συνδυασμό με μηχανική όραση. Πιο συγκεκριμένα, έχουμε εκπαιδεύσει ένα μοντέλο αναγνώρισης αντικειμένων (Yolov5) μέσω ενός προσαρμοσμένου συνόλου δεδομένων (εικόνων) πυρκαγιάς. Στη συνέχεια, το drone μας κατά την πτήση χρησιμοποιεί αυτό το συγκεκριμένο μοντέλο για τον εντοπισμό δασικών πυρκαγιών. Στη συνέχεια, τα δεδομένα που συλλέγουμε από το drone αποστέλλονται μέσω του υπολογιστή σε έξυπνες συσκευές και μέσα από μια εφαρμογή που θα έχουν εγκαταστήσει οι πυροσβεστικές αρχές στα κινητά τους τηλέφωνα θα μπορούν να δουν αμέσως τον τόπο, την ώρα, την ημερομηνία αλλά και τη φωτογραφία της πυρκαγιάς , προκειμένου να παρέμβουν άμεσα για την ελέγξουν αποτελεσματικά.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Όραση, Προγραμματισμός Μη Επανδρωμένου Αεροσκάφους, Ανάπτυξη Εφαρμογών σε Κινητά

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μη Επανδρωμένο Αεροσκάφος, Αναγνώριση Αντικειμένων, Αναγνώριση Φωτιάς, Έξυπνες Συσκευές, Δασικές Πυρκαγιές

Special thanks to my family and friends
who with their constant support
always gave me strength to move forward.

ACKNOWLEDGEMENTS

For the preparation of this thesis, I would like to especially thank my supervisor, Professor Stathis Hatziefthymiadis, where he gave me the opportunity to implement my idea and with his valuable help and constant feedback, it was able to be completed successfully. I am really grateful to have been given the opportunity to implement this project and to be involved with all these interesting topics.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis is an essential part of the prerequisites for obtaining a Bachelor's degree in Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens. The main motivation for the development of this particular project was my need to contribute to a solution in a major environmental and social problem with the help of technology. I hope to be able to help with other similar problems in the future.

1. INTRODUCTION

In recent years, a huge environmental problem that concerns most countries on the planet and especially in Europe, is that of forest fires. Forest fires contribute to global greenhouse gas emissions and negatively affect public health, the economy and the provision of ecosystem services. In the humid tropics, fires are largely human-caused and lead to forest degradation. Studies have shown changing fire dynamics around the world due to changing climate and land use. [17]

The number of forest fires due to climate change is continuously growing. With the expansion of the European Union (EU from now on) appropriate defence against forest fires becomes more and more important. The number of annual forest fires within the EU is between 50–70 thousand, reaching 3–5 thousand square kilometres and causing a lot of financial problems, losing millions of Euros. It was recognized by environmental experts of the EU that forest fires cause significant ecological, economic and social problems in many European countries with possible long term consequences to the natural environment and the economy. Member states of the EU have created common directives and national regulations for the protection of forests and prevention of forest fires and carry on different scientific research in this topic. Planning of environmental security has become a primary task. Investigation of forests' structures, proper knowledge on the development of forest fires and their impact on the environment can help in the protection against forest fires and in fighting against large scale forest fires, which is a complex and difficult task for defence organizations and personnel. [8]

Many ways to identify forest fires have been proposed in recent years. Some of them are, for example, a network of thermal cameras, where identification can be made through them. Also, in recent years, satellite coverage has helped to identify and predict the course of the fire. Unfortunately, these systems are very expensive and require advanced technological infrastructures to maintain and be in continuous operation.

The idea we propose in this thesis is an IoT system, which with the help of machine learning and a Drone, can help in the early identification of forest fires. More specifically, we have trained a machine learning model through a custom fire dataset, and loaded it into our Drone. Thus, in mid-flight, we are able to locate a forest fire with great precision and send constant data from our Drone to a computer. Then, the computer in turn sends this data through an Android application to smart devices, which can be installed by the fire authorities, and thus intervene directly at the point where the fire has started. Through this low-cost concept, we achieve automation in the identification of forest fires, and significantly reduce a large part of the workforce, which can be used for other purposes.

This project is divided into 3 main parts. In *[section 4]*, we do a spatial analysis of the area of interest, as for example we study the elevation of the ground, we measure the area and finally we obtain data that are important in the prediction of a fire, such as for example temperature, humidity, the chance of rain but also the speed and direction of the wind in the area.

In *[section 5]* we train the machine learning model we use for fire recognition. Specifically, we use a custom fire dataset for training data, which consists of 412 images for training and 90 images for validation.

In *[section 6]* , we analyze the software we have written for the Drone. Specifically, how it moves, how it takes photos and above all how it connects to the machine learning model we have trained. We also analyze the Android application we have created.

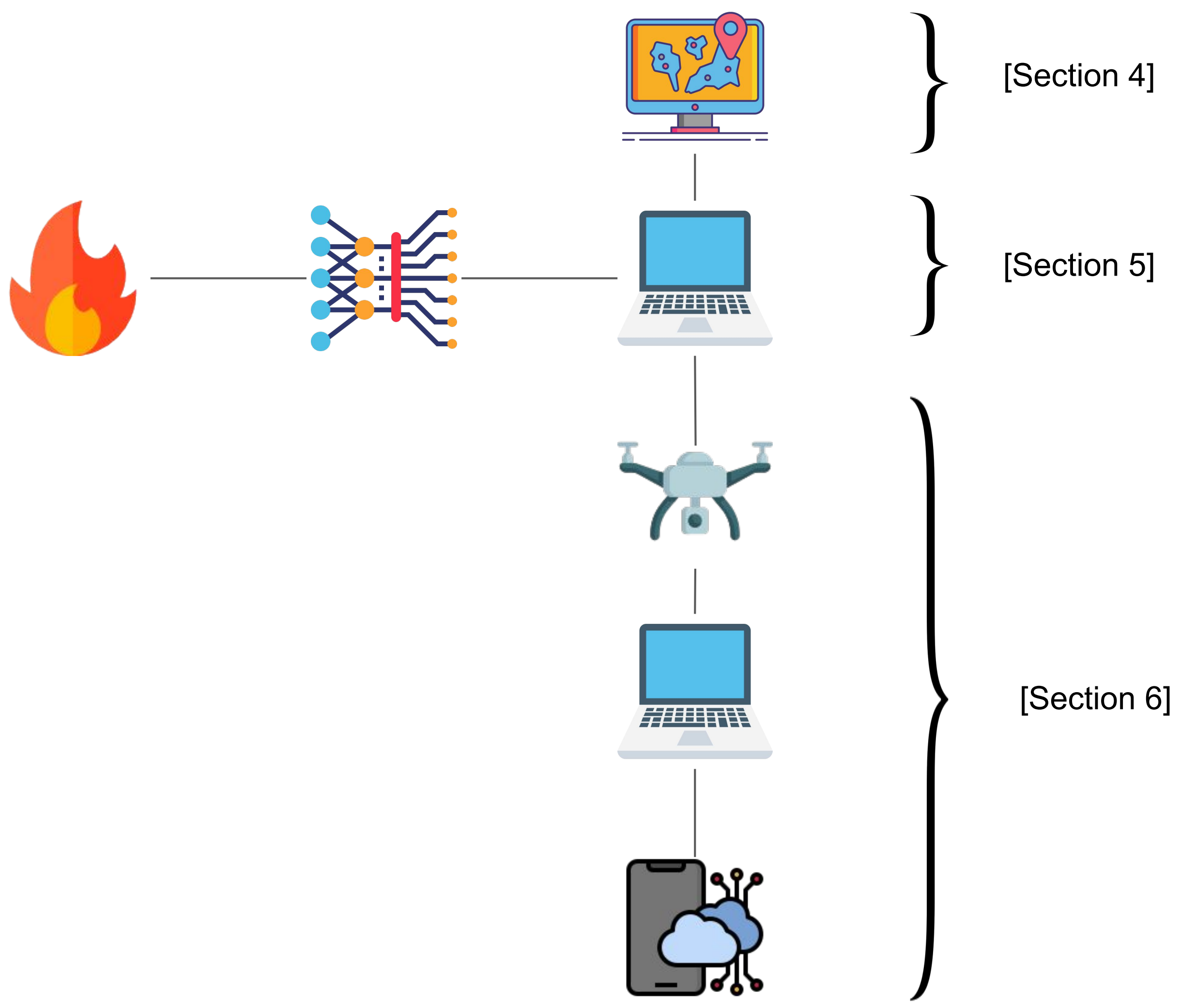


Figure 1.1: IoT System Workflow

2. BACKGROUND AND RELATED WORK

2.1 Forest Fires

First, to get to the point of immediately detecting a forest fire, we must study them. How are they caused, what factors influence it and what is the cost of a forest fire, i.e. how does it affect the economy and the bioclimatic character of a country. A *wildfire*, *forest fire*, *bushfire*, *wildland fire* or *rural fire* is an unplanned, uncontrolled and unpredictable fire in an area of combustible vegetation starting in rural and urban areas. [6]

2.1.1 What are they affected by?

Let us now consider the ”ingredients” of a fire. Scientists and firefighters have come up with a model for describing a fire, which explains the factors that cause a fire to start and continue. This schema is also known as the Fire Triangle [5].
The Fire Triangle is a simple way of understanding the three elements a fire needs to ignite: each side of the triangle represents one of the three ingredients – oxygen, heat, and fuel – demonstrating the interdependence of these ingredients in creating and sustaining fire. A fire naturally occurs when the elements are present and combined in the right mixture, meaning that fire is actually an event rather than a thing. When there is not enough heat generated to sustain the process, when the fuel is exhausted, removed, or when the oxygen supply is limited, then a side of the triangle is broken and the fire will exhaust.

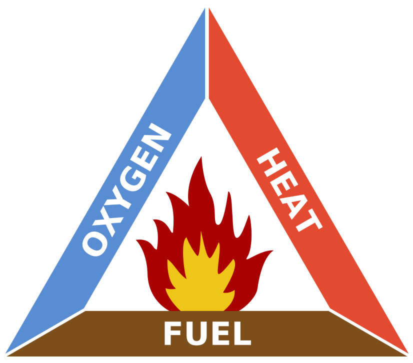


Figure 2.1: Fire Triangle

Heat: A heat source is responsible for the initial ignition of fire, and heat is also needed to maintain the fire and permit it to spread. Heat allows fire to spread by removing the moisture from nearby fuel, warming surrounding air, and preheating the fuel in its path, enabling it to travel with greater ease.

Fuel: Fuel is any kind of combustible material, and is characterized by its moisture content (i.e. how wet the fuel is), size and shape, quantity, and the arrangement in which it is spread over the landscape. The moisture content determines how easily that fuel will burn.

Oxygen: The oxidizer is the third component of the chemical reaction. In most cases, is simply comprised of the ambient air, and in particular one of its components, oxygen. Air contains about 21% oxygen, and most fires require at least 16% oxygen in order to ignite.

Now that we have seen the 3 pillars of a fire, let's look at the factors that contribute to its initiation, i.e. its ignition. As is obvious, the starting factors of a fire are 2. *Natural Causes*, such as lightning, very high temperatures in combination with flammable biological materials contained in forests. And understandably, *Human-induced* Wildfires in recent years have increased rapidly, for economic as well as political reasons.

- **Natural Causes of Wildfires** [2]:

- ***Lightning*** is the most common ignition source that causes the vast majority of wildfires. There are two types of lightning: cold and hot. Cold lightning is usually of short duration and thus rarely a cause of wildfires. The same cannot be said of hot lightning: currents in hot lightning have less voltage but occur for a longer period of time. Because of the intense heat it generates, hot lightning accounts for the majority of natural fires. While this natural phenomenon is completely unpredictable, adequate land management and landscape fire management planning can significantly diminish the intensity of wildfires and prevent unnecessary deaths and the displacement of people and animals.

- ***Climate change*** is undoubtedly the biggest trigger of extreme lightning storms. Warmer and longer summers heat up the land surface. This, coupled with an increase in carbon emissions, causes stronger updrafts that are more likely to produce more powerful and frequent lightning. A 2014 study estimates a 12% increase in the frequency of lightning strikes with every one degree Celsius increase in temperature. In Canada's province of British Columbia, for example, hot lightning causes 60% of the region's wildfires in an average year. The devastating and record-breaking 2020 Bay Area fire that destroyed 5 million acres of land, over 10,000 structures and killed 33 people was also a consequence of lightning storms. These hit the state following two intense heat waves which saw record high temperatures all over the west coast occurring over multiple days.

- **Human-induced Wildfires**:

- ***Human-related events*** that can ignite fires range from open burning such as campfires, equipment failure, and the malfunction of engines to debris burning, negligent discarding of cigarettes on dry grounds as well as other intentional acts of arson. The latter accounts for one of the most common causes of wildfires.

  According to government sources, 40% of wildfires that affect British Columbia in an average year are human-induced. In the US, the amount is more than double, with nearly 85% of the nearly 100,000 wildland fires that affect North America every year caused by human activities, according to data from the National Park Service.

  Here, man-made fires have tripled the length of North America's fire seasons between 1992 and 2012, from 46 to 154 days. Over the 21-year study period, the major causes were debris burning and arson, while campfires and fireworks were responsible for 'only' 5% of fires. Furthermore, an analysis of more recent California fires found that human-sparked wildfires are more extreme and destructive than nature-induced ones as they move more than twice as fast, spreading about 1.83 kilometres per day.

2.1.2 Forest Fires in EU

This year (2022) was a disastrous year for the countries of the European Union. Specifically, Forest fires have burned a record 700,000 hectares in the EU so far this year - the biggest amount since records began from 2006. The area of 700,000 hectares is nearly three times the size of Luxembourg or roughly the same size as Azerbaijan.

Figures from the *European Forest Fire Information System* (EFFIS) [3] show that **Spain** has been the most heavily impacted so far with more than 297,000 hectares burned. It is followed by **Romania** (149,278), **Portugal** (103,462), **France** (61,911), and **Italy** (51,056).

The number of fires has also shot up in recent years with more than 2,300 fires recorded across the bloc by mid-August, well above the average over the 2006-2021 period of 1,349 fires. Below, we present an Annual report of 2022 on the 10 countries from EU that were the most damaged by forest fires.

Table 2.1: Annual Fire Reports based on EFFIS

Country	**Country Area (ha)**	***Year 2022 (ha)***	**Annual Avg. 2006 - 2021 (ha)**
ESP - Spain	50,604,375	297,801	66,965.13
ROU - Romania	23,833,860	149,278	14,313.38
PRT - Portugal	9,187,803	103,462	96,625.44
FRA - France	54,951,621	61,911	9,825.69
ITA - Italy	30,075,506	51,056	53,961.38
HRV - Croatia	5,707,857	32,972	13,113.06
GRC - Greece	13,257,480	22,260	44,640.13
BGR - Bulgaria	11,158,767	12,702	9,733.94
HUN - Hungary	9,305,287	7,287	338.69
SVN - Slovenia	1,998,091	4,388	103.25

Wildfire activity, fuelled by the high temperatures, heatwaves and droughts across the continent, emitted over six megatonnes of carbon over the summer — the highest figures in 15 years.

High emissions were largely caused by the devastating forest fires across southwestern France, Spain and Portugal.

France has seen wildfires burn over 62,000 hectares while Spain has lost almost 300,000 hectares since the start of the year. Both EU member states are expected to record their highest wildfire emissions in the last 20 years.

Slovenia, Czechia, Hungary and Germany also experienced significant wildfire activity during the summer months.

In Germany, for example, wildfires burned 4,293 hectares — almost double the previous maximum.

Overall, forest fires have burned more than 508,260 hectares over the summer months (from June to September), compared to a 2006-2021 average of 215,548 hectares for the same period. For context, a typical football pitch covers about one hectare of land.

2.1.3 The cost of wildfires

The cost of a forest fire is not only direct, for instance, it does not include exclusively the buildings or lands that have been destroyed, but there are also indirect economic consequences. In the figure and the extended economic analysis below, obtained by EFFIS studies [14], we will see exactly how economic losses can occur due to a forest fire.

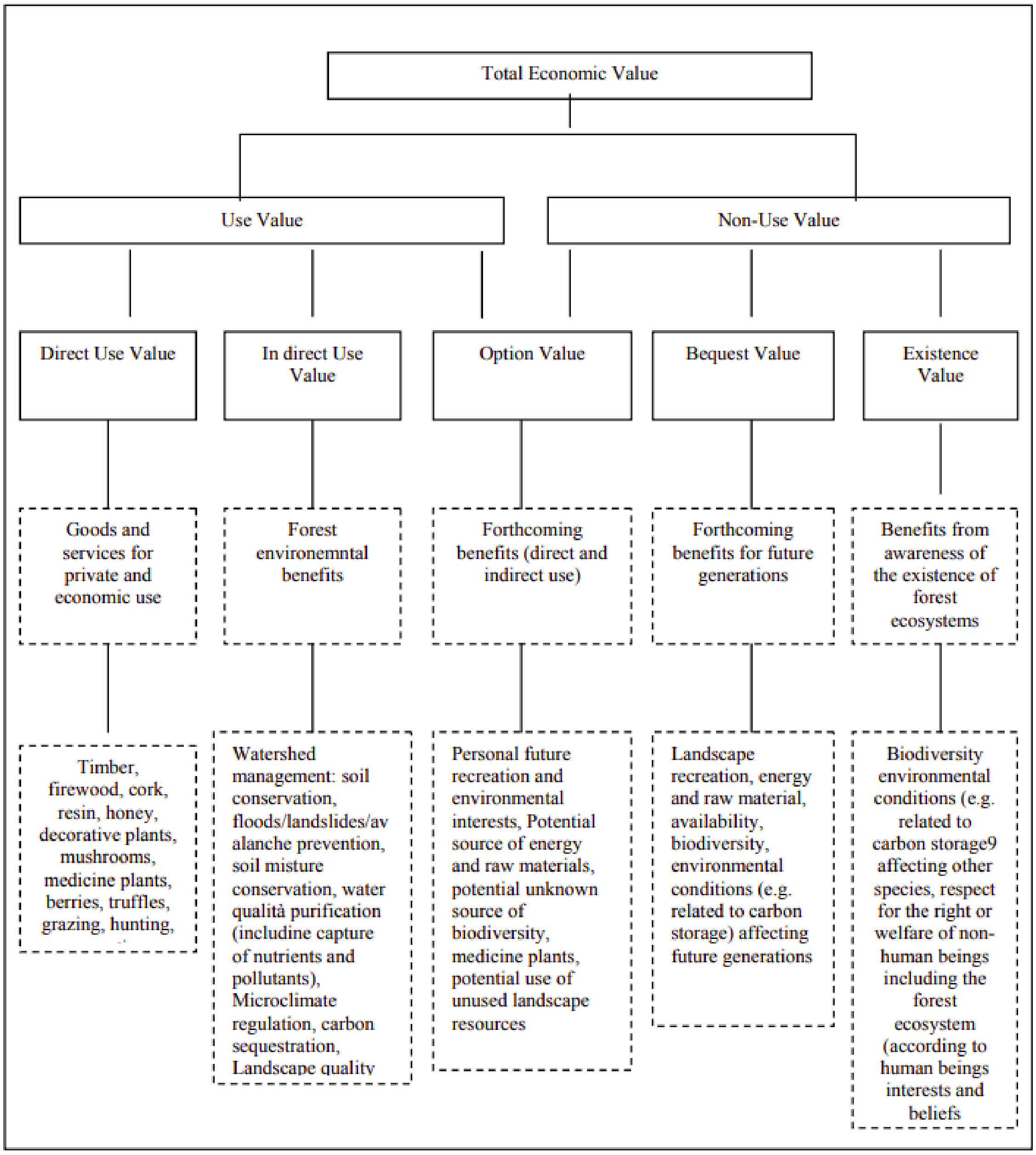

Total Economic Value
Use Value
Non-Use Value
Direct Use Value
In direct Use Value
Option Value
Bequest Value
Existence Value
Goods and services for private and economic use
Forest environemntal benefits
Forthcoming benefits (direct and indirect use)
Forthcoming benefits for future generations
Benefits from awareness of the existence of forest ecosystems
Timber, firewood, cork, resin, honey, decorative plants, mushrooms, medicine plants, berries, truffles, grazing, hunting,
Watershed management: soil conservation, floods/landslides/av alanche prevention, soil misture conservation, water qualità purification (includine capture of nutrients and pollutants), Microclimate regulation, carbon sequestration, Landscape quality
Personal future recreation and environmental interests, Potential source of energy and raw materials, potential unknown source of biodiversity, medicine plants, potential use of unused landscape resources
Landscape recreation, energy and raw material, availability, biodiversity, environmental conditions (e.g. related to carbon storage) affecting future generations
Biodiversity environmental conditions (e.g. related to carbon storage9 affecting other species, respect for the right or welfare of non-human beings including the forest ecosystem (according to human beings interests and beliefs

Figure 2.2: Total Economic Value Components In Forestry

The economic assessment of forest fires is part of the above described plan and requires an estimate of marketed value, costs, use of the different elements that are potentially involved. The damage from forest fires owes its main peculiarity to the nature of mixed good of forest from which derives the existence of a public and a private profile.

The damage estimation to private goods shows many references in the literature, although only Michieli (2002) have dealt in a specific way with damage from fires in wooded areas with a brief case unable to explain fully the complex issue of fires.

In relation to studies on damage to public goods few studies have been summarized in Marangon Gottardo (2001). A calamitous event, with human or natural origin, that affects the territory, can cause direct (or primary) damages, and/or indirect (or secondary) to collectivity.

In particular, for **direct damages** we mean the effects to those (company, consumers, peoples) that suffer an immediate loss of well-being due to the loss or the damaging of goods which are normally used for the productive function and settlements. Thus goods of law, market and private. In the case of forest fires the direct damages are the one which strike private goods and can lead to the loss of agricultural production and of lands productivity with time, to damages to the estate structures and to the decrease of the land value imputable to the different productive potentiality of the ground, the loss of wood and non wood products (mushrooms, truffles and others ground cover products).

The loss of usefulness due to the environmental damage, which happens with the loss or the damaging of the environmental goods falls down in this category, and then the reduction of the relative flows of public services normally produced. In this category service loss of hydrogeological protection, landscape loss, biodiversity damages, carbon dioxide emission are included. The **indirect damages** are still charged to the individuals who, (above mentioned), suffer a loss of benefit as consequence of a reduction or compromising of the functions which, normally, are produced by the damaged resources and that carries, a decrease of the collective well-being.

The **sociale damage** has to be then added to these two fundamental types. In fact, events as acute than diffuse, relative to the land fragility sharpen the one that sociologists call the perception of the environmental risk (Beck, 1986).

Definitively, the evaluation of the imputable damage to forest fires involves the analysis of market goods, prevalently private, and public goods, close to goods and service with environmental nature. While the first are easily referable to the traditional estimate, the second, of more recent interest, has to be examined with the methods of the environmental estimate. In addition to social and economic goods connected to the risk and its perception, have to be then considered.

The concept of TEV appears as a basis from which it is necessary to start, but that will turn out useful to get over. Such methodological difficulties, both in the typologies of considered goods, than in the estimation methodologies, arise from the agriculture analysis.

2.2 Detecting the Fire

Many proposals have been made in recent years in the field of wildfire detection.

Before technology was actively used in dealing with and identifying forest fires, as we will see below, the main way to identify forest fires was with the help of professional and volunteer personnel. The classic way of controlling forest areas was usually **human patrols**, either through the use of fire brigade or volunteer personnel. As we understand, this way of identifying fires requires the commitment of valuable human resources, as well as fuel for the means of transport used for them. Even the military has been used to aid this work.

With the development of high definition and thermal resolution cameras, this task has become easier. Through the installation of **composite camera systems** in a usually panoramic view of an area of interest, recognition could now be done without the commitment of professional personnel.

The job was made even easier when we introduced the use of **satellites** into our fire detection arsenal. We may not yet have effectively achieved the direct identification of forest fires using satellites, but it certainly helps significantly in recording and predicting the fire of a specific area.

In recent years, however, extensive research has been done on the part of prediction, with the help of **unmanned aerial vehicles (UAV)**. Drone patrols for the surveillance of forest areas have become particularly popular in recent times, as we can monitor a point or an area in a short time, and most importantly: Without the physical presence of people. However, an operator is required in most cases for this patrol to be effective. Few approaches have succeeded in automating drone patrols, i.e. without human supervision.

Below we will examine these 4 categories of supervision, and the differences between them.

2.2.1 Terrestrial Patrol

There is not much to say about the early identification of forest fires through forest patrols either on foot or using fire engines. Most countries now use, as we mentioned above, advanced technologies to patrol forest areas. However, there are still patrols by the competent fire and volunteering authorities.

For example, Throughout the fire season which started on March 1, 2022 in Greece, there was increased surveillance by the Fire Department as extensive additional checks - patrols were carried out to prevent the start and spread of fires. In addition, during the patrols, it was checked whether the mandatory fire protection measures were observed, mainly in outdoor work.

It is noted that in that year the ground surveillance increased significantly with the participation of the Armed Forces, especially in the most dangerous areas, while the patrols turned into purely military, liberating forces of the Fire Brigade and the Greek Police.

2.2.2 Cameras System

The use of cameras for the solution of rapid identification of forest fires and their immediate response, was one of the first practical applications of the technology used to assist in the work of forest firefighting.

Initially, before the use of thermal technology, conventional cameras were used for the remote identification of a fire. What was happening was the installation of a network of cameras in various parts of a forest area. This network was then connected to a central system of monitors, where the observer could at any time and moment control every point where the cameras had access.

One such system, which is implemented in several states of America, is **ALERTWildfire** [1], which is a consortium of three universities – The University of Nevada, Reno (UNR), University of California San Diego (UCSD), and the University of Oregon (UO) – providing access to state-of-the-art Pan- Tilt-Zoom (PTZ) fire cameras and associated tools to help firefighters and first responders: (1) discover/locate/confirm fire ignition, (2) quickly scale fire resources up or down appropriately, (3) monitor fire behavior through containment, (4) during firestorms, help evacuations through enhanced situational awareness, and (5) ensure contained fires are monitored appropriately.

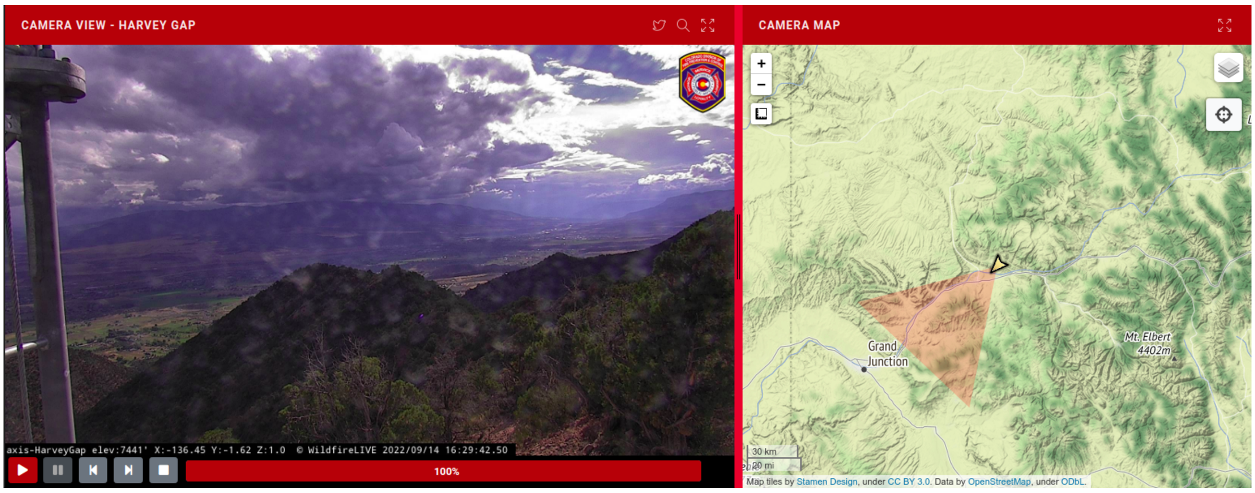

CAMERA VIEW - HARVEY GAP
CAMERA MAP
axis-HarveyGap elev:7441' X:-136.45 Y:-1.62 Z:1.0 © WildfireLIVE 2022/09/14 16:29:42.50
100%
Grand Junction
Mt. Elbert 4402m
30 km
20 mi
Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

Figure 2.3: ALERTWildfire Camera in the region of Colorado

ALERTWildfire is an expansion of the first network, ALERTTahoe, which was a pilot program deploying PTZ cameras and microwave networks in the region surrounding beautiful Lake Tahoe. This initial project was funded through the Nevada Seismological Laboratory (NSL) at UNR, the Tahoe Prosperity Center, the Eldorado National Forest, and the USFS Lake Tahoe Basin Management Unit.

Something that evolved this idea, is to place the camera at a panoramic point in the area of interest, so that there is with the least cost (number of cameras), the maximum range where an area can be observed (observation range).

However, placing these high resolution and wide ranging cameras in the panoramic areas of interest is not an easy process. An extensive study of the terrain morphology is needed to make the most of the range that a camera can cover. Early forest fire detection can effectively be achieved by systems of specialised tower-mounted cameras.

With the aim of maximising system visibility of smoke above a prescribed region, the process of selecting multiple tower sites from a large number of potential site locations is a complex combinatorial optimisation problem. Historically, these systems have been planned by foresters and locals with intimate knowledge of the terrain rather than by computational optimisation tools. When entering vast new territories, however, such knowledge and expertise may not be available to system planners. A tower site-selection optimisation framework which may be used in such circumstances are being proposed in this research [10]. Metaheuristics are used to determine candidate site layouts for an area in the Nelspruit region in South Africa currently monitored by the ForestWatch detection system. Visibility cover superior to that of the existing system in the region is achieved and are obtained in a number of days, while traditional approaches normally require months of speculation and planning.

Of course, the contribution of thermal cameras is also important, as there is also a high risk of forest fires breaking out during the night hours, especially by malicious arsonists, as detection is delayed due to the lack of light. For this reason, thermal cameras can identify the energy or otherwise the temperature of objects. All objects emit infrared energy, known as a heat signature. An infrared camera detects and measures the infrared energy of objects. The camera converts that infrared data into an electronic image that shows the apparent surface temperature of the object being measured.

2.2.3 Satellites

With the entry of satellites into the battle against forest fires, our task has become easier. The satellites we have at our disposal can do pixel-level recognition. As promising as tackling forest fires using satellite imagery and deep learning sounds, this technique has several **disadvantages**.

- The images provided by satellites can often be affected by many types of noise. For example, when the weather is cloudy, a wide range of forest areas can be overlapped by them, making it difficult to identify a fire. Also, naturally, fires create large smokes as time passes and burning progresses, thus making the task significantly more difficult. However, many improvements have been made recently in the field of recognition, through the help of machine learning, achieving very good results, using different satellites for different weather conditions [15].
- Most satellites, unfortunately, have a high latency in terms of updating consequently the data they provide us. Usually from a few hours to a few minutes. So there is not exactly real-time detection, which can be fatal as those few minutes that the satellite takes to take a picture, can be fatal for the progress of the forest fire.
- The spatial resolution is most often high (usually a few kilometers), thus making the identification of small forest fires difficult at times. This is something that is not a huge problem as for this type of forest fires there is usually not a very large financial cost but for example in an environmentally protected area such as the Natura areas, a forest fire in such a place can be very fatal.

Many new proposals for satellite systems that solve the 2 problems above have been made in recent years, for example geostationary satellites, i.e. satellites that move at the same speed as the earth, i.e. are in the same orbit, usually offer a real - time possibility to recognize forest fires, some for example are *GOES* (Geostationary Operational Environmental Satellites, *MSG* (Meteosat Second Generation) , *MTSAT* (Multifunctional Transport Satellite), and *COMS* (Communication, Ocean and Meteorological Satellite) , *MTSAT* (also known as Himawari-7) offered the highest temporal-resolution imagery and best data availability in Australia, before replaced by *Himawari-8*. Himawari-8 provides infrared imagery in Australia with 2 km spatial resolution every 10 minutes. This casts new light on fire management by providing real-time earth observations [18].

2.2.4 UAV's

The first use of UAVs was for military use, mainly for observation and detection of enemy targets. In recent years, the commercial use of drones has become very popular, with most uses being primarily for film shooting. Understandably, it didn't take long to start using them for civil protection purposes, such as assisting, searching and rescuing missing people.

A very important purpose of UAVs is as we will study below the observation and detection of forest fires. Initially, the observation was done with the help of experts who controlled the UAVs from the ground and watched for any suspicious activity. With the introduction of machine learning and deep learning, this task has become even easier.

As we will see, many proposals have been made in recent years on the part of early fire detection with the use of UAVs. In this paper [13], a method using 2 UAVs is used. The first one consists of a (VTOL) fixed-wing UAV, which flies at a height of 350m - 5500m. If this particular drone observes anything suspicious, in order to reduce the chance of a false positive error, a second drone is sent to the area where the suspicious activity was observed, flying at a height of 10m - 350m.

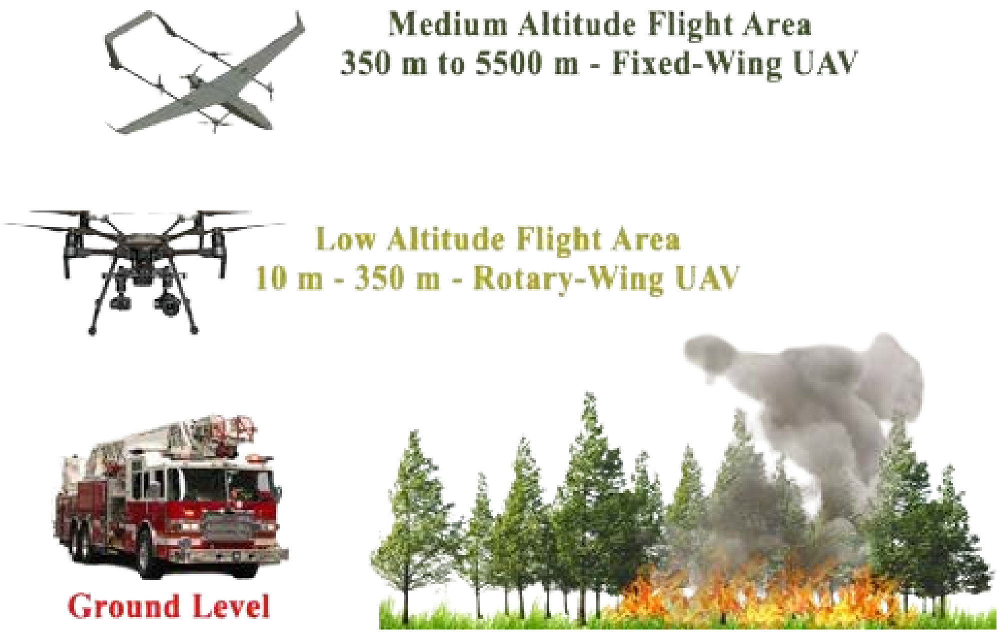

Medium Altitude Flight Area
350 m to 5500 m - Fixed-Wing UAV
Low Altitude Flight Area
10 m - 350 m - Rotary-Wing UAV
Ground Level

Figure 2.4: Fire Detection with the use of 2 UAV's

The specific system consists of a control system on the ground which controls the mobility of these 2 UAVs. The model used for recognition is the Tensorflow Object - Detection API. The input data consists of images that contain smoke as well as fire. The disadvantage of this particular system is initially that it is quite a costly undertaking but also that the monitoring is controlled by a human presence.

On the other hand, beyond machine learning techniques and methods, other methods and algorithms have been proposed for fire recognition at the object detection level. In this particular paper, they use techniques related to color and motion features. The movement is controlled through a sliding mode control (SMC) and a linear quadratic regulator (LQR).

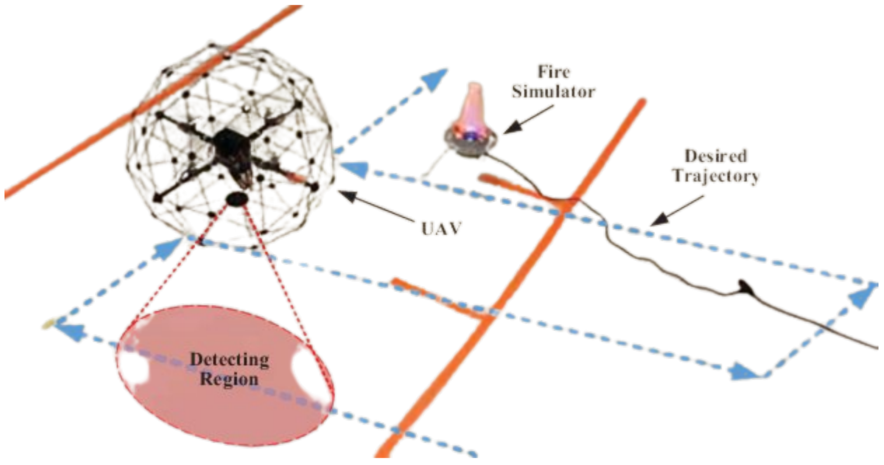


Figure 2.5: UAV Fire Detection using color and motion features

As we can see below, the advantage of this method concerns the recognition of fire, as it does not use machine learning techniques, but acquires the information about the presence of fire through other techniques, which do not need prior knowledge (input data) and thus the specific model has increased reliability and accuracy. The disadvantage, however, in this work has to do with the fact that the route is predetermined, which can be fatal in terms of recognition time, since the movement system is not changing and not dynamic.

2.3 Computer Vision

Computer vision is a concept that is extremely important and we will see it many times, especially in the 5th chapter of this thesis. The best we could do in order to recognize a fire in time, would be to have one or more observers who, through intelligent imaging systems, would constantly see images or live videos from different areas and immediately understand if there was a fire or not. This is exactly what we want to achieve here, with the difference that if we replace the human presence with intelligent machines they would be able to recognize the presence of fire just as well as humans. The branch of Computer Science that deals with pattern/object recognition and categorization is called Computer Vision.

2.3.1 What is Computer Vision?

Computer vision belongs to one of the research fields of machine learning and artificial intelligence. Through computer vision, we have the ability to obtain important information from images, videos and other visual inputs, thus being able to make decisions or make suggestions, based on the data we process. In short, artificial intelligence allows computers to think, and computer vision allows them to ”see”, process and perceive stimuli (mostly) from the outside world.

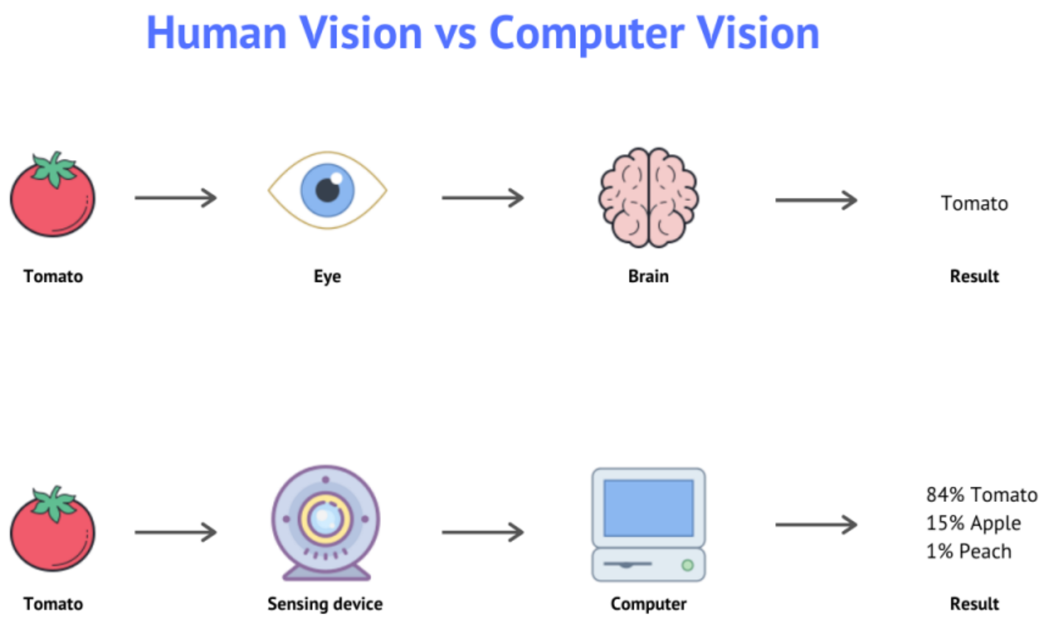


Figure 2.6: Computer Vision System

Computer vision needs a lot of data. It analyzes images and processes multiple data over and over again until it manages to find differences and patterns in the data it receives as input. For example, to train a computer to recognize images of a parrot, we need to ”feed” the model we are training with lots of images of parrots (of the same and different species) until it gets to a good point of learning the differences for example with other species of birds that may be present, thus managing to successfully identify the parrot.

But, how computer vision works? Let’s find why computer vision can be so accurate and what technology lies in the back of this field. The answer here is **Convolutional Neural Networks**. The most algorithms of computer vision, use a convolutional neural network, or CNN. A CNN is a neural network model which is frequently used in machine and deep learning in order to obtain features, like texture edges and shapes, from spatial data.

CNN don’t differ much than basic feedforward neural networks, since they learn from inputs, adjusting their parameters to make an accurate prediction or detection. What makes CNNs special is their ability to extract features from images.

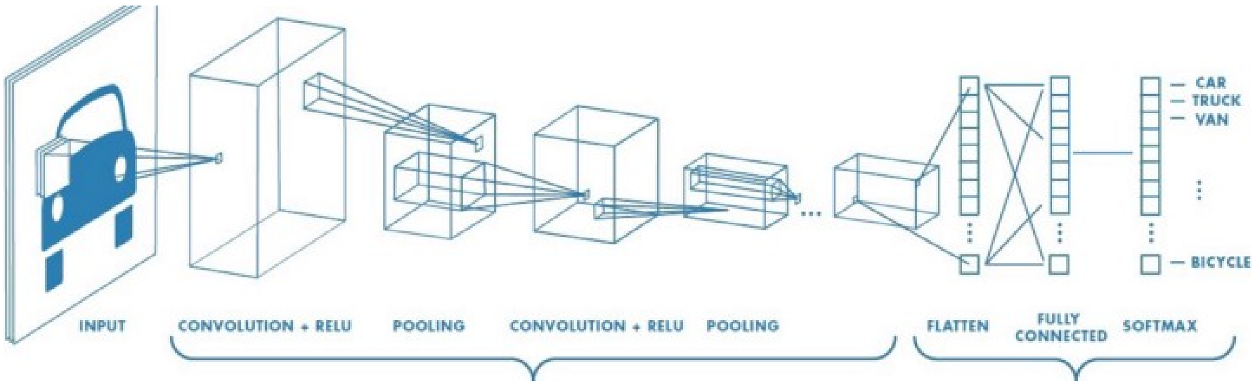


Figure 2.7: Basic CNN Architecture

CNNs are capable to process images like matrices, as they exist and extract spatial features from them, like texture edges and depth. They do this by using convolutional layers and pooling.

2.3.2 CV in Fire Detection

Most surveys dealing with fire detection focuses mainly on 2 major issues. In building a well-trained neural network and finding a sufficiently large and diverse dataset for better fire detection.

In recent years, the entry of artificial intelligence and computer vision have become very popular in the field of fire detection. Many researchers are trying through computer vision techniques to quickly and correctly manage fires, not only in forest areas. Many techniques have been proposed for this purpose, as we mentioned, some of them are fire motion detection, CNNs and background subtraction to make fire recognition as objective as possible regardless of the background, as for example in this paper [9] , which uses Gaussian blur, KNN-based background subtraction, Binarization and Closing.

Some others researches focus on others things more, like this one [16] which focuses more into Motion Detection, Fire Color Classification and Dynamic Texture Classification. Fire has distinctive features such as color, motion, shape, growth and smoke behavior.

3. SYSTEM OF USAGE

3.1 Technical Equipment

Below, we will analyze the technological equipment we needed to achieve this particular project.

3.1.1 DJI Ryze Tello

In recent years, many types of drones have been used for commercial as well as operational purposes. A type of drone that exists is for example, Multi-Rotor Drones, which offer great flexibility during flight. There are also fixed-wing drones which can cover longer distances but are also energy efficient as due to their construction, they only need energy to continue moving straight and not to be held in the air during flight.

We chose the DJI Ryze Tello for 2 reasons:

The first is that it belongs to the category of Multi-Motor drones which is ideal in our case in terms of the flexibility it offers us in detecting forest fires, as it can go directly from point A to point B.

The second reason is that this particular drone offers us the possibility to program it through a special SDK, with the Python programming language. Thus, we have the possibility in real time, to run the fire detection program during the flight, without having to transfer the video streaming to another screen and run our algorithms from there.

Below, we will break down the technical features of the **DJI Ryze Tello**.

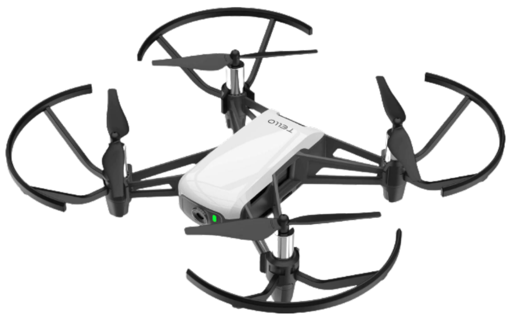


Figure 3.1: DJI Ryze Tello

Table 3.1: DJI Ryze Tello Characteristics

Aircraft	Flight Perfomance	Camera
Weight: 80g	Max Flight Distance: 100m	Photo: 5MP (2592 x 1936)
Dimensions: 98 x 92.5 x 41mm	Max Speed: 8m/s	FOV: 82.6°
Propeller: 3 inches	Max Flight Time: 13min	Video: HD720P30
Batteries: 3	Max Flight Height: 30m	Format: JPG(Photo), MP4(Video)

3.1.2 WiFi Repeater

This particular project, due to its nature, already creates a limitation. The transmission of the signal and the communication between the computer and the UAV. Due to the fact that our Drone patrols will be in forested areas, it is quite reasonable that there will be signal interference between the Drone communication.

Therefore, we had to find a way firstly to expand the range of movement of the Drone to be able to successfully patrol large areas and on the other hand to have the best possible communication, in order not to lose information and data during the surveillance .

We chose to use a WiFi Extender for this purpose. The machine we use is the **tp-link TL-WR802N**. The speed it offers us is 300Mpbs Wireless Speed. More specifically, this router connects to the DJI Ryze Tello network and extends the signal it provides, so that communication with the computer is faster and at a greater distance. We also must say at this point that the WiFi Extender needs energy. We can provide this energy by using a Powerbank (Portable Charger). The Powerbank device we choose is the **TP-Link TL-PB10400** which can provide us total energy of 10.400mAh. Below, we present the workflow of this particular device.

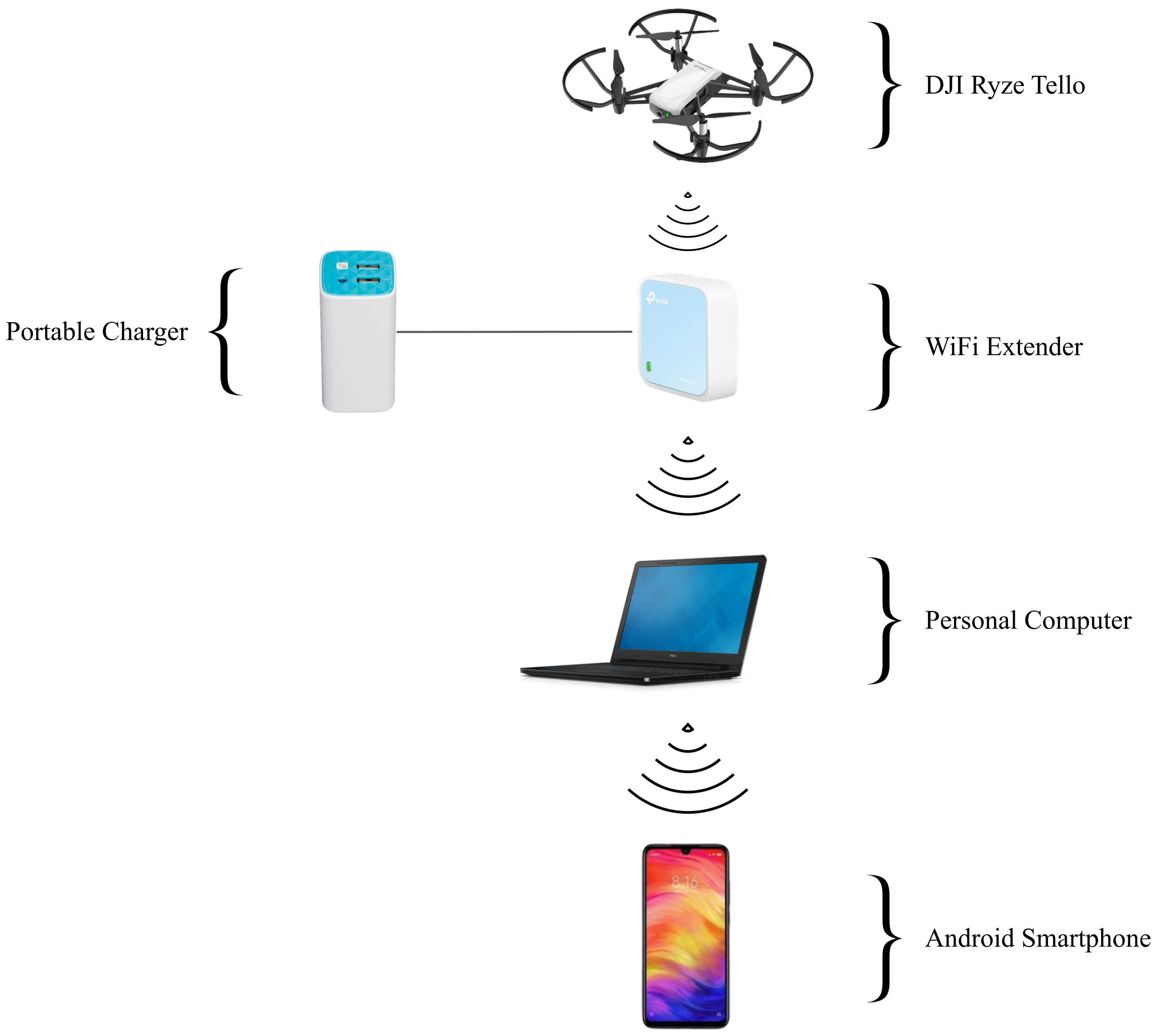


Figure 3.2: System Workflow

3.1.3 Personal Computer

The personal computer we used is **Dell Inspiron 3576**. The processing power we use is *Intel core i7-8550U CPU @ 1.80GHz*. The graphics card is *AMD Radeon 520 2GB)*. For the study of this particular thesis, we use the CPU of the computer, as we do not have the possibility to use the GPU or any other external one.

We have been able to achieve **5 FPS** so far when streaming video from the Drone to the PC. Of course, we could achieve better results by using a graphics card to process the image provided by the UAV.

The reason we can't use the computer's existing GPU is because of its architecture. A computer using some NVIDIA graphics card and using CUDA software could turn out to be very profitable. For example, the interface provided by torch.cuda takes advantage of the use of the graphics card, as long as that type is supported by our computer's architecture. Therefore, in this particular research, we will be satisfied with the processing power provided by the central processing unit of the computer.

4. SPATIAL ANALYSIS

4.1 Selecting the area of interest

There were many areas of interest that we could choose from. More generally, there are many reasons and motivations for someone to choose an area where they believe they need constant or partial surveillance for a forest fire outbreak. Such areas for example are a Natura area. Natura areas, especially in Europe, are very important as they constitute the biological as well as the cultural of a region or country. The well-known Natura 2000 network, which was created for the protection and observation of such areas within the framework of the European Union. Here is a list of these areas in Greece: Natura 200 in Greece.

Another reason one might wish to observe a forested area is for example a farming area or a large-scale farm. Many of the natural disasters (and especially that of forest fires), in recent years have particularly affected many private forest lands, thus costing a lot of money both to professionals in the primary sector and to public entities such as a municipality for example.

The research area we are considering is located southeast of Chalkida, the capital of Evia. The specific area is called Pezonisi. It is also known as the Island of Dreams. We chose this particular island for 2 reasons.

The first is that it is ideal from a geographical point of view, as it consists of a rich forest with a large presence of plantation. Also, as we will see below and in the analysis we have done, there are no large altitude differences on the specific island, thus making our Drone search ideal, as it is understandable, if there are high altitudes at the point we are studying, it would made our task much more easy, as we would simply place the Drone on the highest areas of the island and thus have a panoramic observation view of the area, without having to ”scan” the area to detect any suspicious activity.

The second reason concerns the cultural and economic heritage of the specific island in the wider region. This particular island has been a pole of attraction for many entrepreneurs, as it was for many years an ideal area for the establishment of hotel units, which automatically implies the economic development of the area.

Now, the island has become an ideal place for the locals of the island as it is a place for people to enjoy the beaches as well as the natural beauty that it offers. Furthermore, various cultural events and concerts take place in this part of the island. We must note here, that this particular island also functions as a port for the boats used by the local professional fishermen of the area.

So we understand that the importance of this particular island is considered the highest from an economic as well as a cultural point of view. The destruction of this particular island due to a forest fire would be a huge economic blow to the wider region, both to the professionals who work there.

The coordinates of the specific place are: (*Latitude , Longitude*) : 38.384764368161605, 23.800158067741208

Below we will see a satellite image of the island.

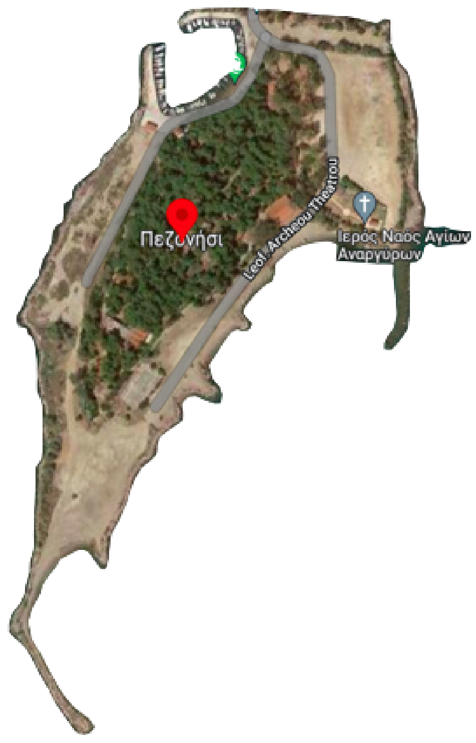


Figure 4.1: Satellite image from Pezonhsi

4.1.1 Why is spatial analysis important?

The importance of spatial analysis in the particular research we are doing is of the utmost importance, for many reasons.

- **Study of the extent and limits of the area**: Initially, the spatial extent of the area of interest and especially the extent and spatial limits of the specific area is a major issue. It is very important to know how many square-metre (m^2) the area consists of. First, to calculate the total area and thus properly plan the search that the drone will perform.

  In addition, we need to know the spatial boundaries of the area in order to plan the flight of the drone so that it does not exceed or overcome these specific geographical points. Finally, it is very important to know the above data in order to know where to place the base of the drone, i.e. the starting area (center of the circle) but also the surveillance area (radius of the circle).

- **Elevation**: Studying the elevation of the area under consideration is also very important. The reason is, as we mentioned above, to know the geographical heights of the area we are considering. The reason this happens is that if there are maximum heights in space, the observation process changes completely, as all we have to do in the case of large fluctuations in elevation is to find the points where these maximum heights occur , and to mainly place our drone in these areas, thus having a panoramic view of the area, and thus, if anything is suspicious is detected it will be dealt immediately.

In this particular thesis, we have chosen to solve the problem of early fire detection in forest areas where there is kind of similar elevation and the rate of change does not exceed approximately ±10 meters.

- **Atmospheric and weather phenomena**: One of the most important, if not the most important point in our research, is the study of the atmospheric data prevailing in the specific area. It is very important to know the temperature, the humidity, the presence or lack of clouds in order to know the possibility of rain and finally, most important of all, the speed and direction of the wind that exists in the area. The reason is, initially, to know an initial percentage of fire occurrence probability, which is calculated from the linear combination of all the above data. For example, if there is high humidity or the possibility of rain, we understand that the probability of a fire breaking out is low. On the other hand, the high temperature and drought in this area increases the risk of fire.

  The most important source of information that we can derive, however, not so much for the possibility of the fire breaking out, but mainly for predicting its course, is none other than the behavior of the wind in this area. The reason is that the fire is significantly affected by the speed and direction of the wind, as the direction it tends to take is parallel to the direction of the wind, thus affecting the model we have created for the drone's mission path.

4.1.2 GIS (Geographic Information System)

A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. By relating seemingly unrelated data, GIS can help individuals and organizations better understand spatial patterns and relationships.

GIS technology is a crucial part of spatial data infrastructure, which the White House defines as "the technology, policies, standards, human resources, and related activities necessary to acquire, process, distribute, use, maintain, and preserve spatial data."

GIS can use any information that includes location. The location can be expressed in many different ways, such as latitude and longitude, address, or ZIP code.

Many different types of information can be compared and contrasted using GIS. The system can include data about people, such as population, income, or education level. It can include information about the landscape, such as the location of streams, different kinds of vegetation, and different kinds of soil. It can include information about the sites of factories, farms, and schools, or storm drains, roads, and electric power lines.

With GIS technology, people can compare the locations of different things in order to discover how they relate to each other. For example, using GIS, a single map could include sites that produce pollution, such as factories, and sites that are sensitive to pollution, such as wetlands and rivers. Such a map would help people determine where water supplies are most at risk. [4]

In this thesis, we will mainly use environmental data, such as the extent of the area and the elevation. The GIS tool that we will use next is Google Earth Engine, which as we will see, provides us with various and different data that can be useful for our research. Below in the figure 4.2, we can see graphics of how a GIS software system works.

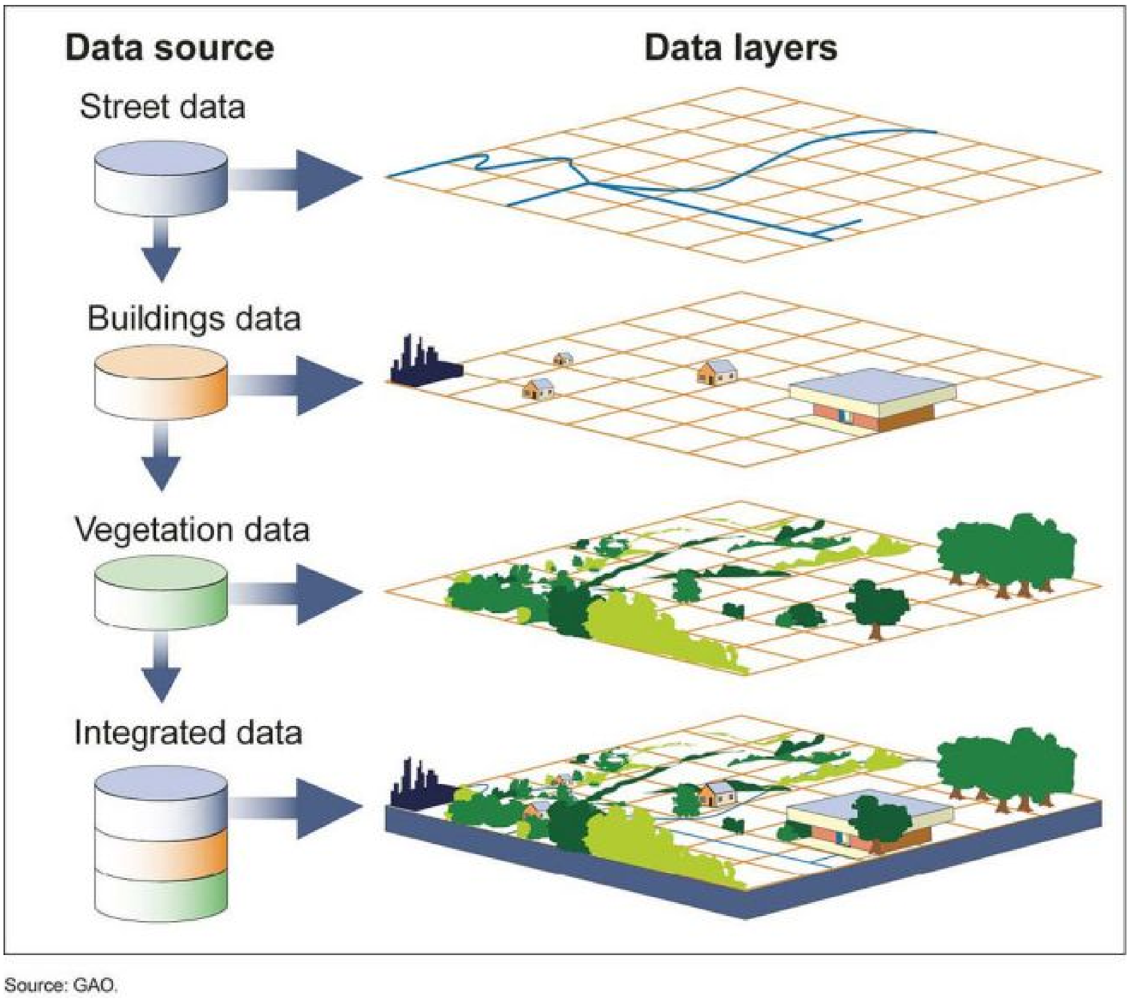


Figure 4.2: Typical GIS System

4.1.3 Google Earth Engine Code

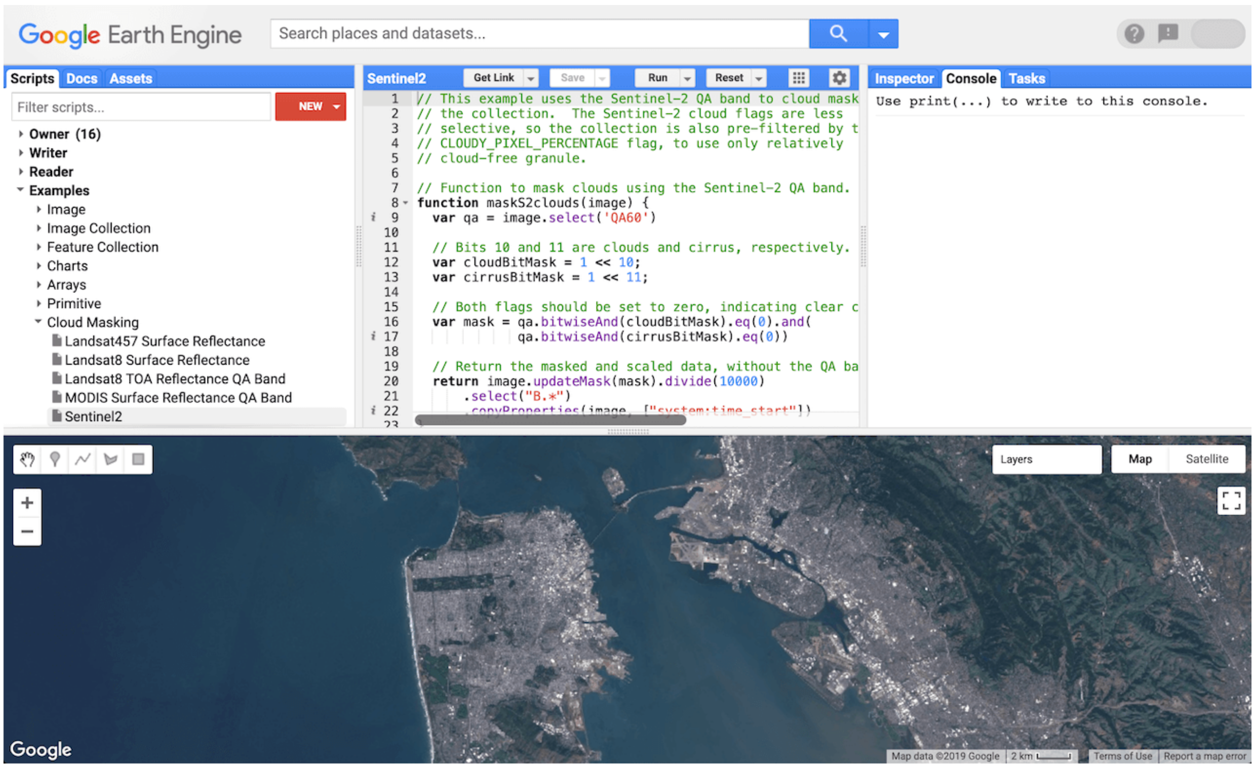
Google Earth Engine is a computing platform that allows users to run geospatial analysis on Google's infrastructure. There are several ways to interact with the platform.

The Code Editor is a web-based IDE for writing and running scripts. The Explorer is a lightweight web app for exploring our data catalog and running simple analyses. The client libraries provide Python and JavaScript wrappers around our web API.

The Code Editor is an interactive environment for developing Earth Engine applications (Figure 4.3). The center panel provides a JavaScript code editor. Above the editor are buttons to save the current script, run it, and clear the map.

The Get Link button generates a unique URL for the script in the address bar. The map in the bottom panel contains the layers added by the script. At the top is a search box for datasets and places.

The left panel contains code examples, your saved scripts, a searchable API reference and an asset manager for private data. The right panel has an inspector for querying the map, an output console, and a manager for long-running tasks. The help button in the upper right contains links to this Guide and other resources for getting help.



```
// This example uses the Sentinel-2 QA band to cloud mask
// the collection.  The Sentinel-2 cloud flags are less
// selective, so the collection is also pre-filtered by t
// CLOUDY_PIXEL_PERCENTAGE flag, to use only relatively
// cloud-free granule.

// Function to mask clouds using the Sentinel-2 QA band.
function maskS2clouds(image) {
  var qa = image.select('QA60')

  // Bits 10 and 11 are clouds and cirrus, respectively.
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

  // Both flags should be set to zero, indicating clear c
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0).and(
             qa.bitwiseAnd(cirrusBitMask).eq(0))

  // Return the masked and scaled data, without the QA ba
  return image.updateMask(mask).divide(10000)
      .select("B.*")
      .copyProperties(image, ["system:time_start"])
```

```
Use print(...) to write to this console.
```

Figure 4.3: Google Earth Engine Code Enviroment

The work we've done on Google Earth Engine Code is the first of 2 parts of Spatial Analysis we've done. This particular tool initially helped us to analyze our geographic area of interest. In more detail, the work we performed here is explained into 2 scripts:

- **poly1**: In this script, we originally loaded *SRTM Digital Elevation Data Version 4* : The Shuttle Radar Topography Mission (SRTM) digital elevation dataset was originally produced to provide consistent, high-quality elevation data at near global scope. This version of the SRTM digital elevation data has been processed to fill data voids, and to facilitate its ease of use.

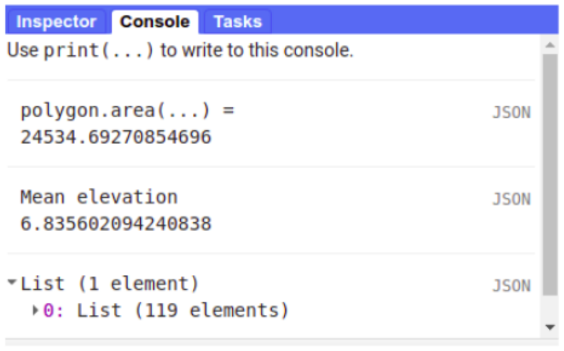
  More specifically, we did a survey about the area, as for example we marked only the area where the forest area exists, thus making a polygon of interest (which is marked with black color). We've trained to be as accurate as possible with the marking process. The polygon which is marked, is displayed at the console window. We have 119 points (elements).

  The area of the region specifically amounts to **24534.69270854696 square meters**.

  The mean elevation of the area is **6.835602094240838 meters**. In addition, from the specific script we extract the list with the coordinates of the polygon, i.e. the geographical points which, if we join them, will give us the expensive formation and the location of the island in space.

  Finally, we have put in an experimental position the starting point of the drone and a sample initial range: **50m**, to see the space occupied by the patrol that our drone will make (represented by the red circle inside our polygon).

  Below it's a figure with this polygon and also the console window displaying the info we've obtained.

Marked Polygon	Console Window
	
Script: **poly1**	

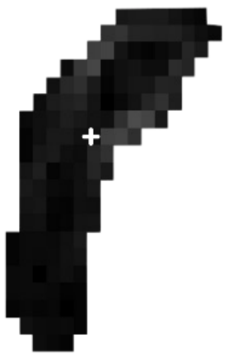
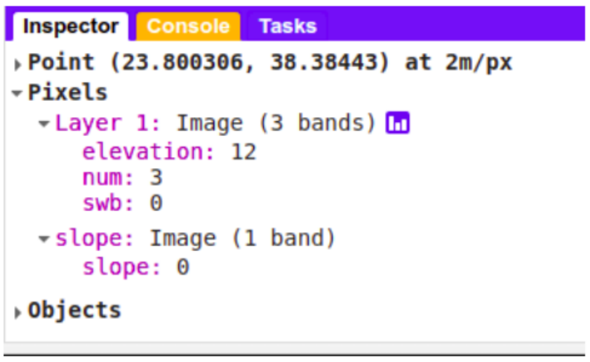
```
Use print(...) to write to this console.

polygon.area(...) =                              JSON
24534.69270854696

Mean elevation                                   JSON
6.835602094240838

▾List (1 element)                                JSON
  ▸0: List (119 elements)
```

Figure 4.4: poly1 script

- **elev1:** In this script, the data we used is the NASADEM: NASA NASADEM Digital Elevation 30m. NASADEM is a reprocessing of STRM data, with improved accuracy by incorporating auxiliary data from ASTER GDEM, ICESat GLAS, and PRISM datasets.

  The most significant processing improvements involve void reduction through improved phase unwrapping and using ICESat GLAS data for control. More specifically, in this particular script, the information we extract is the elevation that exists in the wider area. Through the inspector, we can see in detail the elevation and also the slope. Also, in this area we can see the slope of a point, using the inspector. To better understand the concept of slope, the specific example is given: A rise of 100 feet over a run of 100 feet yields a 100 percent slope. A 50-foot rise over a 100-foot run yields a 50 percent slope. The precision we use to analyze the elevation as well as the slope in the area is 2m/px. The purpose of this script is to verify and check that the elevation remains almost the same, with a relative deviation as we have mentioned, of 10m.

  Thus, we confirm that the study area is a ”relatively” flat area without large altitudinal differences.

Masked Island	Console Window
	
Script: **elev1**	

```
▸Point (23.800306, 38.38443) at 2m/px
▾Pixels
  ▾Layer 1: Image (3 bands)
     elevation: 12
     num: 3
     swb: 0
  ▾slope: Image (1 band)
     slope: 0
▸Objects
```

Figure 4.5: elev1 script

4.1.4 Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more.

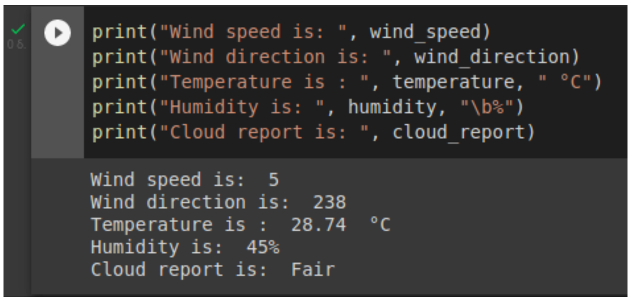
We can use Google Colabs like Jupyter notebooks. They are really convenient because Google Colab hosts them, so we don't use any of our own computer resources to run the notebook. We can also share these notebooks so other people can easily run our code, all with a standard environment since it's not dependent on our own local machines. However, we might need to install some libraries in our environment during initialization.

Figure 4.6: Google Colab

This particular tool is very important for the achievement of our project. Below we will analyze the data processing we do through Google Colab.

1) **Weather Info**: It is very important in the detection of a forest fire, to know the atmospheric and weather data. The data we extract are:

   - The temperature of the area.
   - The chance of rain.
   - Humidity.
   - The direction and speed of the wind in the specific area.
   - Cloud report



```
print("Wind speed is: ", wind_speed)
print("Wind direction is: ", wind_direction)
print("Temperature is : ", temperature, " °C")
print("Humidity is: ", humidity, "\b%")
print("Cloud report is: ", cloud_report)
```

```
Wind speed is:  5
Wind direction is:  238
Temperature is :  28.74  °C
Humidity is:  45%
Cloud report is:  Fair
```

Figure 4.7: Weather Data

The most important element that interests us and for the continuation, is that of the wind.

We extract the data through a get.request with a url containing the location of the specific area (longitude and latitude).

The website we used for data extracting is Windfinder. Wind direction is in **degrees** and wind speed is in **kts** (1 kts = 0.514444444 m/s).

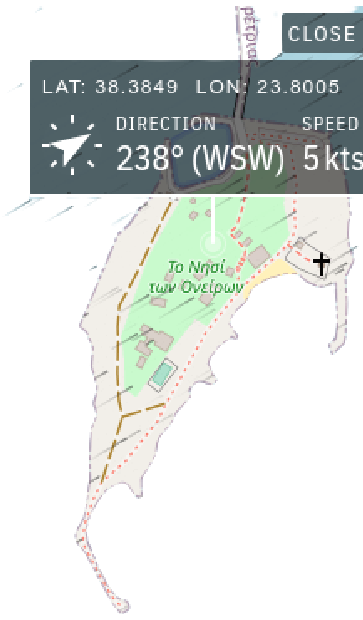


Figure 4.8: Windfinder Interface

2) **Polygon creation command (GeoGebra)**: In order to find the optimal positions of the drone in the polygon area, we need to do some work on GeoGebra (as we'll see below). To be more specific, we need to create the polygon that will represent the island. For this purpose, we fed Google Colab with the list of coordinates that we created in the previous tool (Google Earth Engine Code) and thus, we created the command that we'll parse on GeoGebra afterwards.

   The command will look something like this:

   Polygon((23.800966156784177,38.385898843453276), ... ,
   (23.800982511895043,38.385934219438305),
   (23.800966156784177,38.385898843453276))

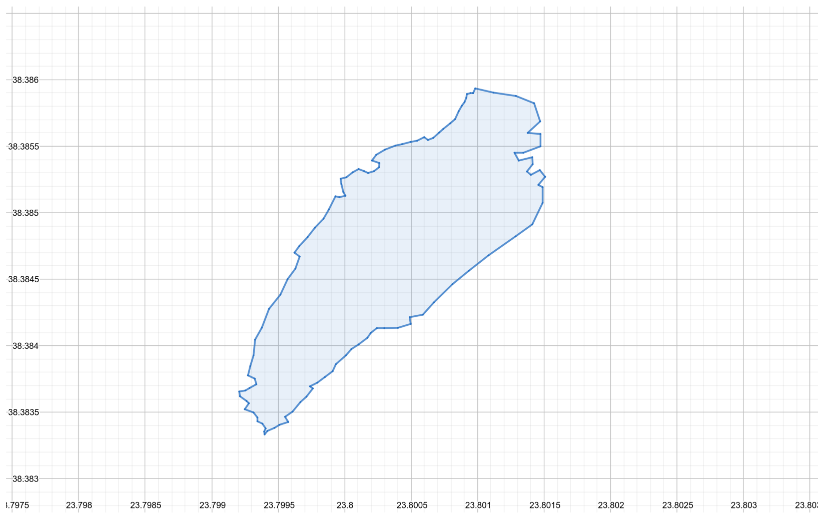


Figure 4.9: GeoGebra Polygon

3) **Line equation based on wind angle**: In order to compute the optimal position for the drone, we must draw *n* lines in the direction of the wind.

   Wind direction (express in deegres), will give us the wind direction. Based on this, we'll compute the line equation, which will be something like:

   $y = a * x + b$.

   But, because we need the angle of the line with y-axis, the equation will be like: $a * y = x + b$

   Since, we don't care now for the line position in x-axis, b will be represent a series of b's => parallel lines.

   So: **b** = $\{b_1, b_2, ..., b_n\}$. b now, will represent a vector of numbers.

   The main problem here is to find the a factor. The following computation is needed to find a:

$$a = \tan\left(radians(wind\_direction)\right)$$

4) **Creating b vector**: The **b** vector is one of the most important parts here. What we need to do here, is to create n lines. In order to do that, we need a **b** vector, that represents parallel lines in aligment with the starting line, $n/2$ to the left and $(n/2) - 1$ to the right (The remaining line is the central one). The centre on x-axis of the polygon, after computing it based on graph, is: $x_{centre} = 23.8022$, and the centre on y-axis of the polygon, is $y_{centre} = 38.3849$.

   So, first of all, in order to create n parallel lines align with the line that passes through the centre of the polygon, we need to find the central line. Below we'll explain this process:

   So, the main concept here is to find the initial b number that will go through the centre of the polygon. In order to find this number we followed this process:

   We have 2 crucial points: $point_a = (x_{centre}, y_{centre})$

   And $point_b = (x_b = (y_{centre} * a), y_{centre})$

   Then, we computed the Euclidean distance between these 2 points. The result of this computation gives us the b number. Graphically is something like this:

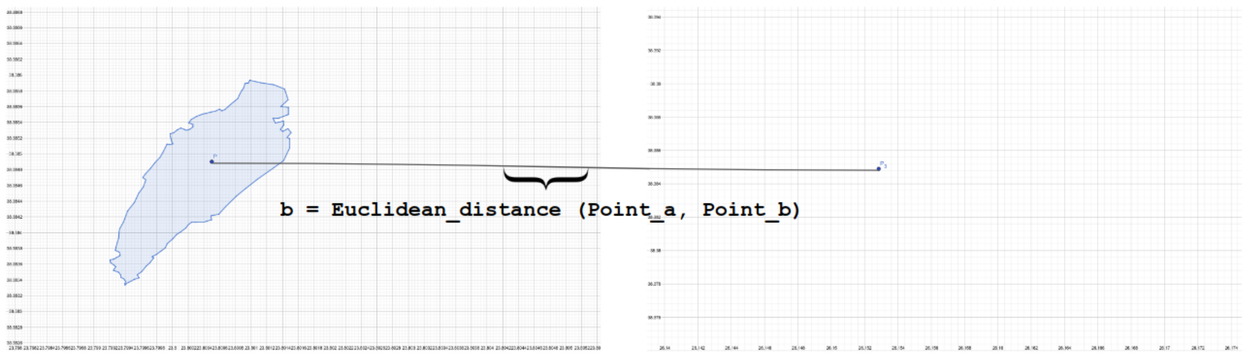


Figure 4.10: Euclidean Distance Computation

Since we computed the initial b number, it's an easy task to compute the b vector. For example, if: $b = 21.944842476369494$, we create $n/2 - 1$ 'right' and $n/2$ 'left' lines, where we decided that $n = 20$. The difference between these lines are by 4 decimals, for example the first right and the first left line in our example are: $\{21.9449, 21.9447\}$. Ofcourse, n it's up to us and it can be changed. We decided that this value is optimal for the area that we have to patrol.

In this moment, we must explain 2 things:

What we mean by saying 'right' and 'left' lines? 'right', in our case, means that we create $n/2 - 1$ lines to the right of the starting line, always relative to the direction of the wind. This can intuitively confuse us, as if the wind direction is < 90° or > 270°, graphically this will have the opposite effect, as relative to how the analyst sees the image, the right lines are essentially the left lines. This needs to be explained as we will mark the right lines in green and the left lines in red, as we will see in the next subsection.

Why is there a need to create and analyze these n lines? So, the main idea is the follow one:

Each line, including the central (initial) line, has the following role. First, they represent the imaginary lines created by the wind in the area. If we consider that the fire tends to follow the direction of the wind, we come to the following question: Which straight lines are considered more dangerous? To define risky, we make the following assumption: Risk in our case, is that the fire will burn a part of the island. So, the most 'dangerous' straight line is considered to be the one that occupies most of the island, as the fire will tend to follow a similar path to the direction of the straight line.

5) **Finding the n_b(number of Batteries) best lines**:

How can we calculate which straight lines occupy or better, cross most of the island? The answer is simple. We calculate the euclidean distance of the points of intersection of the straight line with the outline of the island. In our case, the outline of the island is nothing more than the straight lines that form these 119 points if you connect them.

As we mentioned before, our drone has 3 batteries. So when the battery starts to weakens, our UAV begins it's route to the base. In order to achieve a better patrol, we choose to scan the largest possible part of the island, in the most dangerous places. For this reason, we do a sorting of the individual distances covered by our imaginary straight lines, and from them we end up with 3, where 3 is also the number of batteries. These lines are displayed by purple color.

6) **Design of the area (Circle) that our drone will make it's flight**:

The general idea here is to place the starting point, i.e. the point from where the drone will start its patrol, on the current most dangerous line we have chosen. We find the center of the circle in the following way:

First, we choose a number for the radius of the circle, i.e. the maximum range that our drone can cover. Let's assume for a start, 50 meters. After our computations, we've found that:

1m = 0.00000954495 mm, in GeoGebra. So, in order to solve our equations, we'll take that as a fact and we'll proceed to compute the equation of the Circle that we want.

Then, we find the point of the line, on which, the first point of intersection of the line with the contour of the polygon with the aforementioned point, are 50 meters apart. Thus, we find the center of the circle and thus, the starting point.

4.1.5 GeoGebra

GeoGebra is a dynamic mathematics software for all levels of education that brings together geometry, algebra, spreadsheets, graphing, statistics and calculus in one engine. Below, we'll analyze and explain how the data we calculated in Google Colab, are displayed in GeoGebra Enviroment.

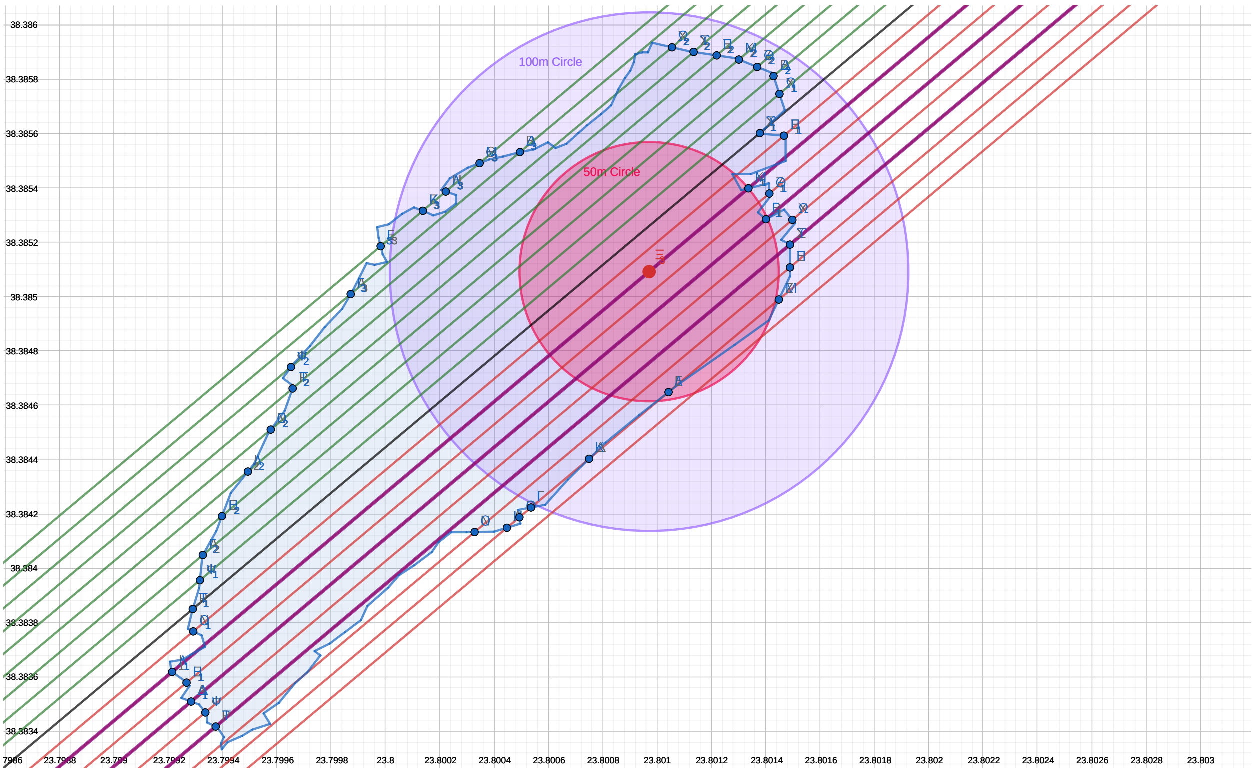


Figure 4.11: GeoGebra Final Model

First, the wind direction in this experiment was 50°.

- **Polygon**: The blue-marked shape is the polygon, that we created from Google Colab as we mentioned before.
- **Black line**: It is the central line, i.e. the line where it passes through the center of the polygon and based on this we will calculate the remaining $n - 1$ lines.
- **Green lines**: The green lines represent these lines of the **b** vector where it is to the right of the central black line. In this figure, because the wind direction is >90°, they can be seen to the left of the black line, but they are in relative position from the right based on the direction of the black line (since the wind is passing from the north to the south in this example).
- **Red lines**: The red lines represent the lines of the b vector where it is to the left of the central black line. In this particular figure, as we mentioned for the green lines, because the wind direction is >90°, they can be seen to the right of the black line, but they are in a relative position from the left based on the direction of the black line (since, the wind passes from north to south in this example).
- **Purple lines**: The purple lines represent nb, where number of batteries = 3, more dangerous lines. The degree of danger starts from left to right, as we see in the

figure. Of course, this is not always binding, as these lines may be differently placed in space and not linear, as they are in this particular case.

- **Blue points**: The blue points represent the points of intersection of the lines with the outline of the island, where as we mentioned above, as outline we define the lines that are created if we join the points that define our polygon.
- **Red point**: The red point, as is understandable, is the center of the red circle, where it also represents the starting point from where our drone will start.
- **Red circle**: The red circle represents the range our drone has, assuming the starting center is the red dot. Of course, in reality, while here the patrol area appears to be a circle, as we analyze the polygon in 2D space, it is actually a hemisphere.

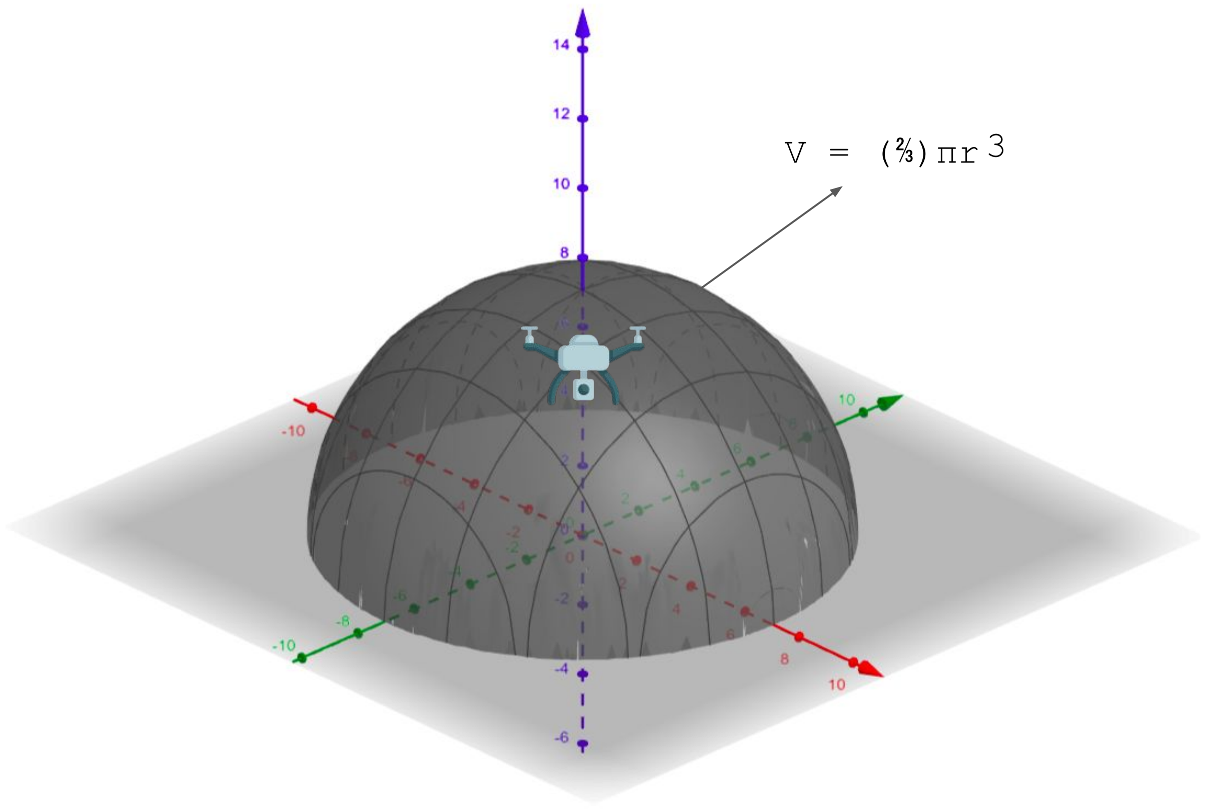


Figure 4.12: UAV’s Hemisphere patrol area

As we see in the figure (4.12), the total area that our drone can cover is up to

$$V = (2/3)\pi r^3$$

- **Purple circle**: The purple circle represents a longer detection range circle of the drone, in case we can achieve more communication between the computer and the WiFi extender.

4.1.6 Cost Function

So since we've figured out where's the starting point of our drone, the only thing remaining to do is to configure how the UAV will travel accross the selected patrol area. So, our drone will be driven by the user's pc and navigate through his control.

Nevertheless, we'll also create a cost function that will evaluate in real-time if our decisions are right or not. We also must considerate our limits of the polygon. Our drone must fly within the limits and to not fly far away from there. The cost function is as follows:

$$C(x, y) = (w * d(x, y) + t(x, y)) * a(x, y)$$

1) **Weight parameter**: $w = \mathbb{R}^*$

   This parameter will determine how "important" is the $d(x, y)$ factor. We've decided that this factor won't be fixed but will be affected by the speed of wind. A first approach is to define this factor equal to the speed of wind, or a linear combination of it.

2) **Distance parameter**: $d(x, y) = d(p(x, y), set(inc_p))$

   Where $p(x, y)$ is the current position of our drone and $set(inc_p)$ is the first 'set' of the intersection points of the lines that are in the same direction of the wind, and the intersect our polygon. The reason that this distance interests us is because the cost of catching a fire close to intersection points is bigger than to be presented inside the polygon.

3) **Time parameter**: $t(d(x, y) > dis('danger'))$

   In particular, the time parameter counts the seconds, when the value of : d(x,y) is bigger than the 'distance of danger', which it means, the maximum distance where the drone is too far away from the intersection points. This parameter help us 'contain' our drone close by the dangerous areas, with an option of freedom to explore different areas of interests near by.

   This parameter may change, because of the speed of wind. If the speed of wind is 'high' enough, when high will be measured by kts (which is the measurement unit of the speed of wind), the distance of range will be smaller.

4) **Activation Function**:

$$a(x, y) = \begin{cases} 1 & Polygon.contains(x, y) \\ \infty & !Polygon.contains(x, y) \end{cases}$$

   This function ensures that the drone will always move within the bounds of the polygon and will never escape from it.

   To make our UAV navigation completely autonomous, we need to find a way to approach that moves inside the polygon. Some initial thoughts are that the drone will follow a meandering navigation.

5. TRAINING CUSTOM YOLOV5 MODEL

In the following chapter we will analyze the way and the process by which we trained the object recognition model (Yolov5) with a custom fire dataset which consists of fire images.

5.0.1 What is Yolov5?

First of all, YOLO an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself.

YOLO is one of the most famous object detection algorithms due to its speed and accuracy.

Yolov5 [7] is the latest release version of ultralytics using the Pytorch framework.

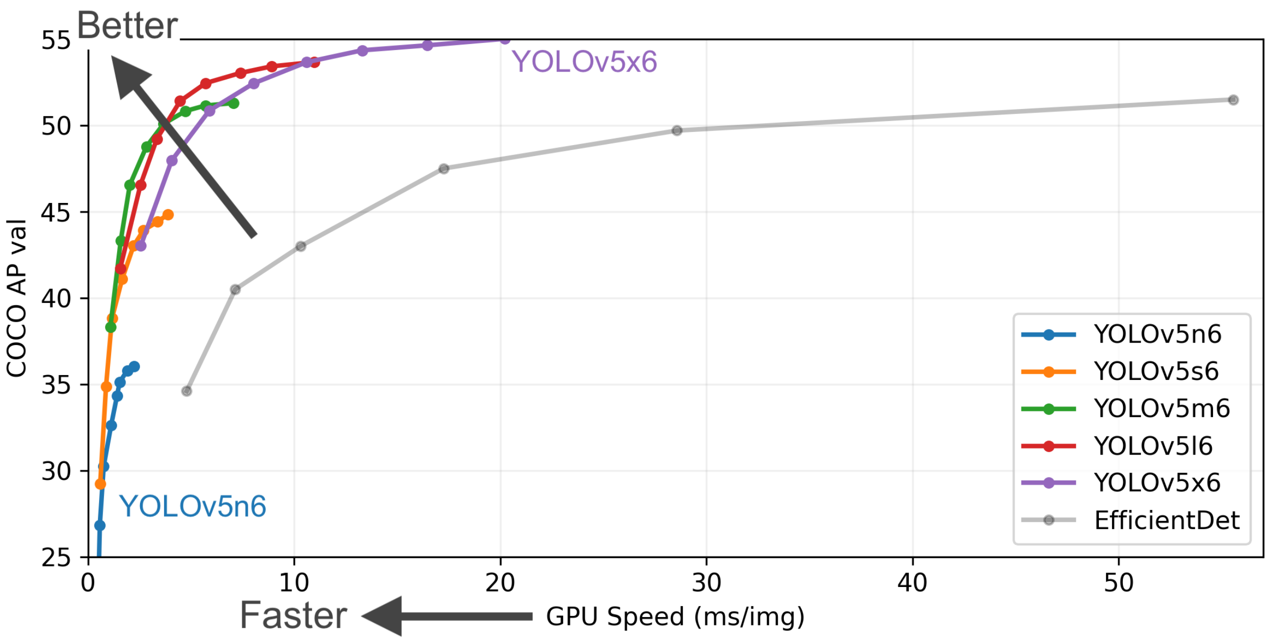


Figure 5.1: Yolov5 Framework Evaluation

But how does Yolov5 work? And what makes it so good and popular compared to other object recognition models? Below we will analyze how Yolov5 works in depth.

Yolo is state of the art of the object detection algorithm and it is so fast that it has become almost a standard way of detecting objects in the field of computer vision. Previously people were using *sliding window object detection*, then more faster version were invented, such as *R CNN*, *fast R CNN*, *faster R CNN*. But in 2015 YOLO was invented which outperformed all the previous object detection algorithms.

First, the task that we want to solve using this particular framework is that of **Image Classification + Object Localization**.

This means that in addition to classifying an image (it is possible to have more than 1 class, but in our case we only have 1, that of fire), we must do localization, that is, within the framework of the image, recognize a or many different objects(that may belong to the same class or not), and frame them in a box.

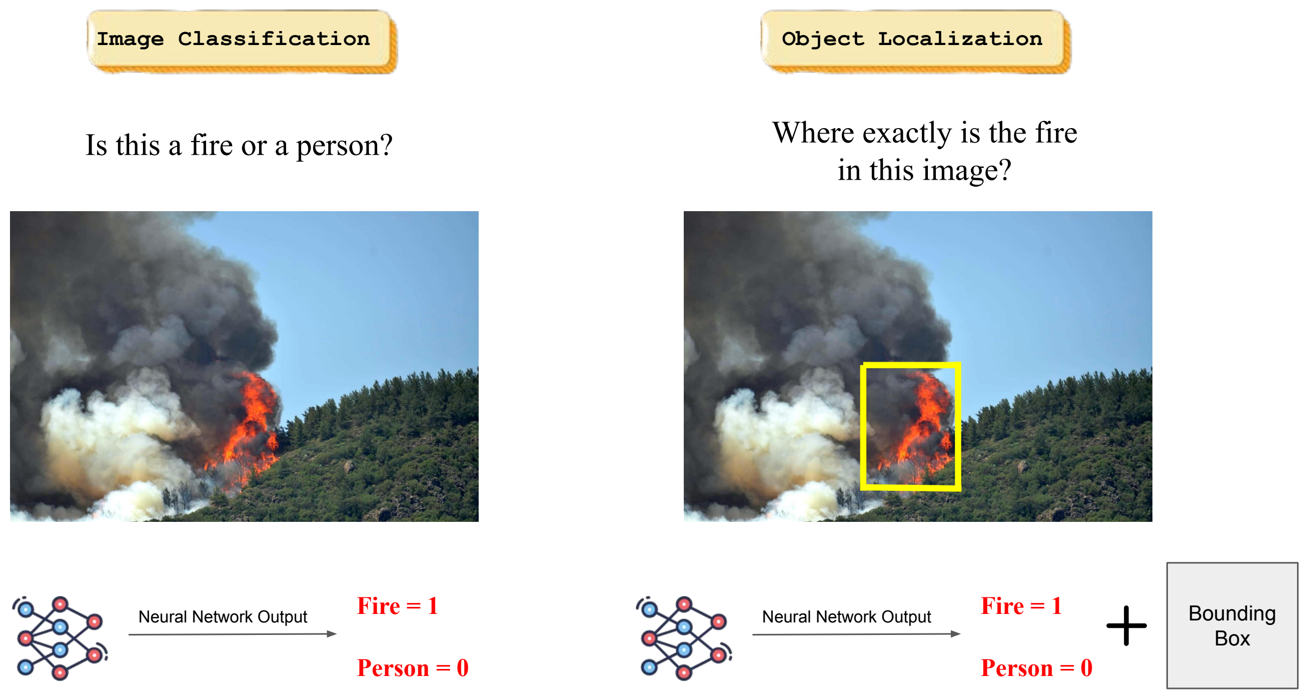


Figure 5.2: Image Classification and Localization

How is classification and localization done? To answer this particular question, we first need to see how to represent the following things: The probability that the image contains an image that belongs to class a or class b or class n (depending on how many classes we have). To which class belongs the image we are studying. The center of the image, i.e. x and y. The dimensions of the image, i.e. length and width.

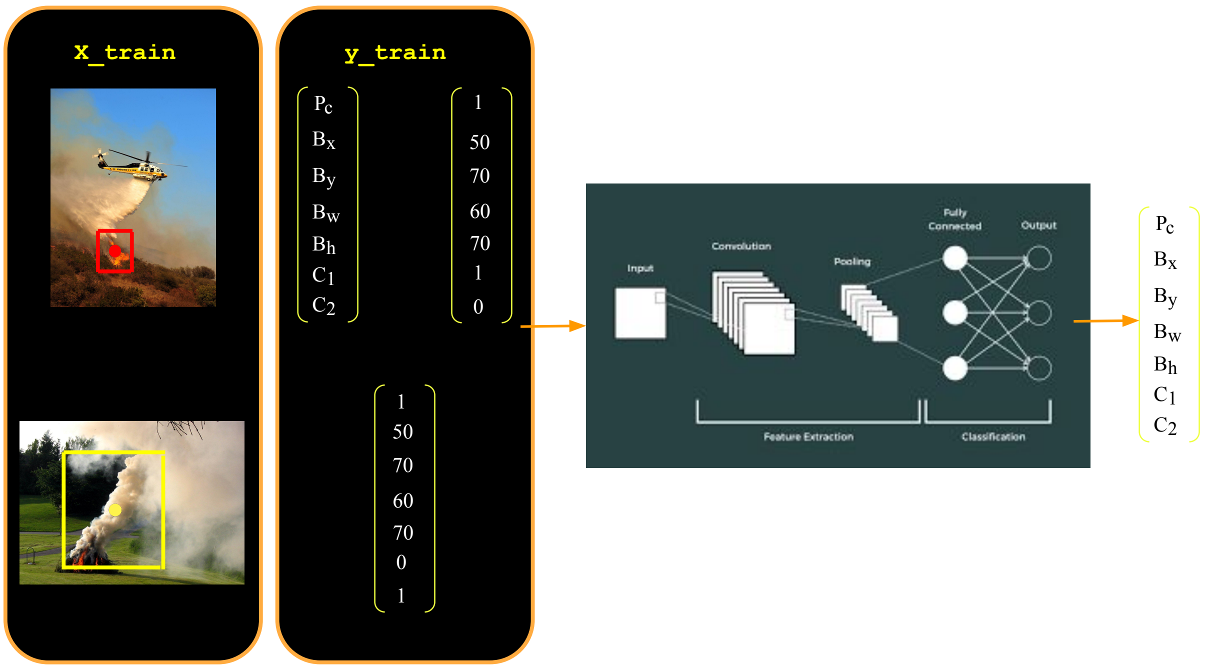


Figure 5.3: Training of the YOLOv5

Let's analyze what we see in the image, how we analyze the boundboxes and how the neural network is trained. The classes we are analyzing in the figure are: Fire and Smoke.

The matrix we're seeing in the figure 5.3 can be explained as follows:

- P_c The propability of a class. If there's a fire or a smoke in the fire, this number will be 1. If there's no smoke or fire the probability will be 0.
- B_x The coordinate in x-axis that determines the centre of our bounding box.
- B_y The coordinate in y-axis that determines the centre of our bounding box.
- B_w The width of the bounding box.
- B_h The height of the bounding box.
- C_1 Fire class
- C_2 Smoke class

So what we do here is to annotate the training images and strictly define the bounding box. The centre of the boxes are displaying with the red and yellow dot respectively. But what if we have multiple objects or we need to annotate many objects within 1 image? This is where the grid concept we mentioned above fits in. Below we will illustrate what training looks like with multiple objects. So, the above process we did about a single image is going to be the same here, with the difference that we'll apply the annotation and plotting for each frame of our grid.

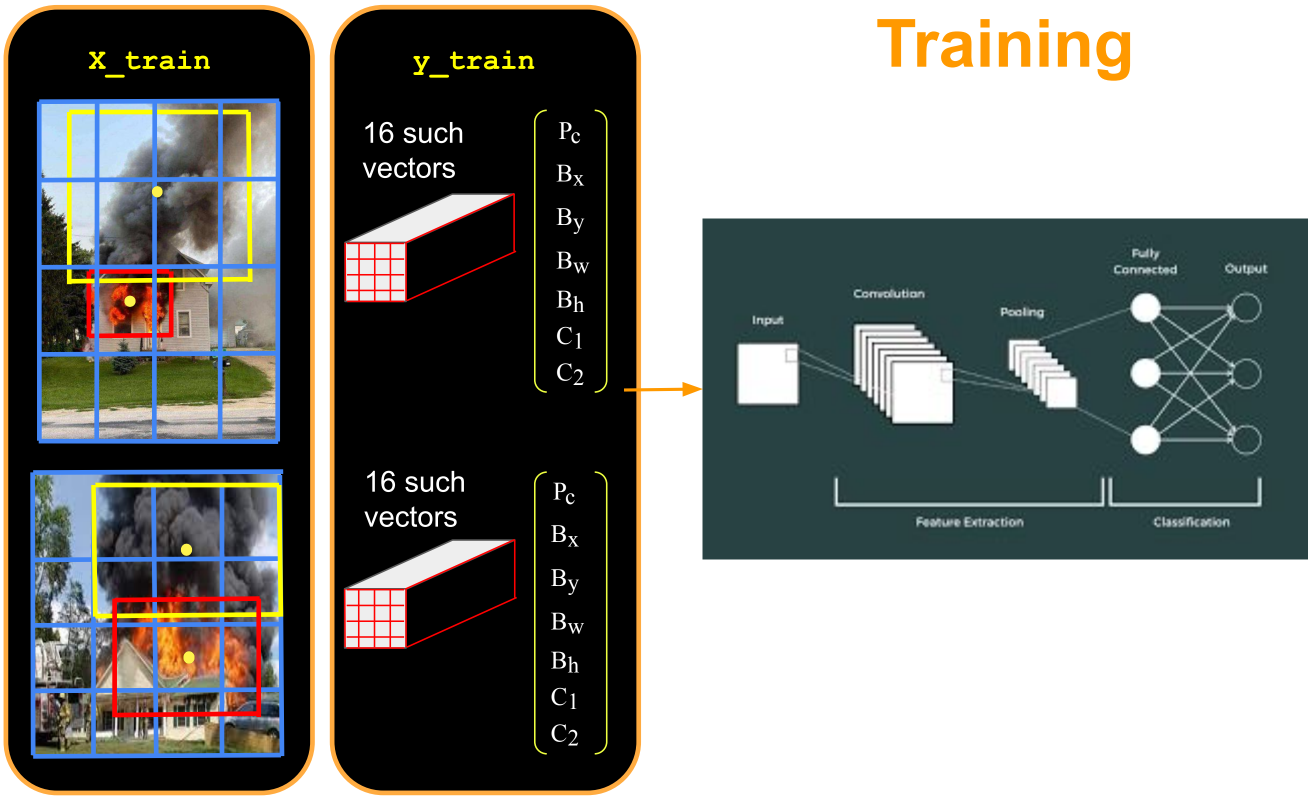


Figure 5.4: Training with multiple Objects

Now, when our model will predict an input image or plenty of images, it'll produce a vector of $n*n$ where $n =$ size of the grid. The YOLO algorithm took him name by the approach it's doing. When we're predicting an image, we are not repeating the prediction process. In on forward pass we are making all of our prediction, we don't need to repeat the prediction for each cell of our grid. Also, one issue with this approach is that of the multiple bounding boxes. The solution to this problem is the IOU (Intersection over union). The formula for this approach is:

$$IOU = \frac{intersect\ area}{union\ area}$$

This figure below show us the process of the prediction of YOLO model.

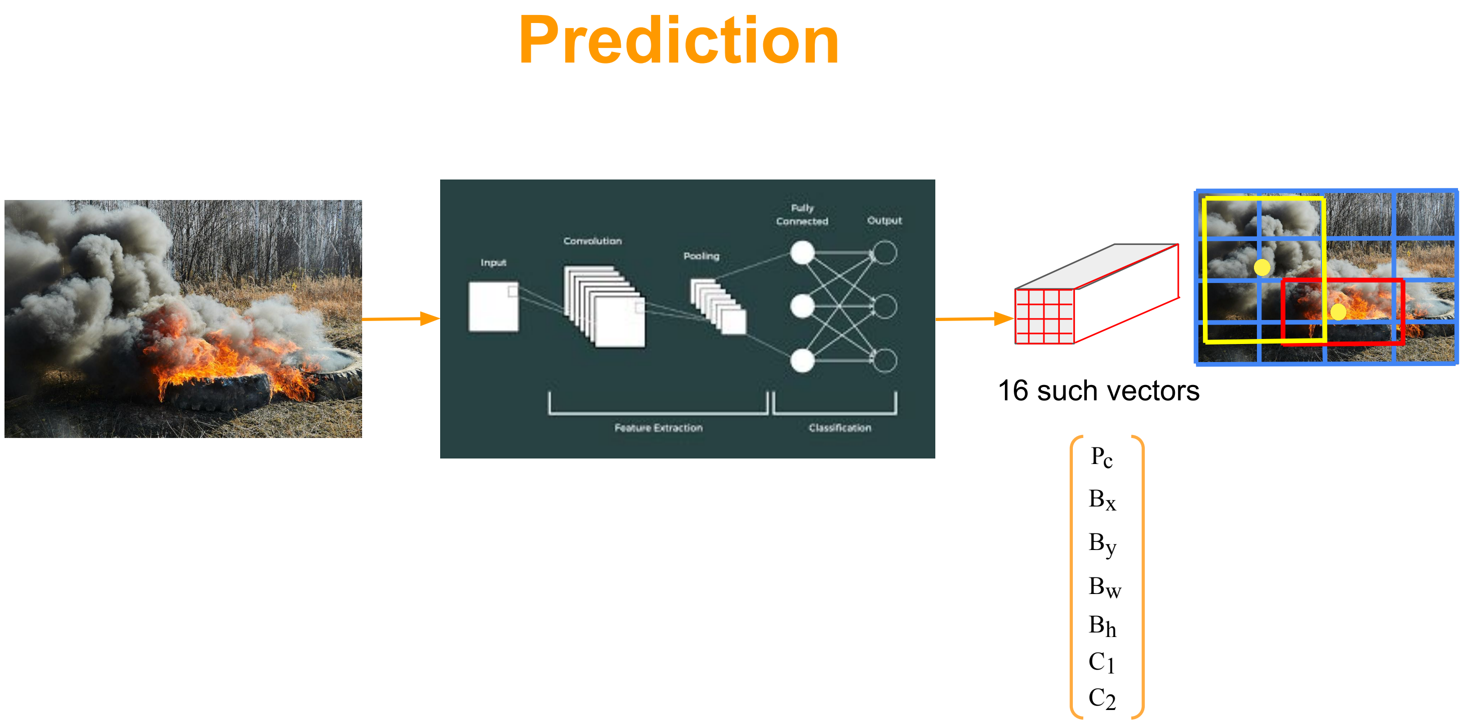


Figure 5.5: Prediction with multiple Objects

5.0.2 Image Fire Dataset

The dataset we have used is that of **FireNET**. FireNET is a real-time fire detection project containing an annotated dataset, pre-trained models, and inference codes, all created to ensure that machine learning systems can be trained to detect fires instantly and eliminate false alerts.

This is part of DeepQuest AI's training machine learning systems to perceive, understand and act accordingly in solving problems in any deployed environment.

We have used the annotated dataset. This consists of 502 images, which have been split into 412 images for train and 90 images for validation. The images contain both small-scale fires (lighters, candles) and large-scale fires (in cars, buildings and also forests).

5.0.3 Training Custom Model

The training of the custom model was done using the Google Colab platform. The run was done using GPU since we don't have our own graphics card that we can use for the training.

Figure 5.6: Tesla T4 GPU

The GPU we have used is the Tesla T4. Before doing the training, we had to do some data preprocessing. The data we have to do the training is, as we mentioned above, an annotated dataset. The annotation is in xml format. To be able to do the training effectively, we need to do some data labeling, since the system we're using is the supervised learning. The number of classes we have in our model is 1 and is that of fire. So our model can only be used with fire inferences.

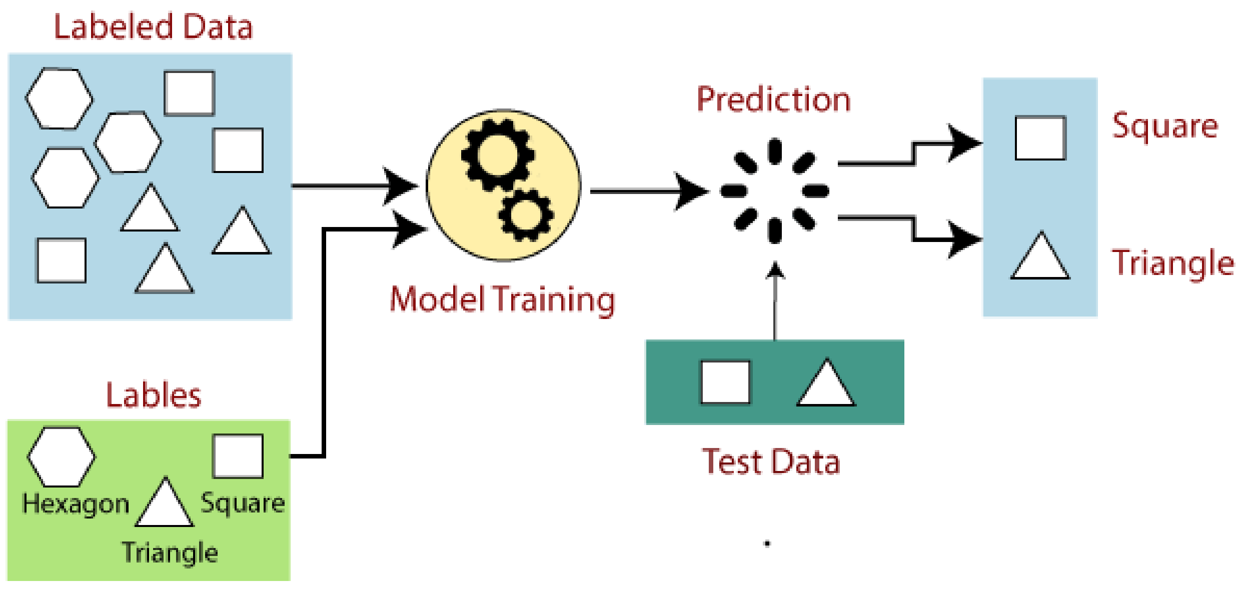


Figure 5.7: Supervised Learning

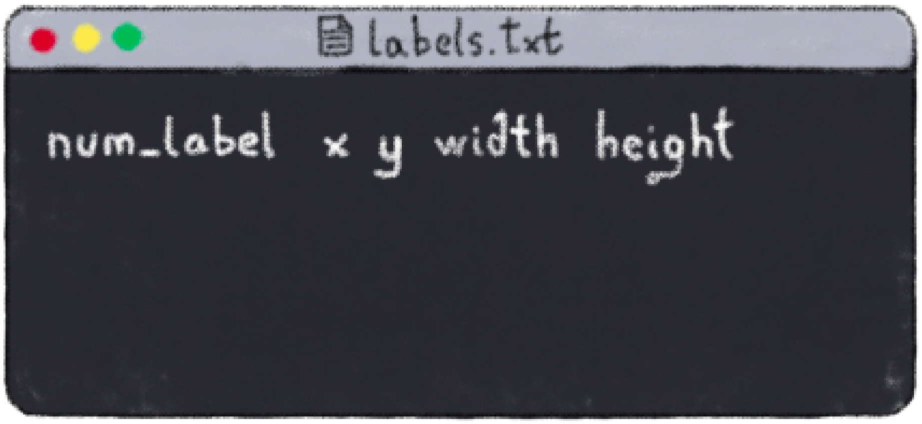
Labels, is be written as follows:

1 num-label: label (or class) number. If you have n classes, the label number will be between 0 and n-1

2 X and Y: correspond to the coordinates of the centre of the box

3 width: width of the box

4 height: height of the box



```
labels.txt
num_label x y width height
```

Figure 5.8: Label format

After the conversion from .xml to .txt, we are ready to train our model. Below, we'll explain the type of arguments we used for the training.

- **img**: Define input image size. Here, the input image size is 416.
- **batch**: Determine batch size. The Batch size we chose is 16. The batch size defines the number of samples that will be propagated through the network. For instance, let's say that we have 1050 training samples and we want to set up a batch size equal to 100.

  The algorithm, takes the first 100 samples (from 1st to 100th) from the training data-set and trains the network. Next, it takes the second 100 samples (from 101st to 200th) and trains the network again. It has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize [12].
- **epochs**: Define the number of training epochs. We've trained our model for total 150 epochs. One Epoch is when an entire dataset is passed forward and backward through the neural network only once.
- **data**: Our dataset location that is saved in the platform enviroment.
- **weights**: Specify a path to weights to start transfer learning from. Here, we choose the generic COCO pretrained checkpoint. The basic premise of transfer learning is simple: take a model trained on a large dataset and transfer its knowledge to a smaller dataset. For object recognition with a CNN, we freeze the early convolutional layers of the network and only train the last few layers which make a prediction.
- **cache**: Cache images for faster training. Image caching essentially means downloading an image to the local storage in the app's cache directory (or any other directory that is accessible to the app) and loading it from local storage next time the image loads.

5.0.4 Evaluation of the Model

Below we will see with real measurements, if the training we did on our model was good enough. We will evaluate it by analyzing how it fared in the various scores and tests we have tested it. Below we see a prediction batch that our model has run.

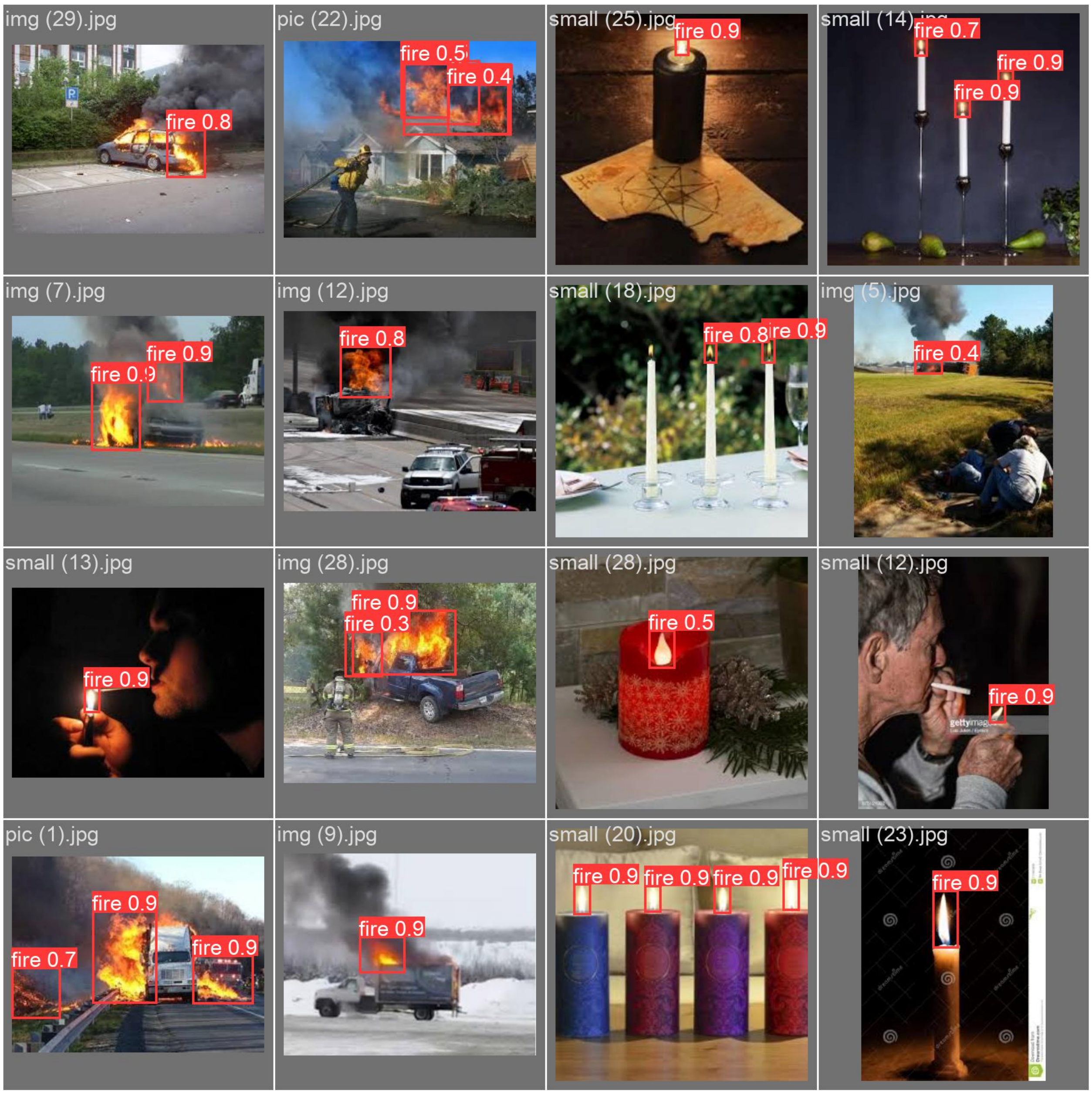


Figure 5.9: Trained Model Prediction Batch

As we can see, the training of our model was quite good. We can conclude the following from the predictions we see.

First of all, the **confident score** that exists in each photo is quite good. That is, when there is a clear image of the fire, our model recognizes it, with fairly good accuracy so to speak. However, we also see that there are some cases where the fire is quite small (like the one in the candles, in image 18), or where the fire is from a far distance. In these cases, our model either achieves a low confident score or does not even recognize the fire. Of course, we can avoid this problem by giving our model more photos during the training process.

Nevertheless, we find that our model predicts new images quite well. But how can we measure this? We understand that it is not possible every time we want to check if our model is good, to check ourselves if the annotation and the confident score are satisfactory on each image. So, in the object detection field, there are certain metrics that tell us if our model is good, and thus, the work of evaluation is done automatically. This metric is mAP (mean average precision), which we will analyze further. Let's first look at the general results of our model in the evaluation that was done.

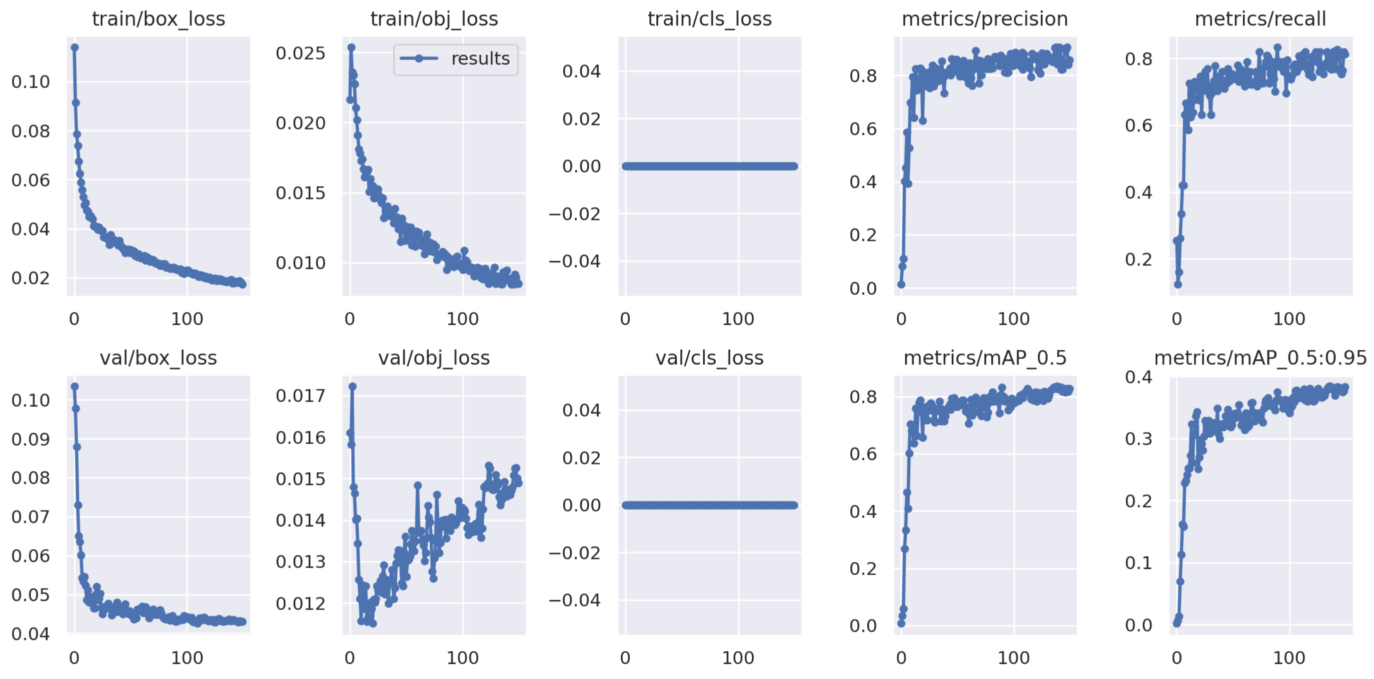


Figure 5.10: Evaluation Results

There are three different types of loss shown in Figure 5.10, box loss, objectness loss and classification loss [11].

- **box loss**: The box loss represents how well the algorithm can locate the centre of an object and how well the predicted bounding box covers an object. It's also know as bounding box regression loss MSE (Mean Squared Error).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

- **objectness loss**: Objectness loss is essentially a measure of the probability that an object exists in a proposed region of interest. If the objectivity is high, this means that the image window is likely to contain an object. Overall, the confidence of object presence is the objectness loss BCE (Binary Cross Entropy).

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i + log\hat{Y_i} + (1 - Y_i) \cdot log(1 - \hat{Y_i})$$

- **classification loss**: Classification loss gives an idea of how well the algorithm can predict the correct class of a given object. Since we don't have more than 1 class (fire), the classification loss (Cross Entropy) is always zero.

$$L_{CE} = -\sum_{i=1}^{n} Y_i \cdot log\hat{Y_i}$$

In the figure, we can also see that we measured our model performance with the precision and recall metrics.

Precision refers to the number of true positives divided by the total number of positive predictions(i.e, the number of true positives plus the number of false positives). Or, in other words, is a measure of, ”when your model guesses how often does it guess correctly?”.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made. Or in other words is a measure of ”has your model guessed every time that it should have guessed? Are there any guesses that they shouldn't have been guessed?”.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Let's now do some explain of the most important metric in Object Detection Field, the **mAP**. Models that involve an element of confidence can tradeoff precision for recall by adjusting the level of confidence they need to make a prediction.

In other words, if the model is in a situation where avoiding false positives (stating a RBC is present when the cell was a WBC) is more important than avoiding false negatives, it can set its confidence threshold higher to encourage the model to only produce high precision predictions at the expense of lowering its amount of coverage (recall).

The process of plotting the model's precision and recall as a function of the model's confidence threshold is the precision recall curve. It is downward sloping because as confidence is decreased, more predictions are made (helping recall) and less precise predictions are made (hurting precision).

We can also measure corectness using interesection over union. Object detection systems make predictions in terms of a bounding box and a class label.

In practice, the bounding boxes predicted in the X1, X2, Y1, Y2 coordinates are sure to be off (even if slightly) from the ground truth label. We know that we should count a bounding box prediction as incorrect if it is the wrong class, but where should we draw the line on bounding box overlap?

The Intersection over Union (IoU) provides a metric to set this boundary at, measured as the amount of predicted bounding box that overlaps with the ground truth bounding box divided by the total area of both bounding boxes. Picking the right single threshold for the IoU metric seems arbitrary. One researcher might justify a 60 percent overlap, and another is convinced that 75 percent seems more reasonable. **So why not have all of the thresholds considered in a single metric? Enter mAP**.

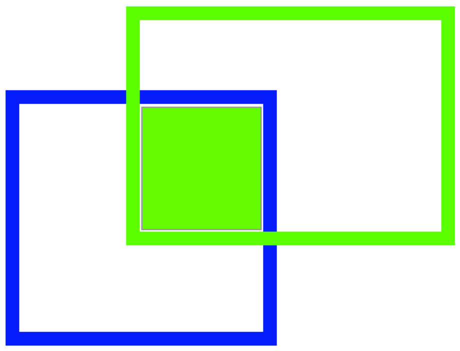


Figure 5.11: IOU

In order to calculate mAP, we draw a series of precision recall curves with the IoU threshold set at varying levels of difficulty.

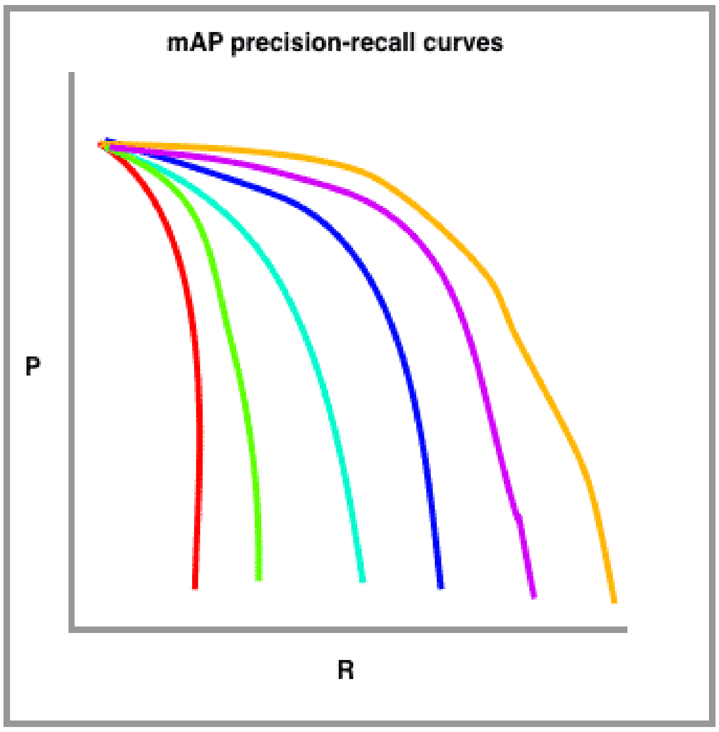


Figure 5.12: mAP precision-recall curves

In this sketch, red is drawn with the highest requirement for IoU (perhaps 90%) and the orange line is drawn with the most lenient requirement for IoU (perhaps 10%). The number of lines to draw is typically set by challenge. The COCO challenge, for example, sets ten different IoU thresholds starting at 0.5 and increasing to 0.95 in steps of .05.

The metric calculates the average precision (AP) for each class individually across all of the IoU thresholds. Then the metric averages the mAP for all classes to arrive at the final estimate. But, if our case, because we have only 1 class, the mAP will be the same, because we don't have any other classes for the mAP to be considered.

6. UAV AND MOBILE APP DEVELOPMENT

6.1 Drone Programming

To program the DJI Ryze Tello, we used the **python** programming language (Python 3.8.0 version). We felt that this is the best choice of language, because of its ease, compatibility and also the variety of libraries that this programming language offers us.

6.1.1 Tello SDK 2.0

The Tello SDK connects to the aircraft through a Wi-Fi UDP port, allowing users to control the aircraft with text commands.

TELLO
SDK 2.0 User Guide
V1.0 2018.11

Figure 6.1: Tello SDK 2.0

Also, we have to mention that we imported the ***djitellopy*** library, and the ***tello*** module, in order to interact with the drone.

Table 6.1: Drone Commands

Command	Description	Possible Response
.connect()	Establish connection between PC and Drone.	ok/error
.takeoff()	Auto takeoff of the Drone.	ok/error
.land()	Auto landing of the Drone.	ok/error
.streamon()	Enable video stream.	ok/error
.streamoff()	Disable video stream.	ok/error
.get_frame_read()	Reads the current frame from video stream.	ok/error
.get_battery()	Reads the current battery status.	ok/error
.send_rc_control(lr, fb, ud, yv)	Controls the movement within 4 int.	ok/error

The most important command here that we're using, is the *send_rc_control*, because it's the command that allows us to control manually our drone from the keyboard of our PC.

- **lr(int)**: Stands for left_right_velocity (-100,+100) (left/right).
- **fb(int)**: Stands for forward_back_velocity (-100,+100) (forward/backward).
- **ud(int)**: Stands for up_down_velocity (-100,+100) (up/down).
- **yv(int)**: Stands for yaw_velocity (-100,+100) (yaw).

The figure below it's explaining quite graphically how's the drone is navigating through the keyboard.

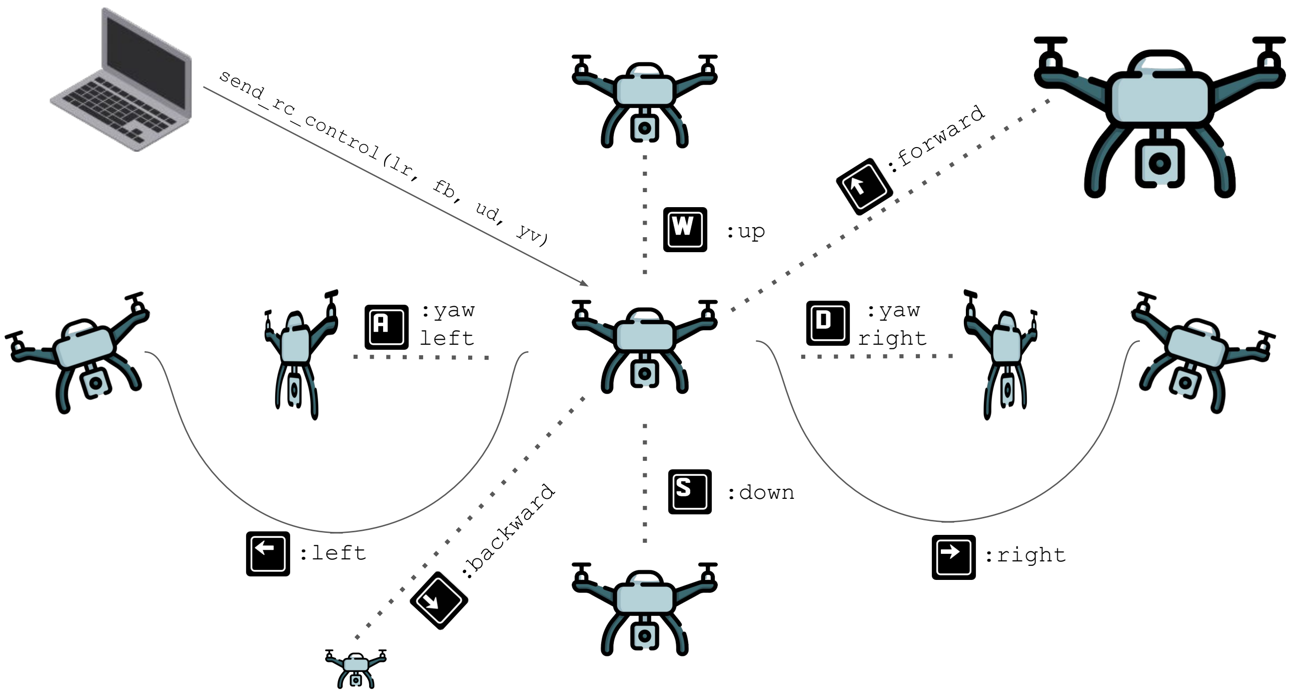


Figure 6.2: Drone Navigation

The library we used to detect keyboard activity was the KeyPressModule from pygame 2.1.2. Pygame is a set of Python modules designed for writing video games. Pygame adds functionality on top of the excellent SDL library. This allows us to create fully featured games and multimedia programs in the python language.

Figure 6.3: Pygame

For instance, when we press the 'w' key, as we see in the photo, we're sending via the send_rc_control this specific array: [lr, fb, ud, yv] $->$ [0, 0, int, 0]. Thus, the drone is elevating int cm up in the air. The int value represents the 'speed' that we defined in our program and determines the speed of the movements that are being expressed as cm. The maximum value of speed is +100. For each 'positive' movement (positives movements are the movements that when you'll give them positive values, are doing specific things, as for going up, forward, turning right or going left) there are also specific outcomes. For each 'positive' movement, we have also a 'negative' one, which simply is the inverse move of its corresponding positive one.

6.1.2 Python Scripts

To achieve our goal, we had to write an extensive software that combines all the utilities that we need to combine in order to reach our goal. The utilities we want to achieve in this thesis are the following ones.

- **Movement of the drone**: Initially, the most important function of the UAV is the movement it will perform. Our initial approach to this task is to move the drone through the computer, using a real operator. However, the final plan and where we intend as a continuation of this project, is to find specific algorithms that can perform this movement automatically, without the need for human presence. Some initial ideas are the meandering exploration of the space, or even the mapping of the space of interest as a complex of graphs, where we will be able to apply search algorithms on it, such as for example BFS, DFS, Dijkstra, A*. Also, as we also mentioned in section 4, an initial metric that will help us to ”evaluate” in real time, if the decisions we make regarding the movement of the drone are good or not, we have constructed a Cost function, which takes as data the location of the drone, its relative distances to the intersection of the wind lines and the area of interest, as well as the time the Drone flies away from a specific ”danger” area.

  The software we developed in this particular case concerns the movement that the UAV will make through commands from the computer. As we mentioned above, the movements will be made through specific keys that we chose from the keyboard. We followed the convention that already exists in most so-called video games. That is, the well-known combination of the W, A, S, D keys and $\uparrow (upkey), \downarrow (downkey), \rightarrow (rightkey), \leftarrow (leftkey)$.

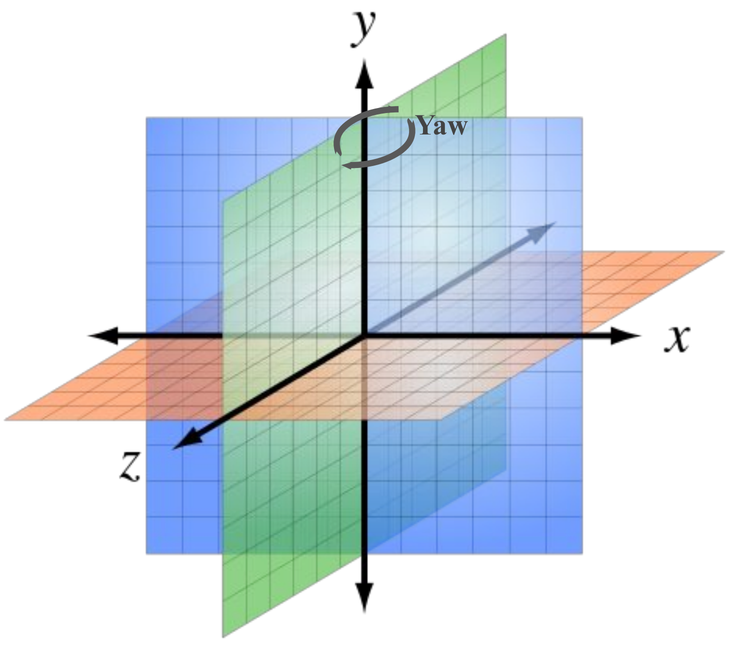


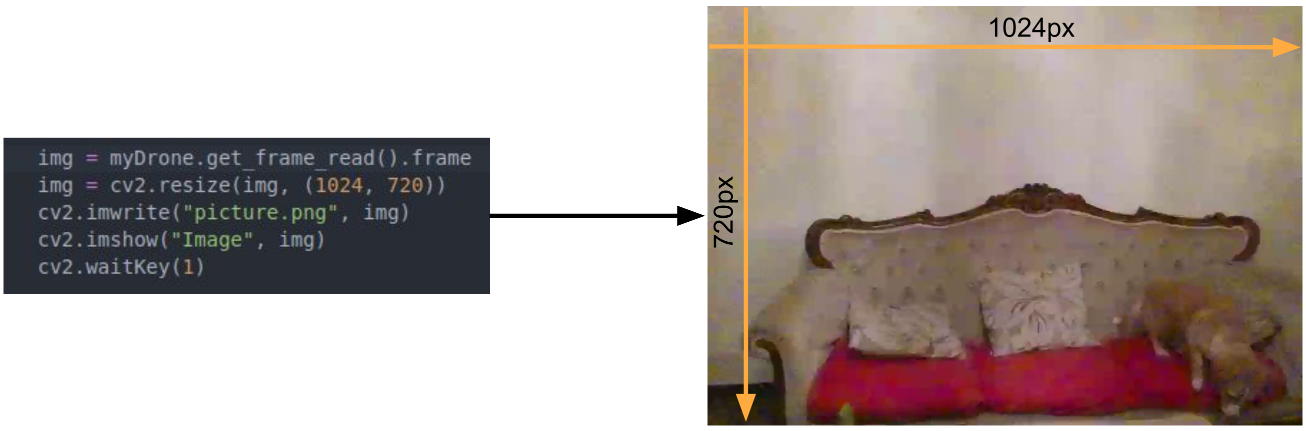
Figure 6.4: 3D Axis Movement

As we see, these are the movements we are doing in 3D axis. The movements we are doing in x-axis, are being controlled by the right and left key. The movements in z-axis, are being controlled by the up and down key. The movements in y-axis, are being controlled by the w and s key. And finally, the yaw movements on the right-clockwise and left-clockwise, are being controlled by the d and a key respectively.

- **Video and frame capture**: The next most important function that we need to achieve is that of capturing the video and saving images from our drone's camera. To achieve this, we first need to enable the ability to capture video from the drone's camera, via the .streamon() command. This command, as mentioned above, turns on video streaming. But, as we know, since the video is a series of continuous images (frames per second), we need to read continuously within a While loop, continuous frames which our drone outputs, through the command .get_frame_read().frame. To see in real time, the video we produce, we use the OpenCV library.

Figure 6.5: OpenCV

We display the video we take in a window which has dimensions (1024, 720). Finally, to record and save an image we want, again through the OpenCV library, we use the command .imwrite("name_of_the_saved_picture", frame). Below, we see how through these lines of code, we capture the capture of an image in practice.



```
img = myDrone.get_frame_read().frame
img = cv2.resize(img, (1024, 720))
cv2.imwrite("picture.png", img)
cv2.imshow("Image", img)
cv2.waitKey(1)
```

Figure 6.6: Tello captured frame

- **Fire Identification**: The most work is done by now, because we already trained our neural network and we've saved the weights of our model. So, the only thing here to do is to connect the YOLOv5 trained model with our custom dataset, with the UAV's camera. Unfortunately, YOLOv5 does not supports directly live video streaming from a different video source different than the PC's webcam. If that was possible, there's a python file (*detect.py*), that we could use to do the fire detection and our life would become more easier. Lucky for us, we can feed our model with each frame from our video streaming and do the job.

  Through this line of code *results = model(frame)*. In results variable, we're getting back the possibility of the object to belong in the fire class, and also the coordinates of the bounding box. Results variable follow the format of 5.8 figure, with the only difference that the first 4 elements of our array are the coordinates of our bounding box and the last element (5th), is the confidence of the object belongs to the class we annotated it (fire class).

  Below, we'll see a first approach of this script. We used a lighter for this experiment, and the figure below will show us the way that the object and also the bounding box is displaying in our window.

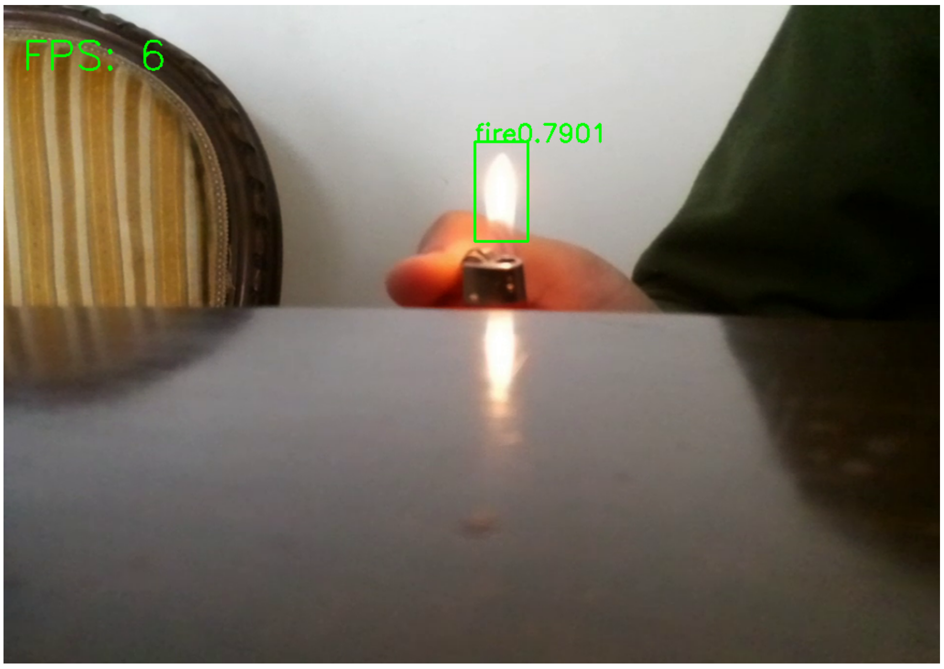


Figure 6.7: Lighter Fire Detection

  As we see, we managed to get a score of 0.7901 of a small scale fire. The FPS are around 5-6, because we don't use any GPU, but we run the script through our CPU.

- **Keeping track of the Drone's position**: One such important as the previous scripts, it's also this one which keeps track of the drone's movement and position. Through this script, we can easily calculate the drone's position, which is relative to the position of the starting point (where our computer will be).

  In order to achieve this, we open a OpenCV window in our PC, and then we display the drone's movement within this window, so we can see and track in real time our drone's movement. It's a very helpful utility, because we are being able to follow a specific movement (for instance, a meandric approach for our mission path).

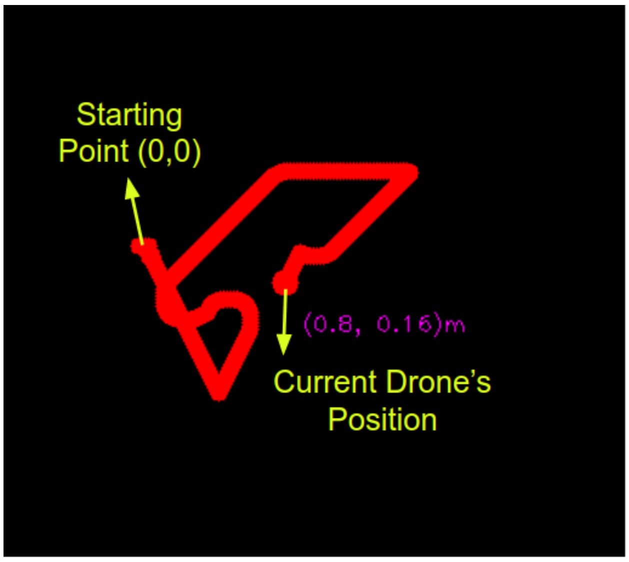


Figure 6.8: Mapping of the Surveillance

As we can see, we mapped the whole drone's surveillance. The coords of the starting point are being fixed as (0,0), on (x,y)-axis. The current drone's current position is showing with the red circle, and the relative distance within the starting point is being displayed with the purple coordinates. For instance, the current drone's position is 0.8m more right than the starting point, and respectively, is also further down for 0.16m.

The last task we have to complete in order to have a fully functional connected program, is to compute the longitude and latitude of the drone's position. That's an easy task, from the moment that we already know the coords of the starting point.

First of all, let's define the coords of starting points as (str_x, str_y) or as $(latitude, longitude)$. The formulas that we need in order to do this computation are the following ones:

Computation of the meter, based on the earth radius:

$m = (1/((2 * pi/360) * earth\_radius))/1000$

Computation of the new latitude (drone's x-axis position):

$latitude_{new} = str_x + (variation_x \cdot m)$

Computation of the new longitude (drone's y-axis position):

$longitude_{new} = str_y + (variation_y \cdot m)/math.cos(str_x * (pi/180))$

6.2 App of Usage

The last part that we will look at is that of the mobile application. The mobile application will be updated by the computer and will take the data that we have acquired during our aerial observation through the UAV. For reminder, these data are: The date and hour of the incident, the location of the fire and a picture of the fire.

6.2.1 TCP/IP Communication

Due to the architecture and structure of the system we propose, we quickly come to the following conclusion. That the computer to be able to communicate with the computer needs to be constantly connected to the drone's network, otherwise communication becomes impossible, and no data transmission will be possible.

The problem is that in order for the computer to send messages to an Android device, it needs to be connected to a WiFi network. So we immediately understand that it is not possible to simultaneously send messages to smart devices and be connected to the Drone network.

To solve this problem, our options are these:

1) Either as soon as the recognition is done, our Drone immediately returns to the base, we connect to a network that will provide us with access to the internet, and from there we send a message to a smart device. This, as we understand, is a problem because the time it will take the drone to return to the base can be enough for the fire to develop rapidly in a short period of time.

2) Either, with a second computer, communicate via TCP/IP, and the second computer that will receive the messages from the first, be connected to a network in order to send data to the smart device.

TCP/IP uses the client-server model of communication in which a user or machine (a client) is provided a service, like sending a webpage, by another computer (a server) in the network.

Collectively, the TCP/IP suite of protocols is classified as stateless, which means each client request is considered new because it is unrelated to previous requests. Being stateless frees up network paths so they can be used continuously.

The transport layer itself, however, is stateful. It transmits a single message, and its connection remains in place until all the packets in a message have been received and reassembled at the destination.

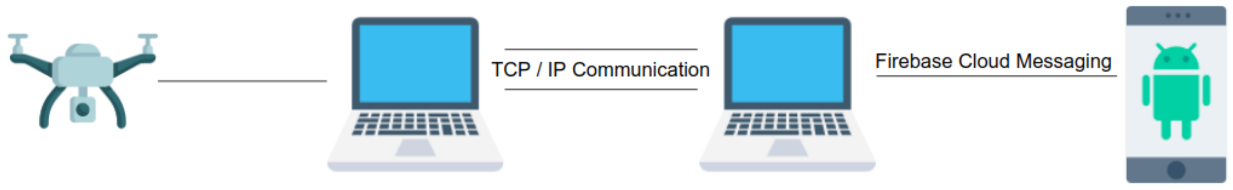


Figure 6.9: TCP/IP Communication

6.2.2 Android Application

The Application we developed, it's written in Android Studio Project. We decided this enviroment because we can directly test our App through an Emulator or an actual Physical Device. Because of the utilities that enviroment has to offer us, it's the best choice for our case. The language that the App is written is in Java (14.0.2). The messaging is delivered through the Firebase Cloud Messaging, a software that covers our needs and sends a push notification directly into a smart device. Through a PHP (7.2.24) Script, we send the data into the FCM API and through this, it goes within a POST Request, to the FCM Server, and then directly to selected smart devices.

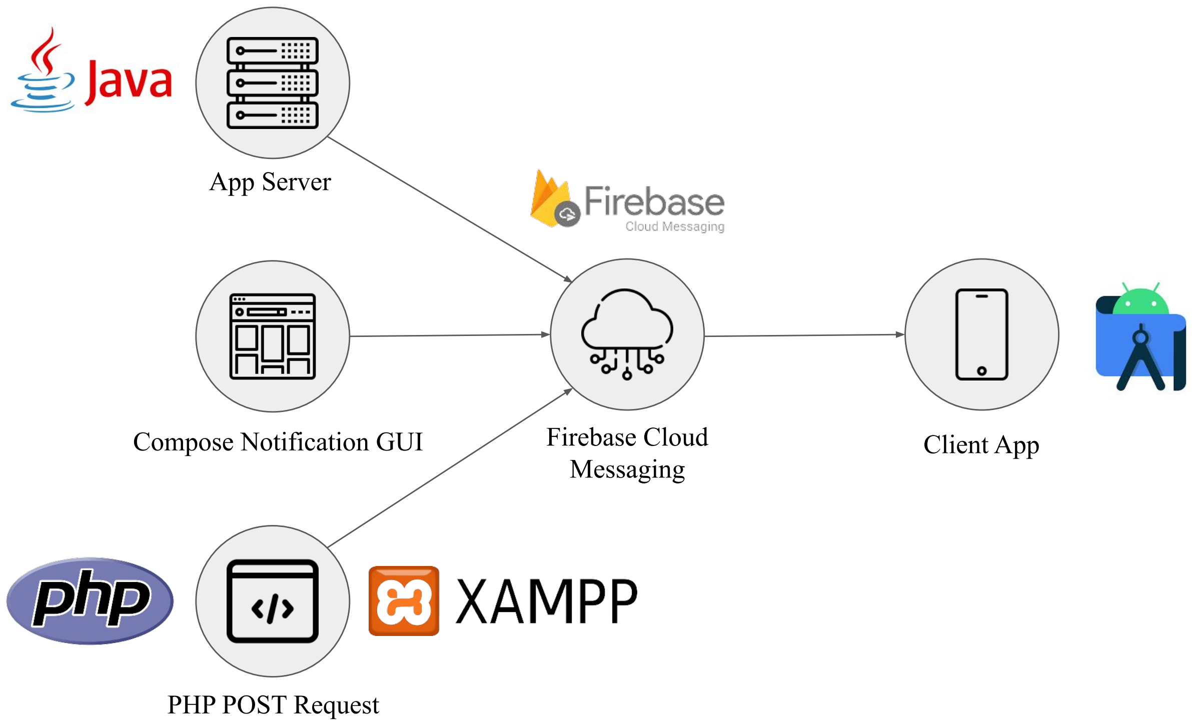


Figure 6.10: Notification App Workflow

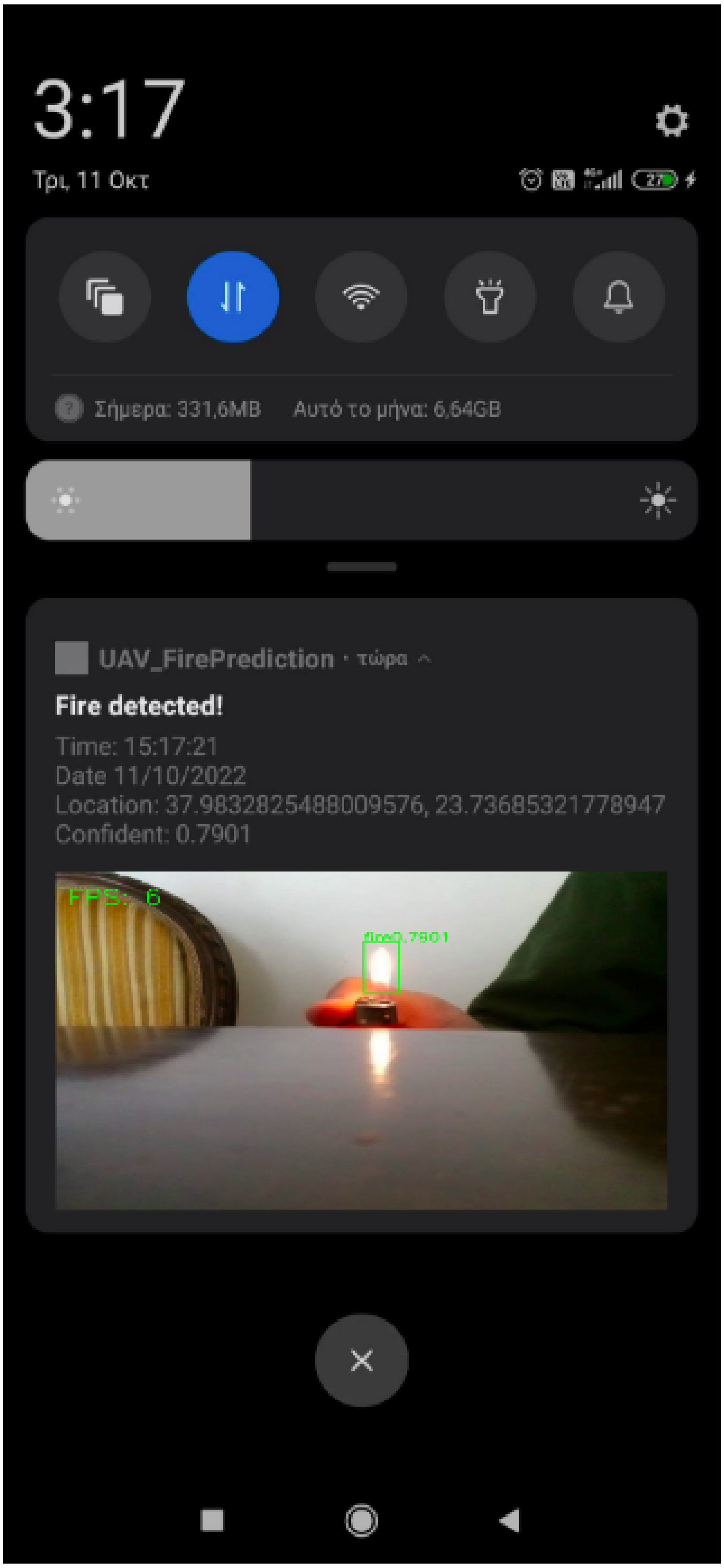
The application we have created works as follows. First we connect our app to Firebase Cloud Messaging. Then, going to the Cloud Messaging field, through the Compose Notification GUI provided by Firebase, we send a test push notification to our application, to see that we have successfully connected.

It is necessary to obtain the API Key from Firebase, in order to achieve communication both through the GUI and through the POST request that we will see later.

The next step is to send a POST request through a php script. When we create the body of the message (which will be dynamic and will change according to the data that the Drone sends to the computer), then we define as the sender the device or devices from which we have defined as Clients. Then, in order to finally be able to make this POST request, we use XAMPP to define our server (in this particular case it will be localhost), in order to be able to take the role of the sender of the message.

XAMPP has the ability to serve web pages on the World Wide Web. A special tool is provided to password-protect the most important parts of the package. XAMPP also provides support for creating and manipulating databases in MariaDB and SQLite, among others.

Below, we'll see a fire identification which took place and then, a push notification send immediately into our device. We can see that the push notification is displaying in our phone's screen.



Fire detected!

Time: 15:17:21
Date 11/10/2022
Location: 37.9832825488009576, 23.73685321778947
Confident: 0.7901

Figure 6.11: Push Notification Display

6.3 Results and Experiments

For the proof of concept, we decided to test our system by setting a real fire in a controlled forest area. The fire was a small to medium scale and it was controlled all the time through this experiment. The fire lasted about 10min and it was constantly monitored by human presence.

All the necessary protective measures were taken into account (there was a water supply near the experiment site, a fire extinguisher to shut down the fire as well as other protective measures, such as sand, as well as a special fire blanket used by fire safety professionals to extinguish the fire). Below we will see the weather data that prevailed in the specific area at the time of the experiment.

6.3.1 Real Fire Experiment

Table 6.2: Atmospheric Data

Wind Speed	Wind Direction	Humidity	Temperature	Cloud Report	Date	Time
2kts	43°	42%	24C°	Sunshine	09/10/2022	17:20:42

In order to check our model ability to identify the fire, we tested our drone in multiple altitudes. So, we're going to see in which altitude had the best detection and if our model could possibly be useful at different conditions. The problems we've faced are two in this case: The first is the wind speed, which was good in our case, but because the fire creates self winds conditions and can easily change the *wind's speed and direction*, when we were back from the fire (in the first case, we are looking the fire forward), the drone had a hard time staying in the same position. The second problem, is the *battery life*. Each test lasted only about 13min, and we had to change the batteries in mid of the flight in order to continue our experiment

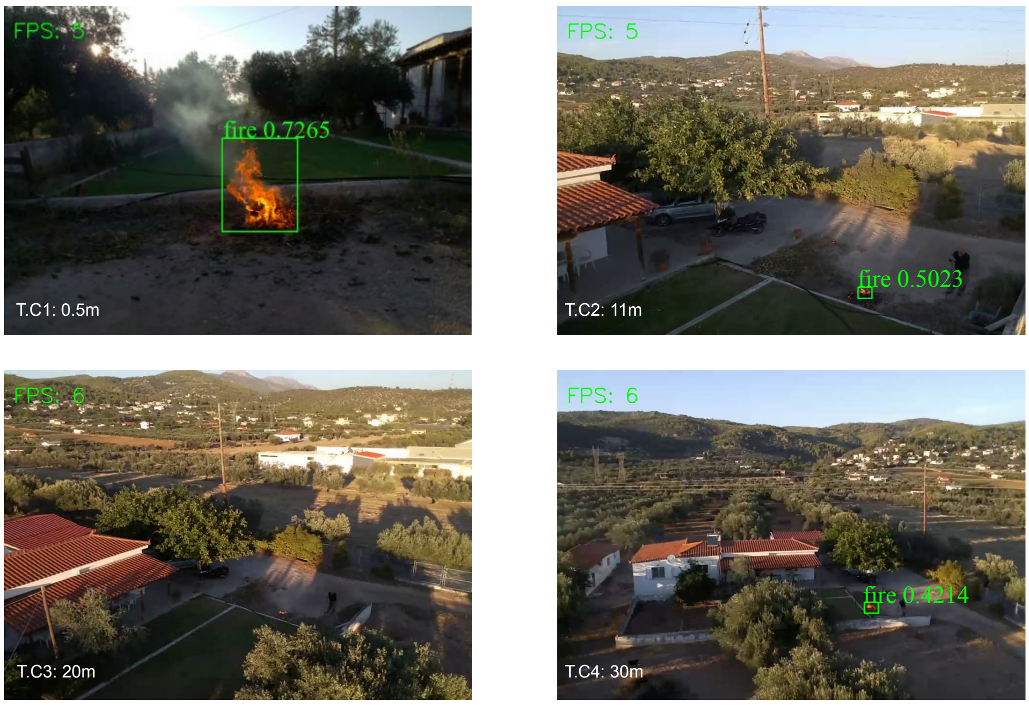


Figure 6.12: Fire cases

- **Altitude: 0.5m**: So, the first case we tested, was this of the altitude of 0.5m. Our drone was near from the ground, and was looking straight into the fire. That's the best case we can test, because we are looking straight into the fire. In real test cases, that would be not possible, because we would doing the surveillance far away from the ground in a high altitude.

  As we can see, the confidence score is pretty good, with **0.7265** score. That means that in lower altitudes the identification of the fire is doing good and we've had no problem detecting the fire.

- **Fire at 11m**: The second case we tested, was this of the altitude of 11m. Our drone was pretty high from the ground, and didn't had the best view of the fire. Nevertheless, the altitude was good because in real test cases that's a good high to patrol the ground.

  Here, we managed to get a confidence score of **0.5023**. As we see, the higher we go, the confidence is getting lower, due to the size of the fire. But, we still managed to detect the fire.

- **Fire at 20m**: The third case we tested, was this of the altitude of 20m. As we can see at the figure below, our model didn't worked well and it didn't recognize the fire. That's something we expected to happen, because the fire is not showing well in the image and also the altitude was higher than the previous one. In order to detect the fire we need a good image of it, otherwise, it's hard for our model to detected it.

- **Fire at 30m**: The last and 4th case we tested, was this of the altitude of 30m. This is the higher we managed to get so far, but luckily, we managed to detected it. We managed to get a score of **0.4214**. As we can see, the higher we get, the worst image we have of the fire and thus, the detection is getting harder and harder. We need a large scale fire in order to detected it successfully.

6.3.2 Response Time

In order to test the quality of our service, we had to count the response time. We define the response time as the time it passed from the moment we detected the fire until we send the message in our device. We tested the response time at Test Case 1: 0.5m. If we formalize the response time, we'll get something like that:

$$Response\_time : t(return\_to\_base) + t(connect\_to\_WiFi) + t(send\_messages))$$

Let's explain what these factors are.

- $t(return\_to\_base)$: This time parameter is the total time that our drone needs to return to home from the point of detection. This factor depends, because our UAV might be far away from the base, or close to it. It also depends on the direction of the wind and the speed of it, because if the wind is face to our Drone, it's more hard for it to get pass through it, thus the time it needs to return will be longer.

- $t(connect\_to\_WiFi)$: This factor is simply the total time that our computer needs to disconnect from the Drone's network and connect into a different one that provides us access to the web. Let's remind why we need to get access to a WiFi network: Because we need to send a message from our computer to selected devices.

- $t(send\_message)$: This time factor is just the time we need from the moment we run the PHP script until the moment the message lands into the selected device. This time parameter is almost zero, but depends on the speed of our internet connection.

So, in our case, we counted the time for all 3 of these time factors. We had the followup results:

t (return_to_base)	17s
t (connect_to_WiFi)	32s
t (send_message)	4s
t (sum)	53s

So, as we can see, our system managed to send the message in the selected android device, under 1 minute after it detect the fire. That's a pretty good time that it can be better, if we used 2 computers to send the data that we obtained from the drone to each other. That's a good time, considering the the fire was near our base and it had no great wind to make our work harder. The only time that we worry about is the first time factor ($t(return\_to\_base)$), because we may be far away from the base, and if the wind it's against the drone's tour to home, it may get even worse than 17s.

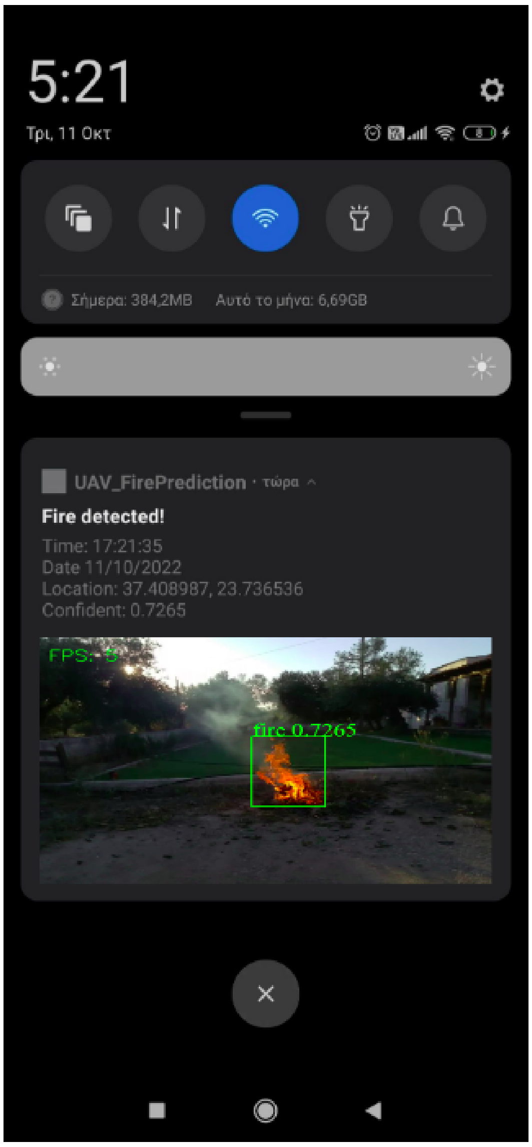

5:21
Τρι, 11 Οκτ
Σήμερα: 384,2MB Αυτό το μήνα: 6,69GB
UAV_FirePrediction · τώρα
Fire detected!
Time: 17:21:35
Date 11/10/2022
Location: 37.408987, 23.736536
Confident: 0.7265

Figure 6.13: Push Notification Test Case 1

7. CONCLUSIONS AND FUTURE WORK

Completing this thesis, we can say with certainty that the future in the prediction of forest fires has to do with the collaboration of all the technologies we have at our disposal (satellites, cameras, drones) and the processing of all this large amount of data we have at our arsenal for the detection or prediction of a forest fire (atmospheric data such as humidity, soil type, temperature, wind and also real-time data such as the images from the various parts of the area that we are studying).

We successfully managed using an IoT system to instantly identify a forest fire, also updating selected smart devices, in less than 1 minute. This recognition was done at a relatively low cost, comparing similar smart systems that can be very expensive. We managed to succeed with a semi-autonomous system, very little time in recognition and updating through a relevant application that we developed.

Nevertheless, there's a lot room for improvement yet and there are many things that we can do to make our system better, in terms of navigation and detection.

1. Make the system completely **autonomous**. Our system, is an semi-autonomous system that requires human presence in order to work. A person needs to be in the base to control and navigate the drone within the limits of the area of research. To make our system autonomous, the first thing we need to figure out, is what approach we should use for the flight path.

   Some ideas are to find a suitable search algorithm by converting the points of our search area into a graph. Then, we can combine this search algorithm by updating our cost function, which will make the drone take better decisions during the flight time.

2. Make the recognition better by feeding our model with **more data**. We used the FireNET dataset, which contains total of 502 images. We need more data in order to make the confidence score better and also increase the recall score. In other words, we need to aim finding all the possible fire instances even from far away.

   It's more easier and less damaging detecting a small-scale fire before it goes larger and more destructive. We also to focus on finding aerial images that contains fire, because we are using an aerial machine in order to detect fires. This work can be done by searching on internet for aerial fire images or by using the help of volunteers researchers and fire workers by providing us this kind of data.

3. Finding better starting points or better **'danger-zones'**. By this term, we mean that there are some points or areas inside the area of research, that are more 'dangerous' to others for the fire to start. These points can be detected by considering how much flammable are some materials, what's the angle of the sun hitting these points, the temperature and some more atmospheric data. These can help us to navigate through more important areas and thus finding the possible fire faster without using time.

ABBREVIATIONS - ACRONYMS

UAV	Unmanned Aerial Vehicle
CV	Computer Vision
TEV	Total Economic Values
OGC	Open Geospatial Consortium
CV	Computer Vision
FOV	Field of View
GIS	Geographic Information System
XML	Extensible Markup Language
mAP	Mean Average Precision
CNN	Convolutional Neural Network
RCNN	Region-based Convolutional Neural Network
MSE	Mean Squared Error
BCE	Binary Cross Entropy
IOU	Intersection Over Union
SDK	Software Development Kit
UDP	User Datagram Protocol
BFS	Breadth First Search
DFS	Depth First Search
TCP	Transmission Control Protocol
IP	Internet Protocol
FCM	Firebase Cloud Messaging
GUI	Graphical User Interface
API	Application Programming Interface

BIBLIOGRAPHY

[1] Alertwildfire: . https://www.alertwildfire.org/about/.

[2] Earth org: Causes of wildfires. https://earth.org/what-causes-wildfires/.

[3] European forest fire information system: Annual reports. https://tinyurl.com/2p8pj783.

[4] Gis (geographic information system) https://education.nationalgeographic.org/resource/geographic-information-system-gis.

[5] Redzone: Fire triangle. https://www.redzone.co/2016/02/17/wildfire-101-the-fire-triangle-and-the-fire-tetrahedron/ .

[6] Wikipedia: Wildfire. https://en.wikipedia.org/wiki/Wildfire.

[7] Yolov5 object recognition framework https://github.com/ultralytics/yolov5.

[8] László FÖLDI and Rajmund KUTI. General forest fires stats. In *Characteristics of Forest Fires and their Impact on the Enviroment*, page 5, 2016.

[9] Deyang Wu Hao Wu and Jinsong Zhao. An intelligent fire detection approach through cameras based on computer vision methods. In *Process Safety and Environmental Protection*, 2019.

[10] Andries Heynsa, Warren du Plessis, Michael Koschd, and Gavin Hough. Location of the camera network. In *Optimisation of tower site locations for camera-based wildfire detection systems*, page 2, 2019.

[11] Margrit Kasper-Eulaers, Nico Hahn, Stian Berger, Tom Sebulonsen, Øystein Myrland, and Per Egil Kummervold. Yolov5 losses. In *Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions using YOLOv5*, page 5, 2021.

[12] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. Generalization gap and sharp minima. In *Large-Batch Training for Deep Learning*, 2017.

[13] Diyana Kyuchukova, G.V. Hristov, Jordan Raychev, and Пламен Захариев. Early forest fire detection using drones and artificial intelligence. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019.

[14] Prof. Davide Pettenella, Prof. Marco Marchetti, Prof. Davide Marino, Dott. Angelo Marucci, Dott. Marco Ottaviano, and Dott. Bruno Lasserre. Total economic value components in forestry. In *Economic Damages Final Report - Economic Damages Study: 'Proposal for a harmonized methodology to assess socio-economic damages from forest fires in Europe'*, pages 13–15.

[15] Nguyen Thanh Toan, Phan Thanh Cong, Nguyen Quoc Viet Hung, and Jun Jo. Satelites wildfire recognition. In *A deep learning approach for early wildfire detection from hyperspectral satellite images*, 2019.

[16] Nicholas True. Computer vision based fire detection.

[17] Alexandra Tyukavina, Peter Potapov, Matthew C. Hansen, Amy H., Pickens, Stephen V., Stehman, Svetlana Turubanova, Diana Parker , Viviana Zalles , André Lima , Indrani Kommareddy, Xiao-Peng Song , Lei Wang , and Nancy Harris. Forest fires concequenses. In *Global Trends of Forest Loss Due to Fire From 2001 to 2019*, page 1, 2022.

[18] Guang Xu and Xu Zhong. Real-time wildfire detection and tracking in australia using geostationary satellite: Himawari-8. In *Remote Sensing Letters*, 2017.