



Capstone Project Phase B

Dynamic Real Time Target Detection and Liquidation with Drones

Project number: 21-2-D-17

Supervisors:

Dr. Elena Ravve

Dr. Dan Lemberg

Alon Barenboim 206008179 Alon.Barenboim@e.braude.ac.il

Alexey Smirnov 324618107 Alexey.Smirnov@e.braude.ac.il

Abstract. *The modern warfare has changed rapidly over the last few decades, with the expanding usage of UAV (Unmanned Aerial Vehicle)'s and drones instead of conventional manned military planes for various situations. This is the seventh project in a series of capstone projects. In the previous projects, the students tried to achieve POC (Proof Of Concept) in Camouflaged and Aerostats object detection using Machine Learning algorithms in a static images or short videos. Two other teams are in their second phase of their projects: 1) Real Time Camouflage Detection with Drones (Mohamed Aboraya, Razie Fadool), 2) Real Time Detection of Aerostats with Drones (Fatina Sirhan, Shifaa Salih), 3) Real Time Target Liquidation with Drones (Soaad Amer, Omar Hallomi). In our project, we tried to integrate all the previous and new products of all the teams into a complete system, installed on a drone, that should be capable to detect and notify about aerostats and camouflaged targets in the field, as well as liquidate them using a high powered laser beam.*

Keywords: *camouflage, camouflage detection, clustering, sklearn SVM, Haralick features, sub-images (window).*

1. INTRODUCTION

1.1. Organization Of The Paper

1.2. What Was Done In The Previous Projects?

In the previous project, different topics were tackled:

- (1) The first project was made by Rani Halabi and Samir Kinaan in 2019. The students implemented an algorithm that successfully identified camouflaged objects in a static image using linear SVM (Support Vector Machine) method and Haralick algorithm.

- (2) The second project in this series was of Eden Kisliankov and Nael Halabi. They tried to implement a real time dynamic camouflaged objects detection algorithm using python. Their implementation improved the processing time of one image from 17 seconds to 1.8 seconds, which is a significant improvement, but not enough for real time stream processing.
- (3) Dor Dahan and Hai Sidi were the third team to continue the project. They created a database that contains big number of camouflaged objects in images, and trained the machine for successful SVM classification.
- (4) In Soaad Amer's and Omar Hallomi's project, the students implemented the laser mechanism and the corresponding software. Their implementation consists of a receiver sensor, a moving mechanism that directs the laser component to the target's coordinates, and a microcontroller that runs their software.
- (5) In Mohamed Aboraya's and Razie Fadool's project, the camouflaged targets detection should be implemented. The students tried different approaches to tackle this problem, like color-recognition and texture-recognition, in an attempt to make the process faster and more reliable in real time than the products of the projects they continuing.
- (6) In Fatina Sirhan's and Shifaa Saliha's project, they tried to implement real-time detection of aerostats. The drone should detect in real time, calculate where to aim and return their coordinates of the target.

1.3. What Did Go Well In The Previous Projects?

In Rani Halabi and Samir Kinaan project, the students succeeded to build an algorithm implement it so it can identify a camouflage network in an image by using linear SVM algorithm for clustering, assisted by Haralick algorithm to extract features from the sub-image.

In Eden Kisliankov and Nael Halabi project their algorithm succeeded to recognize the camouflaged object. They also optimized the algorithm and speeded up the runtime of the algorithm from 17 seconds per image, and it takes 1.8 seconds to process one second video.

In Dor Dahan and Haim (Hai) Sidi project, they succeed to build a database that contains the textures to compare with as well as learn the machine for SVM classification.

In Soaad Amer's and Omar Hallomi's project, the students built mechanism that holds and aims the laser and programed the Arduino controller. With drone's and target's locations as parameters, it aimed the laser correctly.

In Mohamed Aboraya's and Razie Fadool's project, their algorithm runs on recorded video from drone's flight and successfully detects camouflaged in real time 24 fps and detected in 4 seconds.

In Fatina Sirhan's and Shifaa Saliha's project, their algorithm successfully detects aerostats from a video in 5 fps but in limited cases only.

1.4. What Didn't Go Well In The Previous Projects?

In Rani Halabi and Samir Kinaan project, the conjecture was to match the relevant cluster to a new given image; however, it didn't work as they expected. They tried several algorithms and distances and it didn't provide them with stable results.

They tried the following algorithms and distances:

- K-Means - for detecting a camouflage in the image by dividing the image to sub-images and classifying each sub-image to the relevant class.
 - Silhouette - for finding the most relevant number of clusters (k) for the k-Means algorithm.
 - PAM (Partition Around Medoids) - a method found to be more accurate and robust than the K-Means algorithm.
- Geodesic Distance for calculating how similar the sub-image is to another one.

In Eden Kisliankov and Nael Halabi project, they tried to implement dynamic camouflage detection in Python unsuccessfully. Because the implementation was in Python they weren't able to achieve the necessary performance level to make it work in real time. Python is not ideal for real time because it is a heavily dynamic language — which makes it too slow and unpredictable for our usage.

In Fatina Sirhan's and Shifaa Saliha's project, their algorithm did not provide real-time detection on the Graphics Processing Unit (GPU). Although they did test the algorithm on a high-end GPU it worked in 5 fps what wasn't good enough.

1.5. What Are The Challenges Of Our Project?

Algorithmic challenges:

- 1) The algorithms that will be used in the final product may not be the ones that the previous teams decided to implement, as the current implementation is yet to be fully tested in real conditions and may not answer our requirements.
- 2) Combining different algorithms to work simultaneously in one complete system may require massive changes to the original implementation and could be a big challenge.

Software engineer challenge:

- 1) The integration of different programs and diverse programming styles is expected to be a tough challenge, especially, taking in account the real-time manner of the software.
- 2) Failures are not accepted by any mean in the final product, as any mistake, no matter how "negligible" it is, could cost in human lives and wide consequences in such a complex and lethal product. As a result, the final program has to be as accurate, widely tested and bug free as it can be, as well as secured and invulnerable to hacking and foreign control.

Hardware challenges:

- 1) Due to Corona virus worldwide lockdown, not all components were supplied; some of them were in fact simulated or emulated by software.
- 2) The other teams working alongside us may create a working mechanism that controls an effective laser component installed on a drone successfully. However, in order to integrate it in a complete system that can also detect different objects, we also need to learn and understand their work.

Technical challenges:

- 1) We are not familiar with any of the involved hardware: Drones, cameras and different sensors, small computation units and so on.

1.6. How Did We Plan To Overcome The Problems And What Was The Reality?

Algorithmic challenges:

- 1) *The algorithms that will be used in the final product may not be the ones that the previous teams decided to implement, as the current implementation is yet to be fully tested in real conditions and may not answer our requirements.*

If the scenario of the previous algorithms being irrelevant to the final product would come reality, we planned on trying to find alternatives in different platforms that are possible to implement in short period of time.

The implementation of algorithms, which the previous groups provided, worked on a recorded video file and not a video stream. Working on a recorded file, the implementation preloaded

the video, which makes it easier for the algorithm to run on it. While running the algorithm on a video stream, the implementation receives the video not as a whole video but by parts, on which the algorithm works one part after another. The work that was done before giving each frame to the algorithm can't take as long as when working with a recorded file. So, of course we had to work to change what was done before and make sure that the algorithm runs in real time.

At the beginning, the aerostat detection algorithm of the previous group was worse than camouflage detection, detection in 5 fps compared to 24 fps. Therefore we focused on the aerostat detection algorithm. We retrained the algorithm after some changes, and it detected the aerostat from the same video the previous group used for tests. It detected in a higher rate 10fps, and on video stream with a little bit high rate but with a noticeable delay. After seeing that the implemented improvements are not enough, we tried to work with other algorithms that are on the web. We achieved detection in a rate of 30 [fps](#) in average. The drone is capable of streaming at [24/25/30/48/50/60](#) [fps](#). So, if the [drones transmit](#) rate is set as 24 fps, we can provide real time processing. The transmission rate can be set in the drone's settings menu in the Smart Remote Controller.

- 2) *Combining different algorithms to work simultaneously in one complete system may require massive changes to the original implementation and could be a big challenge.*

In order to preserve the functionality of any algorithm individually, we planned on making the necessary changes to the implementations and code so all the previous outcomes would work simultaneously, and we would design a comprehensive testing process to make sure that all the algorithms were integrated successfully, and its output is accurate and expected.

To make those algorithms work together, they need to work well enough separately. We worked and tested the algorithms. Each detection algorithm needs to detect a target and return its coordinates. We moved the drone and the targets and compared the results to the expected coordinates. With what components that we have we could and tested the algorithms with the aiming code and mechanism. With these tests we could see if the aim was on target.

Software Engineer Challenge:

- 1) *The integration of different programs and diverse programming styles is expected to be a tough challenge, especially, taking in account the real-time manner of the software.*

We planned on learning what the previous teams had created and why they chose this path to tackle the challenges they faced with, basically from scratch, as our product is expected to be the final product of this long series of projects and their products. We assumed that as soon as we know any aspects of the previous work that has been done, we would be able to modify and/or use all the previous outcomes more easily and successfully.

We learned from previous groups about their work. Throughout the work of the previous groups on their project, we joined their meetings with the supervisors. Also, we asked them about some parts in their work.

After testing the algorithms themselves, we understood what we can use, what we should improve and what we should remove or replace.

- With the aiming algorithm there were no problems. We even tested it already in previous semester.

- The camouflage detecting worked well, 24 fps on a video. Unfortunately, we didn't succeed in make it work with a video stream at all.
- However, with the aerostat detection we knew that it worked only in 5 fps and could understand why though only through working hands on. The previous group worked with a recorded video, where the implementation queued the video in the GPU. This is also the reason why it worked faster with a better GPU.

2) *Failures are not accepted by any mean in the final product, as any mistake, no matter how "negligible" it is, could cost in human lives and wide consequences in such a complex and lethal product. As a result, the final program has to be as accurate, widely tested and bug free as it can be, as well as secured and invulnerable to hacking and foreign control.*

Even though we expect the previous outcomes to be at least basically secured, as this is also one of the considerations they had to take into account in their project, we still expect the security and testing be a major part in our project as we need to take the overall security and stability of the system in to account, and they didn't. We planned to expend our knowledge in those fields by learning and consulting with our supervisor and colleagues, and implement multiple layers of protection. In order to ensure that the system will be bug\failure free, we'll design comprehensive tests to check every aspect of the system.

Thankfully our drone isn't armed with something more serious than a laser. We worked hard to make the detection fast and without miss categorizing targets. After each major change we tested the algorithm with series of tests. The categorization was good due to work that the previous groups did. They made excellent training data that we used to retrain the algorithms when needed. However, for the correct categorization to be useful it must be quick, at least 24 fps. Therefore we worked hard to make the detection fast, to be considered real time detection.

Hardware challenges:

1) *Due to Corona virus worldwide lockdown, not all components were supplied; some of them were in fact simulated or emulated by software.*

Due of the fact, it was hard to guess how it would work out in the end. Ideally, the improvement of the world's state, should allow the ordered components to arrive for the previous groups to work with them. In this scenario, we planned to help these groups to work with these components and learn to operate them themselves. In case these components would not arrive, we would work on the corresponding software in a way that it would be easy to replace the simulations/emulation with the hardware.

Unfortunately, we didn't manage to receive all needed parts. We learned to with previous groups how to work with part that did arrive and with those which didn't we tried to work around it. We don't have the transmitting device and could not test the whole system in flight. However, we still could receive a broadcast video from the drone. The algorithm runs on the broadcast and gives signals to parts that we connected directly to computer.

2) *The other teams working alongside us may create a working mechanism that controls an effective laser component installed on a drone successfully. However, in order to integrate it in a complete system that can also detect different objects, we also need to learn and understand their work in depth.*

We see several ways to solve this potential problem.

First, to mount the computational unit on the drone. To do so, we need to know the weight of the unit (there are several options) and the additional weight that the drone can handle after the liquidation system is attached to the drone.

Second, on the ground. In this situation, there is distance between the drone and the computational unit that may cause a delay.

Why should it work?

There is non-trivial risk that it will not work. We still don't know if these methods or even one of them will work. Under this base assumption, we defined the full system as the main purpose of our project.

Expected difficulties and limitations

One of the big limitations is that we cannot mount any kind of computational unit on the drone because of the weight it can handle. Another significant challenge is to have real-time processing and communication with the computational unit that is on the ground.

In the drone's description it's given the maximal weight it's able to lift and fly with. Summing the weights from descriptions of the components that supposed to be on the drone, the drone should be able to fly with them without a problem. But we could not attach the parts because some parts didn't arrive and because the structure of this model of drone. This model has sensors at the bottom that should not be blocked or else it will refuse to take off from ground. So, the attachment of all the additional parts to the drone require a custom-made construction to be attached to drone, hold all the parts and that will not block the sensors. We can't confirm if the drone will fly with them.

Technical challenges:

- 1) *We are not familiar with any of the involved hardware: drone, Android based devices, small computation units and so on.*

We planned on overcoming this problem by reading articles on the internet about drones. Also, we planned to go to college and use the drone there and see how it takes the photos and how to control it. Moreover, there are some groups worked with the drone before us, and one of them learned about it deeply so we can ask them.

Unsurprisingly there are many people on the internet who use and develop on Android, Arduino and work with this model of the drone. There are many tutorials and articles explaining how to use these devices and how to develop on.

Also, in the first half of the project, in the previous semester we learned a lot from previous groups about the work with these components.

2. BACKGROUND AND RELATED WORK

2.1. Drone

Drone is an aircraft without a human pilot on board. Unmanned Aerial Vehicle (UAV) is a component of an Unmanned Aircraft System (UAS), which include a UAV, a ground-based controller,

and a system of communications between the two. The flight of UAVs may operate with various degrees of autonomy: either under remote control by a human operator or autonomously by onboard computers referred to as an autopilot.

2.1.1.1. DJI Mavic Pro 2 Drone: technical characteristics

We use DJI Mavic Pro 2 Drone, see Fig 1. The relevant features are summarized in Table 1:



Fig. 1 Small-form cameras.



Fig. 2 Folded DJI Mavic 2.
(Pictures taken from the company's website)

Mavic 2 Pro Camera	Specification
ISO Range ISO – International Organization for Standardization. The ISO number is the value of sensitivity. Sensitivity – sensitivity is a measure of the camera's ability to capture light.	Video: 100-6400
	Photo: 100-3200
Video Recording Modes	Display Standard: 4K Resolution: 3840×2160 (Pixels) Frame per second options: 24/25/30p (FPS – Frame Per Second)
	Display Standard: 2.7K Resolution: 2688x1512 (Pixels) Frame per second options: 24/25/30/48/50/60p (FPS – Frame Per Second)

	Display Standard: FHD -Full High Definition Resolution: 1920×1080 (pixels) Frame per second options: 24/25/30/48/50/60/120p (FPS - Frame Per Second)
Sensor	1” CMOS Effective Pixels: 20 million

Table 1. DJI Mavic 2 Pro Camera Specification.

The DJI (Da-Jiang Innovations) Smart Controller (see Fig.3) has a 5.5-inch built-in screen that displays clear, bright images including an HDMI (High-Definition Multimedia Interface) output port that is designed to work with the DJI Mavic 2.



Fig. 3 The DJI Smart Controller

DJI Smart Controller	Specification
Max Transmission Distance (unobstructed, free of Interference)	2.400-2.4835 GHz: 8 km (FCC), 4 km (CE), 4 km (SRRC), 4 km (MIC) 5.725-5.850 GHz: 8 km (FCC), 2 km (CE), 5 km (SRRC)
Transmitter Power (EIRP)	2.400-2.4835 GHz: 25.5 dBm (FCC), 18.5 dBm (CE), 19 dBm (SRRC), 18.5 dBm (MIC) 5.725-5.850 GHz: 25.5 dBm (FCC), 12.5 dBm (CE), 18.5 dBm (SRRC)
Battery	18650 Li-ion (5000 mAh @ 7.2 V)
Rated Power	15W

Working Time	2.5 hours
Video Output Port	HDMI Port

Table.2 DJI smart controller specification

The DJI Mavic Pro 2 drone comes equipped with the all-new Hasselblad L1D-20c camera, which possesses Hasselblad's unique Hasselblad Natural Color Solution (HNCS) which have 20-megapixel. The transmitter located in the drone, sends radio signals to the video receiver located in the drone controller from there we will stream the video to the user's laptop with the help of an external video card. The external video card will transfer the video from the drone's controller via the HDMI output to the USB input in the laptop (see Fig.4).

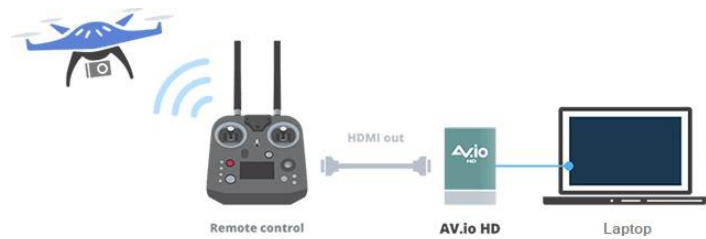




Fig. 4: Streaming the video from the drone to a laptop.

The DJI Smart Controller is better than the default controller. We choose this controller for having an HDMI output port. Via the Smart Controller, we will connect the drone's controller to the AV.io video capture card using an HDMI cable as in input, the card provides a USB output that we can connect to the portable computation (see Fig.4).

External capture card	video	Speciation
AV.io 4K		Input connector: HDMI max resolution: 4096×2160 Capture latency: Near-zero price: 550\$  <i>Fig.5 AV.io HD.cf[17]</i>
AV.io HD		Input connector: DVI and adapter for HDMI max resolution: 1920×1200 Capture latency: Near-zero price: 400\$  <i>Fig.6 AV.io HD.cf[15]</i>


StarTech	Input connector: HDMI max resolution: 1920×1200 Capture latency: 5-10 frames price: 100\$	
----------	--	---

Fig.7 StarTech.cf[18]

Table.3 External video capture card specification

An external video capture card. We need this device as part of the drone Smart Controller to be able to stream the drone video to the portable computation unit (see Fig.8). It is not possible to stream the video from the drone controller directly to the PC because both the drone controller and the PC have HDMI outputs and no HDMI input. Therefore, when we are using external video capture card, we can convert the HDMI output of the drone controller to a USB input which goes into the PC. Here, we show a table of devices and their specifications that may satisfy our needs (see Table.3).

Our project is a real-time project, that's why the latency of the external video card should be near-zero (0-2 frames latency). Moreover, we need a HDMI input without the use of any extra adapter, if we choose to use AV.io HD we must use an external adapter for HDMI.

After many researches we choose to use AV.io 4K as our external video card. The specifications of the AV.io 4K is rather given in Table.4



Fig. 8 AV.io HD, cf. [15]

Interface	USB 3.0, USB 2.0
OS drivers	UVC and UAC device
Dimensions	3.54" × 2.36" × 0.91" (90 mm × 60 mm × 23 mm)
Connectors	DVI-I (integrated, digital & analog), USB 3.0 B-Type connector
Input	HDMI (including audio), DVI, VGA
Input resolution	Up to 1920×1200
Capture latency	Near-zero. However, third-party applications may contribute to capture delay.
HDMI Audio (input)	16-bit and 24-bit PCM encoded audio at 32 kHz, 44.1 kHz, and 48 kHz sampling rates
OS Support	Windows 7, Windows 8.1, Windows 10, Mac OS X 10.10 and up, Linux distribution with kernel 3.5.0 or higher supported

Table .4 AV.io specification table

DJI Mavic Pro 2 Drone has a transmission system that allows the user to transmit radio signals to long range (8 KM).

Advantages: We do not need any internet connection. Moreover, this method ensures real-time computation.

Disadvantages: we need a portable computation unit, which must have a USB port, say a laptop. Also, we need to use an external video capture card. In addition, we must make sure that the portable computation unit should be placed in a two-meter radius because of the HDMI cable length.

We will list the minimal requirements for the laptop in the following Table 5, as listed in the AV.io 4K technical specifications, cf. [19].

	Specification
CPU	Intel core i7-7700 / Amd Ryzen 7 2700X
GPU	Nvidia Gtx 1050ti
Ram	16 Gb
Laptop battery	5500 mAh
Operating System	Windows 10
Ports	1X USB 2.0 Port

Table .5 Laptop requirements

2.2. Camouflage Detection

Camouflage related work can be divided into two big classes. The first is the camouflage assessment and design. The second is camouflage detection. Camouflage Identification Systems (CIS) or Decamouflaging are use in order to reveal the target object from its background, means discriminating foreground object from camouflaged image. There are many potential applications of CISs like discriminating enemies in war field, detecting defects in product during manufacturing, identifying duplicate products during logistics, etc. So, concept of Decamouflaging is how to detect specific texture to identify from the provided background.

2.2.1. Color Based Algorithm

An RGB image can be viewed as three different images (a red scale image, a green scale image and a blue scale image) stacked on top of each other. An RGB image is basically a $M \times N \times 3$ array of color pixel, where each color pixel is associated with three values which correspond to red, blue and green color component of RGB image at a specified spatial location. So, the color of any pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location.

Determining the strengths of the different colorants is pleasant by placing the integers between 0 and 255 for each of the variables R, G and B. 0 means not illuminating the pixel at all, and 255 is a full-color illumination symbol.

It is acceptable to translate numbers to the hexadecimal base (base 16), where only 2 characters are needed to represent numbers in the above range. In this way, each of the three primary colors (RGB) assigns two characters from left to right.

In each color the range is from 00 to FF (from 0 to 255 in decimal), which represent influence of the same color. Because each component can create 256 different shades, the number of different colors that can be represented in this way is 256 with the power of three which are 16,777,216 different colors.

To specify a value for grayscale, use the following: 0 means black, 255 means white. In between, every other number — 50, 87, 162, 209, and so on — is a shade of gray ranging from black to white.

Color images are represented as three dimensional arrays, a collection of three two dimensional arrays, one each for red, green, and blue channels. There are lots of ways to convert RGB image to grayscale image. Here you are some of the methods: cf. [3], [4]:

- Simple averaging – We take the average of three colors. Since it is an RGB image, it means that we have to add R to G and B and then divide the sum by 3 to get the desired grayscale image, by the following formula: $\text{Gray} = (R+G+B)/3$
- Weighted average - Averages the values, but it forms a weighted average to account for human perception by the following formulas: $\text{Gray} = 0.3R + 0.59G + 0.11B$ or $\text{Gray} = 0.2126R + 0.7152G + 0.0722B$ or $\text{Gray} = 0.299R + 0.587G + 0.114B$
- Desaturation - Averages the most prominent and least prominent colors by the following formula: $\text{Gray} = (\min(R,G,B) + \max(R,G,B))/2$
- Custom number of gray shades - Usually a grayscale image has intensities that range between 0 to 255. If one wants to reduce the number of shades, one can divide the interval $[0, 255]$ in a number of subintervals equal with the number of grey shades that we want to use. Let $p < 256$ be the number of shades we want to use. We consider the intervals:

$[\alpha_0 = 0, \alpha_1), [\alpha_1, \alpha_2), \dots, [\alpha_{p-1}, \alpha_p = 255], \alpha_i - \alpha_{i-1} = 255/p$. All the pixels with intensities in interval $[[\alpha]_{(i-1)}, \alpha_i)$ will be set to a common value (the middle point of the interval, for example). If we have a color image then, first, compute one of the weighted average of the color components or simple average and finally apply the algorithm described below.

The algorithm:

Algorithm convention:

NumberOfShades will be the number of shades we want to use. This number is determined by the user, he chooses how many shades he wants to have

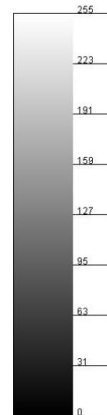


Fig. 9 Grayscale range

in his image, based on his needs. The possible values are 1 to 256 and the default value is 256.

Conversion Factor is the number or formula you need to convert a measurement in one set of units to the same measurement in another set of units. In our case the default value of Conversion Factor will be 1.

Custom number of gray shades algorithm:

```
ConversionFactor = 255 / (NumberOfShades - 1);
AverageValue = (Red + Green + Blue) / 3;
```

```
Gray = Integer((AverageValue / ConversionFactor) + 0.5) *
ConversionFactor;
```

- Custom number of grey shades with error-diffusion dithering - We explain error diffusion using an example. Consider the case of transforming a grayscale image into a binary one. The first pixel in the image is dark grey, assume it has 65 intensity, being closer to 0 (black) than to 255 (pure white) the pixel will be assigned to 0. Error diffusion works by spreading the error of each calculation to neighboring unvisited pixels. If it finds a pixel of 85 gray, it too determines that 85 is closer to 0 than to 255 and so it would make the pixel black. The error diffusion algorithm makes note of the “error” in its conversion, specifically, that the first gray pixel we have forced to black was actually 65 steps away from black. When it moves to the next pixel, the error diffusion algorithm adds the error of the previous pixel to the current pixel. If the next pixel is 85 gray, instead of simply forcing that to black as well, the algorithm adds the error of 65 from the previous pixel. This results in a value of 150, which is actually closer to 255, and thus closer to white. The Floyd-Steinberg Dithering algorithm is using the following diffusion mask:

$$\begin{array}{ccc} \# & * & \frac{7}{16} \\ \frac{3}{16} & \frac{5}{16} & \frac{1}{16} \end{array}$$

The * marks the position of the currently processed. The error is diffused only on pixels that were not already visited (the pixel marked # is left unchanged, because it was already processed).

Basic Floyd-Steinberg dithering algorithm:

```
I = input image
OutputImage = 0;
for I = 1 to m
    for j = 0 to n
        OutputImage[i,j] = NearestColor(I[i,j]);
        err = I[i,j] - OutputImage[i,j];
        I[i,j+1] += err * (7/16);
        I[i+1,j-1] += err * (3/16);
        I[i+1,j] += err * (5/16);
        I[i+1,j+1] += err * (1/16);
    end for j
end for i
```

- Decomposition –
 - Maximum by the following formula: $\text{Gray} = \max(R,G,B)$
 - Minimum by the following formula: $\text{Gray} = \min(R,G,B)$

At the beginning we will try to convert with the simple averaging method, because it's the simplest one. In this method we lose some information regarding the image. So, if it will not work then we will try one of the other methods that we described above. If none of the methods above will work, we will try other approaches.

2.2.2. Machine learning

Machine Learning (ML) is the scientific study of algorithms and models that computer systems use to progressively improve their performance. ML algorithms built a mathematical model of sample data known as “training data”, and then perform a test on the “test data” based on this “training data”. ML algorithms are built in order to make prediction or decisions without being explicitly programmed to perform the task. The ML uses have grown in the last few years, because of the potential that it has.

2.2.3 Support vector machine (SVM):

Input:

$N \times M$ dimension matrix, cf. [10], where N is the size of each vector (so there is M vectors: x_1, x_2, \dots, x_m), and another additional array in M size for the vector classification. SVM can be applied to any kind of vectors, which encode any kind of data. The main point is a definition of distance between the vectors. This means that in order to leverage the power of SVM texts, graphs, where the data have to be transformed into vectors.

Output:

Array in M size of weights w (w_i), one for each vector and scalar $b \in \mathbb{R}$.

SVM is a supervised learning model, which means that the programmer needs to specify about the training data, this step is called labelling that training data. Using the training data, the SVM builds a decision line that divide the space into 2 parts. When we input a new input to the system, using this line the model decides to which part it belongs.

In fact, cf. [10]:

We want to find the "maximum-margin hyperplane" that divides the group of points \vec{x}_i for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point \vec{x}_i from either group is maximized. Any hyperplane can be written as the set of points \vec{x} satisfying: $\vec{w} \cdot \vec{x} - b = 0$, where \vec{w} is the normal vector to the hyperplane.

The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector \vec{w} .

If the training data is linearly separable (linear separability is a property of two sets of points. This is most easily visualized in two dimensions by thinking of one set of points as being colored blue and the other set of points as being colored red. These two sets are linearly separable if there exists at least one line in the plane with all of the blue points on one side of the line and all the red points on the other side), we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalized or standardized dataset, these hyperplanes can be described by the equations:

$$\vec{w} \cdot \vec{x} - b = 1.$$

$$\vec{w} \cdot \vec{x} - b = -1.$$

Geometrically, the distance between these two hyperplanes is $\frac{2}{\|\vec{w}\|}$, so to maximize the distance

between the planes we want to minimize $\|\vec{w}\|$. We also have to prevent data points from falling into the margin, we add the following constraint: for each "i" either:

$$\vec{w} \cdot \vec{x}_i - b \geq 1, \text{ if } y_i = 1.$$

$$\vec{w} \cdot \vec{x}_i - b \leq -1, \text{ if } y_i = -1.$$

These constraints state that each data point must lie on the correct side of the margin.

This can be rewritten as:

$$y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, \text{ for all } 1 \leq i \leq n.$$

We can put this together to get the optimization problem:

Minimize $\|\vec{w}\|$ subject to $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$ for $i = 1 \dots n$.

The \vec{w} and b that solve this problem determine our classifier $\vec{x} = \text{sgn}(\vec{w} \cdot \vec{x} - b)$.

An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those \vec{x}_i that lie nearest to it. These \vec{x}_i are called support vectors.

Dual problem:

$$\begin{aligned} \max L_D(a_i) &= \sum_{i=1}^l a_i - \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l a_i a_j y_i y_j (x_i \cdot x_j) \\ \text{subject to: } &\sum_{i=1}^l a_i y_i = 0 \text{ \& } a_i \geq 0 \end{aligned}$$

If we take the derivative with respect to α and set it equal to zero, we get the following solution, so we can solve for α_i : $\sum_{i=1}^l \alpha_i y_i = 0$, $0 \leq \alpha_i \leq C$. Now knowing the α_i we can find the weights w for the maximal margin separating hyperplane: $w = \sum_{i=1}^l \alpha_i y_i x_i$. And now, after training and finding the w by this method. Given an unknown point u measured on features x_i we can classify it by looking at the sign of:

$$f(x) = w \cdot u + b = (\sum_{i=1}^l \alpha_i y_i x_i \cdot u) + b$$

2.2.4. Infrared Detectors

Infra-Red (IR) detectors can be used in a variety of fields. One of them is in the military, units used IR goggles to collect IR radiation particles by sensing the emitted heat and using these IR particles to create a picture.

2.2.5. Haralick Features

Robert Haralick is an expert in ML who described the important of textural features in image processing in his article in 1973 [5]. In image processing, the pictorial information is represented as a function of two variables (x, y). The image in its digital form is usually stored in the computer as a two-dimensional. Each feature is a mathematical formula and calculated using these two variables.

There are 13 features which he relies on in his article:

- 1.1.1. Angular Second Moment
- 1.1.2. Contrast
- 1.1.3. Correlation
- 1.1.4. Sum of Squares: Variance
- 1.1.5. Inverse Difference Moment
- 1.1.6. Sum Average
- 1.1.7. Sum Variance
- 1.1.8. Sum Entropy
- 1.1.9. Entropy
- 1.1.10. Difference Variance
- 1.1.11. Difference Entropy
- 1.1.12. Information Measures of Correlation
- 1.1.13. Maximal Correlation Coefficient

For each one of these features, Haralick invent an equation based on **co-occurrence** matrix which is statistical computed from a **gray scale** image, which is a 2D matrix. The reason, why the **co-occurrence** matrix is so widely used in image analysis, is that it represents characteristics of the texture in a given region. Haralick features are statistics defined to emphasize certain texture properties. The co-occurrence matrix consists of numbers, which are counts of co-occurrences of the same gray-scale color (intensity) in two pixels separated by oriented separation vector. Number of intensity values has to be finite and relatively small, in order to have any co-occurrences in the co-occurrence matrix. Number of intensity value is called “quantization constant”, and denoted by q . Later, an extension for these features was invented for medical image analysis [6]. Each one of the above features “received an update” from a 2D matrix into 3D matrix.

How is it calculated?

As Haralick explained in his article [5], in 2D matrix, suppose we have a rectangle image, that has N_x resolution cells in horizontal direction and N_y resolution cells in vertical direction.

Let $L_x = \{1, 2, 3, \dots, N_x\}$ and $L_y = \{1, 2, 3, \dots, N_y\}$, these will be the spatial domains in both direction. The gray tone appearing in each resolution cell is quantized to N_g levels. Let $G = \{1, 2, 3, \dots, N_g\}$. Then, the set $L_y \times L_x$ is the set of resolution cells of the image ordered by their row-column designations. An essential component of our conceptual framework texture is measure or more precisely four closely related measures from which all of our texture's features are derived. These measures are arrays termed angular nearest neighbor gray tone spatial dependence matrices and to describe these arrays we must emphasize our notion of adjacent or nearest neighbor resolution cells themselves. We consider a resolution cell excluding those on the periphery of an image; it means we have eight nearest neighbor resolution cells (see Fig.10).

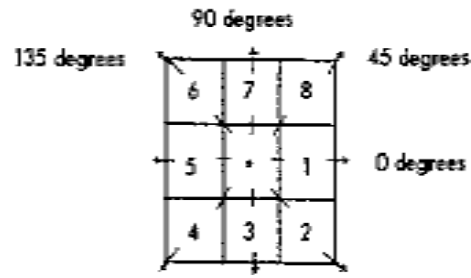


Fig. 10 Resolution cells 1 and 5 are 0° (horizontal) nearest neighbors to resolution cell *; resolution cells 2 and 6 are 135° nearest neighbors; resolution cells 3 and 7 are 90° nearest neighbors; and resolution cells 4 and 8 are 45° nearest neighbors to *.

(picture and explanation taken from Haralick article [5]).

Extended Haralick features

As explained in the 3D extension of Haralick texture features for medical image analysis article [6]. In 2D case, the co-occurrence is calculated for the whole image, in 3D case, in order to get better results for each region, the features are calculated for each pixel. As explained, it is possible to calculate this matrix on reasonable small window around every pixel. For 3D image, the intensity array P is defined by individual intensities $p_{\vec{v}}$, for $\vec{v} = (x, y, z)$. Same as in 2D case, $p_{\vec{v}}$ are quantized to q intensities. Co-occurrence matrix $C_{\vec{v}}^{\vec{s}}$ is defined by its components $C_{\vec{v}}^{\vec{s}}(i, j)$. The separation vector is $\vec{s} = (s1, s2, s3)$. Window $W(\vec{v})$ of the size $2w + 1$ around the voxel \vec{v} is defined as:

$$W(\vec{v}) = \{ \vec{u} \in I : \vec{u} \leq \vec{v} + \vec{w}; \vec{u} \geq \vec{v} - \vec{w} \} \quad (1)$$

where $\vec{w} = (w, w, w)$ and I is the whole image region. The co-occurrence matrix is defined like this:

$$C_{\vec{v}}^{\vec{s}}(i, j) = \frac{\text{card}\{ \vec{c} \in W(\vec{v}) : p_{\vec{c}} = i, p_{\vec{c} + \vec{s}} = j \}}{\text{card } W(\vec{v})} \quad (2)$$

Having the co-occurrence matrix C with components $c(i, j)$ calculated, Haralick features can be computed using formulas given below. At first, statistics have to be defined:

$$\mu_x = \sum_{i=1}^q \sum_{j=1}^q (i - 1) c(i, j) \quad (3)$$

$$\mu_y = \sum_{i=1}^q \sum_{j=1}^q (j - 1) c(i, j) \quad (4)$$

$$\text{var}_x = \sum_{i=1}^q \sum_{j=1}^q (i - 1 - \mu_y)^2 c(i, j) \quad (5)$$

$$\text{var}_y = \sum_{i=1}^q \sum_{j=1}^q (i - 1 - \mu_y)^2 c(i, j) \quad (6)$$

$$c_x(i) = \sum_{j=1}^q c(i, j) \quad (7)$$

$$c_y(j) = \sum_{i=1}^q c(i, j) \quad (8)$$

$$c_{x+y}(k) = \sum_{j=1}^q \sum_{i \in \{1,2,\dots,q\}; i+j=k-1} c(i, j) \quad (9)$$

$$c_{x-y}(k) = \sum_{j=1}^q \sum_{i \in \{1,2,\dots,q\}; |i-j|=k-1} c(i, j) \quad (10)$$

$$HXY = - \sum_{i=1}^q \sum_{j=1}^q c(i, j) \text{Log}(c(i, j)) \quad (11)$$

$$HX = - \sum_{i=1}^q c_x(i) \text{Log}(c_x(i)) \quad (12)$$

$$HY = - \sum_{j=1}^q c_y(j) \text{Log}(c_y(j)) \quad (13)$$

$$HXY1 = - \sum_{i=1}^q \sum_{j=1}^q c(i, j) \text{Log}(c_x(i) c_y(j)) \quad (14)$$

$$HXY2 = - \sum_{i=1}^q \sum_{j=1}^q c_x(i) c_y(j) \text{Log}(c_x(i) c_y(j)) \quad (15)$$

where $c_{x+y}(k)$ is defined for $k = 1, 2, \dots, 2q - 1$ and $c_{x-y}(k)$ for $k = 1, \dots, q$. Finally after defining these equations, we can define the new Haralick features formulas.

$$\text{uniformity of energy: } \sum_{i=1}^q \sum_{j=1}^q c(i, j)^2 \quad (16)$$

$$\text{texture correlation: } \sum_{i=1}^q \sum_{j=1}^q \frac{(i-1-\mu_x)(j-1-\mu_y)c(i, j)}{\sqrt{\text{var}_x \text{var}_y}} \quad (17)$$

$$\text{variance: } \sum_{i=1}^q \sum_{j=1}^q (i-1-\mu_x)^2 c(i, j) \quad (18)$$

$$\text{inverse difference moment: } \sum_{i=1}^q \sum_{j=1, \dots, q; j \neq i}^q \frac{c(i, j)}{|i-j|} \quad (19)$$

$$\text{sum average: } \sum_{k=1}^{2q-1} (k-1) c_{x+y}(k) \quad (20)$$

$$\text{maximum probability: } \max_{i,j=1,\dots,q} c(i, j) \quad (21)$$

$$\text{texture homogeneity: } \sum_{i=1}^q \sum_{j=1}^q \frac{c(i, j)}{1+|i-j|} \quad (22)$$

$$\text{sum entropy: } - \sum_{i=1}^{2q-1} c_{x+y}(k) \log(c_{x+y}(k)) \quad (23)$$

$$\text{entropy: } - \sum_{i=1}^q \sum_{j=1}^q c(i, j) \text{Log} c(i, j) \quad (24)$$

$$\text{difference entropy: } - \sum_{k=1}^q c_{x-y}(k) \log(c_{x-y}(k)) \quad (25)$$

$$\text{texture contrast: } \sum_{i=1}^q \sum_{j=1}^q |i, j| c(i, j) \quad (26)$$

$$\text{maximal correlation coefficient: } \sum_{k=1}^q \frac{c(i, k) c(k, j)}{c_x(i) c_y(j)} \quad (27)$$

2.2.6. Center Of Mass

According to [7], the terms "center of mass" and "center of gravity" are used synonymously in a uniform gravity field to represent the unique point in an object or system which can be used to describe the system's response to external forces and torques. An experimental method for locating the center of mass is to suspend the object from two locations and to drop plumb lines from the suspension points. The intersection of the two lines is the center of mass.

The shape of an object might already be mathematically determined, but it may be too complex to use a known formula. In this case, one can subdivide the complex shape into simpler, more elementary shapes, whose centers of mass are easy to find. If the total mass and center of mass can be determined for each area, then the center of mass of the whole is the weighted average of the centers. This method can even work for objects with holes, which can be accounted for as negative masses.

2.2.7. Optical Flow

Definition: according to [8], optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image.

Uses: Motion estimation and video compression have developed as a major aspect of optical flow research. While the optical flow field is superficially similar to a dense motion field derived from the techniques of motion estimation, optical flow is the study of not only the determination of the optical flow field itself, but also of its use in estimating the three-dimensional nature and structure of the scene, as well as the 3D motion of objects and the observer relative to the scene. Optical flow was used by robotics researchers in many areas such as: object detection and tracking, image dominant plane extraction, movement detection, robot navigation.

Optical flow sensor: An optical flow sensor is a vision sensor capable of measuring optical flow or visual motion and outputting a measurement based on optical flow. Various configurations of optical flow sensors exist. One configuration is an image sensor chip connected to a processor programmed to run an optical flow algorithm. Another configuration uses a vision chip, which is an integrated circuit having both the image sensor and the processor on the same die, allowing for a compact implementation.

Estimation: The optical flow methods try to calculate the motion between two image frames which are taken at times t and $t+\Delta t$ at every voxel position. These methods are called differential since they are based on local Taylor series approximations of the image signal; that is, they use partial derivatives with respect to the spatial and temporal coordinates.

Frames Motion Estimation:

Frames Motion Estimation

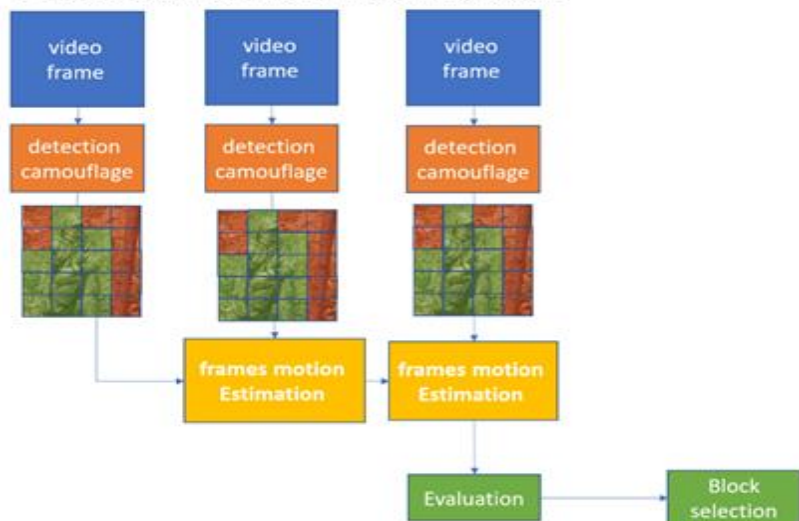


Fig. 11: Frames motion estimation

Motion estimation is the process of determining frames motion vectors that describe the transformation from one 2D image to another; usually from adjacent frames in a video sequence. Just a short reminder – our drone flies and shoots depending on the weather conditions, I mean it isn't static. Therefore, we need to balance the photo and find the transformation in order to find the same area between the different frames.

Illustration of Frames Motion Estimation:

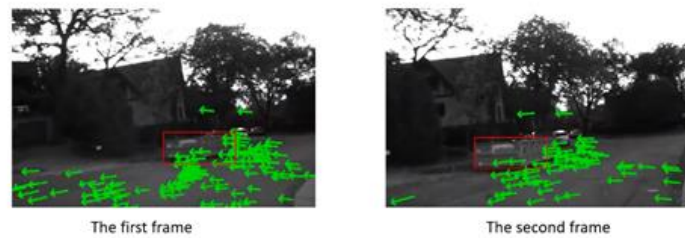


Fig. 12: Illustration of frames motion estimation

Here, we see motion estimation used by algorithm that is called **Optical flow**.

Optical Flow Algorithm

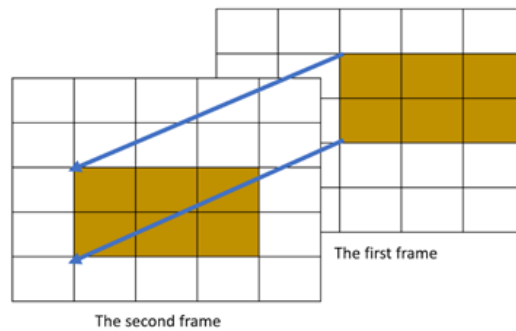


Fig. 13: Use an optical flow algorithm to calculate the transformation

Given:

$$(1) \quad I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

$$(2) \quad I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t + \text{higher order terms}$$

We use the Optical Flow Algorithm in order to calculate the transformation. We chose to use this algorithm because we assume there is no significant difference between the frames, and also the algorithm is based on Taylor series.

Here we are given two frames, and we need to calculate the transformation between them. After calculating, we know where the most similar areas between the frames are.

2.2.8. PX4FLOW:

An optical flow smart camera, the PX4FLOW (Optical Flow) Sensor is an open source and open hardware design of an optical flow sensor based on a machine vision CMOS image sensor for indoor and outdoor applications with very high light sensitivity. Optical flow is estimated on an ARM Cortex M4 microcontroller in real-time at 250 Hz update rate. Angular rate compensation with a gyroscope and distance scaling using an ultrasonic sensor are performed onboard. The system is designed for further extension and adaption and shown in-flight on a micro air vehicle.



The PX4FLOW (Optical Flow) Sensor is a specialized high resolution downward pointing camera module and a 3-axis gyro that uses the ground texture and visible features to determine aircraft ground velocity. Although the sensor may be supplied with a built-in Maxbotix LZ-EZ4 sonar to measure height [20].

Fig. 14: PX4FLOW optical flow smart camera

Specifications:

- 168 MHz Cortex M4F CPU (128 + 64 KB RAM)
- 752x480 MT9V034 image sensor, L3GD20 3D Gyro
- 16 mm M12 lens (IR block filter)
- Size 45.5 mm x 35mm
- Power consumption 115mA / 5V
- MT9V034 machine vision CMOS sensor with global shutter
- Optical flow processing at 4x4 binned image at 400 Hz
- Superior light sensitivity with 24x24 μm super-pixels
- Onboard 16bit gyroscope up to 2000°/s and 780 Hz update rate, default high precision-mode at 500°/s
- Onboard sonar input and mount for Maxbotix sonar sensors. (HRLV-EZ4 recommended, SparkFun Product Link)
- USB bootloader
- USB serial up to 921600 baud (including live camera view with mission planner program)

- USB power option
- Does fit the MatrixVision Bluefox MV mounting holes (camera center off-centered)

2.2.9. PIXHAWK:

PIXHAWK is a novel hardware and software system for micro air vehicles (MAV) that allows high-speed, low-latency onboard image processing. It uses up to four cameras in parallel on a miniature rotary wing platform. The MAV navigates based on onboard processed computer vision in GPS-denied in- and outdoor environments. It can process in parallel images and inertial measurement information from multiple cameras for multiple purposes (localization, pattern recognition, obstacle avoidance) by distributing the images on a central, low-latency image hub. Furthermore the system can utilize low-bandwidth radio links for communication and is designed and optimized to scale to swarm use. Experimental results show successful flight with a range of onboard computer vision algorithms, including localization, obstacle avoidance and pattern recognition.



Fig. 15: PIXHAWK device

The flow sensor is attached via UART to a PIXHAWK Cheetah quad rotor, which performed the flight and data logging onboard. The sensor sends ground speed estimates in the MAVLink protocol format to the onboard autopilot, where the individual measurements are rotated with the current heading and integrated to a position estimate in global coordinates[1 Flight Performance].

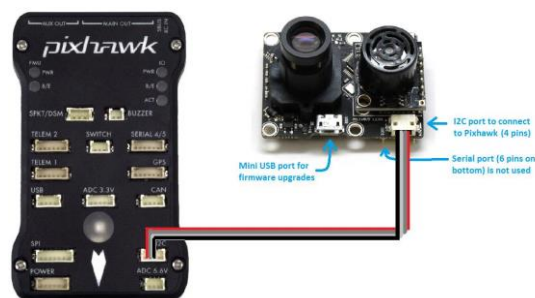


Fig. 16: Connection between PX4FLOW and Pixhawk hardware

2.2.10. Microcontroller:

A microcontroller is a small computer on a single metal-oxide-semiconductor integrated circuit chip. In modern terminology, it is like but less sophisticated than, a system on a chip, a SoC (System On a Chip) may include a microcontroller as one of its components. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR is flash, or PROM (Programmable Read Only Memory) also often included on-chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general-purpose applications consisting of various discrete chips.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys, and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed-signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. In the context of the internet of things, microcontrollers are an economical and popular means of data collection, sensing and actuating the physical world as edge devices, cf. [46].

2.1.12 Arduino:

We choose to work with Arduino microcontrollers because they have more advantages than other microcontrollers.

1- Ready to Use:

Arduino comes in a complete package form which includes the 5V regulator, a burner, an oscillator, a microcontroller, serial communications interface, and headers for the connections. We do not have to think about programmer connections for programming or any other interface. Just plug it into USB port of your computer and that is it.

2- Examples of codes:

Another big advantage of Arduino is its library of examples present inside the software of Arduino

3- Effortless functions:

During coding of Arduino, we will notice some functions, which make the life so easy. Another advantage of Arduino is its automatic unit conversion capability. We can say that during debugging we don't have to worry about the unit's conversions.

4- Large community:

There are many forums present on the internet in which people are talking about the Arduino. Engineers, hobbyists, and professionals are making their projects through Arduino. We can easily find help about everything. Moreover, the Arduino website itself explains each and every functions of Arduino. So, we should conclude the advantage of Arduino by saying that during working on different projects you just must worry about your innovative idea. The remaining will handle by Arduino itself, cf. [26].

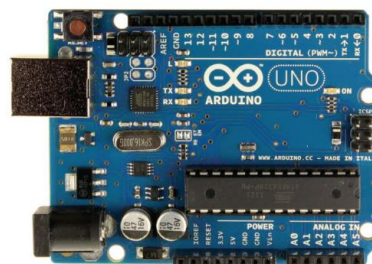


Fig. 17: Arduino device

Arduino is an open-source programmable circuit board that can be integrated into a wide variety of maker space projects both simple and complex. This board contains a microcontroller that can be programmed to sense and control objects in the physical world. By responding to sensors and inputs, the Arduino can interact with a large array of outputs such as LEDs, motors, and displays. Because of its flexibility and low cost, Arduino has become a very popular choice for makers and maker spaces looking to create interactive hardware projects cf. [45].

How to program Arduino:

Once the circuit has been created on the breadboard, we need to upload the program (known as a sketch) to the Arduino. The sketch is a set of instructions that tells the board what functions it needs to perform. An Arduino board can only hold and perform one sketch at a time. The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment. The software can be found at: <https://www.arduino.cc/en/Main/Software>.

Every Arduino sketch has two main parts to the program:

void setup () – Sets things up that must be done once and then do not happen again.

void loop () – Contains the instructions that get repeated over and over until the board is turned off.

We used a quick Arduino video to know how everything works together, cf. [45].

2.1.13. SiK Telemetry Radio :

In operation of any robotic aircraft, a human operator should always be able to intervene. The typical 30-100 m range of WiFi **does not** generalize to most outdoor applications, which make a communication architecture scaling down to radio modems also desirable for the off-board communication.

A *SiK Telemetry Radio* is one of the easiest ways to setup a telemetry connection between your Autopilot and a ground station. This article provides a basic user guide for how to connect and configure your radio.

[jDrones](#) and [RFDesign](#) offer *long-range SiK*-compatible telemetry radios. The radios provide reliable connectivity at greater than 5km ranges with normal antennas (and have been reported to achieve much greater ranges) [21] [25].



Fig. 18: Sik telemetry device for transmission radio

2.1.14. Mission planner :

Mission Planner [24] is a ground control station for Plane, Copter and Rover. It is compatible with Windows only. Mission Planner can be used as a configuration utility or as a dynamic control supplement for your autonomous vehicle. Here are just a few things you can do with Mission Planner:

- Load the firmware (the software) into the autopilot board (i.e. Pixhawk series) that controls your vehicle.
- Setup, configure, and tune your vehicle for optimum performance.
- Plan, save and load autonomous missions into you autopilot with simple point-and-click way-point entry on Google or other maps.
- Download and analyze mission logs created by your autopilot.
- Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.
- With appropriate telemetry hardware you can:
 - Monitor your vehicle's status while in operation.
 - Record telemetry logs which contain much more information the the on-board autopilot logs.
 - View and analyze the telemetry logs.
 - Operate your vehicle in FPV (first person view)

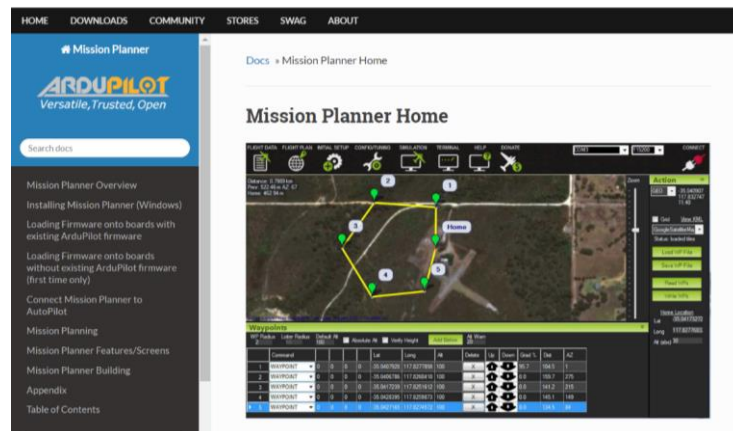


Fig. 19: Mission Planner Home Application

2.1.15. How the system works:

We actually mentioned a lot of devices, so we will now explain about the connection between the different things. We must always remember the goal of the project that is the extermination of a sniper on the ground.

To achieve this goal we must locate it using coordinates and transfer the coordinates to the unit on the ground and to the calculation unit of the laser. In our case it works this way: Pixhawk contains the coordinates, and from here they must be transmitted to two places:

2.1.15.1 The unit on the ground via radio transmission:

Radio broadcasting is the protocol for transmitting the image and the various parameters to a computer that is on the ground. All the various information and data will be transmitted via radio frequency, and the data will be displayed in Mission Planner Application.



Fig. 20: Antenna inside Pixhawk on the drone



Fig. 21: Antenna inside PC to get data

2.1.15.2 To Arduino (laser controller) device using a cable:

We need to pass the information of the coordinates inside PIXHAWK device to the laser unit Arduino, both are on the drone, very important to connect them because the mission of the laser is based on the coordinates inside the Pixhawk.

In other words, the Pixhawk transmits the information about the coordinates to the unit of calculation of the laser using a cable.



Fig. 22: 4 pin cable for the connection [23]

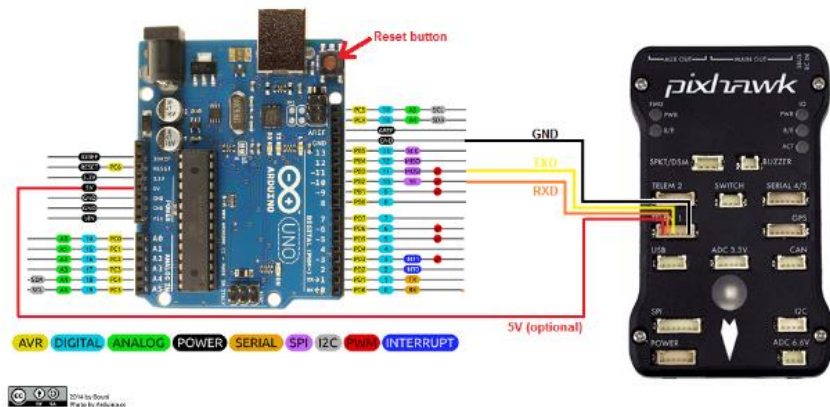


Fig. 23: 4 pin cable that connecting between Pixhawk and Arduino [22]

2.2 Aerostat detection

2.2.1 Aerostat

An aerostat is a craft that gains lift using a buoyant gas, such as helium or hydrogen, and therefore is lighter than air. All known field operational systems today use helium as their key “lifting” gas as it is non-flammable therefore safer. Beyond that, aerostat systems vary greatly in quality, durability, simplicity, size and carrying capacity, creating a range of possible payloads, coverage, operation and cost

2.2.2 Algorithmic Challenges

2.2.2.1 Color Based Algorithm:

An RGB (Red Green Blue) image can be viewed as three different images (a red scale image, a green scale image and a blue scale image) stacked on top of each other. An RGB image is basically a $M \times N \times 3$ array of color pixel, where each color pixel is associated with three values which correspond to red, blue, and green color component of RGB image at a specified spatial location. So, the color of any pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel’s location.

2.2.2.2 Convolutional Neural Network (CNN)

CNN, cf. [18]. is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to learn spatial hierarchies of features automatically and adaptively, from low- to high-level patterns.

CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers Fig. 25, and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can

hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and ground truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

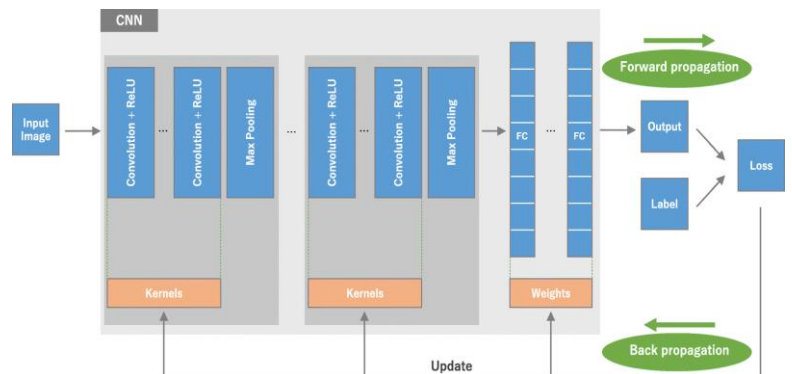


Fig 24. An overview of a convolutional neural network (CNN) architecture and the training process. cf. [18].

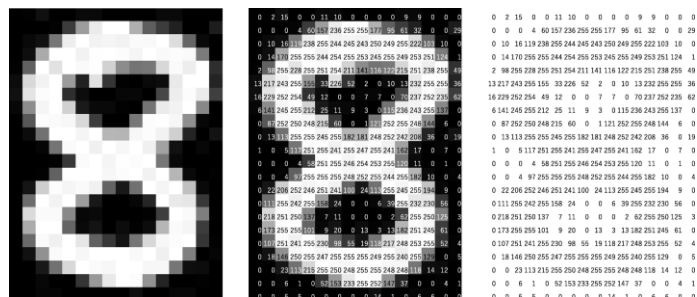


Fig 25. A computer sees an image as an array of numbers. cf. [18].

2.2.2.3 Using CNN in our project

Previous projects dealt with texture recognition because the main implementation was detection of camouflage. In our project, the balloons are not hidden. That is why we plan to use a different approach, which is CNN. In general, it is possible to identify the balloons by colors but in our case. However, in our particular case, we do not even need colors. We just need to teach our network what is the shape of a cluster of balloons.

The challenges:

1. Training the CNN.
2. This cluster might not capture our whole image. It means that we need to extract from the image a part that is the cluster.

How do we do that?

- a. How do we train the CNN?

We train the CNN to detect aerostats through sending at least tens of thousands of examples that show what an aerostat looks like in order to successfully identify the aerostat in the picture that is going to be sent to it. We searched for balloon images to our dataset on Flickr, cf. [20].

- b. How do we extract the relevant sub-picture?

First of all, we divide the taken image, by making an equal division in both the x-axis and the y-axis. Otherwise, if we divide differently, we deform the image and get different proportions. We will decide at the implementation stage what the more appropriate divisions are.

Let us explain our approach by the following real life example. On Fig. 30, you can see our drone. Now, we provide several pictures, taken from different distances, see Figs. (31-42). We expect that our drone should detect the balloon(s) when it is pretty far from them. It means that the balloons may appear in any part of the picture. The different divisions of the picture allow definition of the sub-rectangular (sub-square) with the balloons for identification.

Once we decide, how we divide the image, we get parts whose size is different from the size of the images that the CNN trained. The derived challenges are:

- a. We have lots of "small" sub-pictures of the taken picture from the drone and "big" pictures from the training set of CNN. We need to pump each of the "small" pictures to the size of the "big" ones.

To do this, we will enlarge the "small" picture to the size of "big" one, we are actually introducing empty spaces in the original base picture, from the image below, Fig. 26, an image with dimension ($w_1 = 4$, $h_1 = 4$) is to be enlarged to ($w_2 = 8$, $h_2 = 8$). The black pixels represent empty spaces where interpolation is needed.

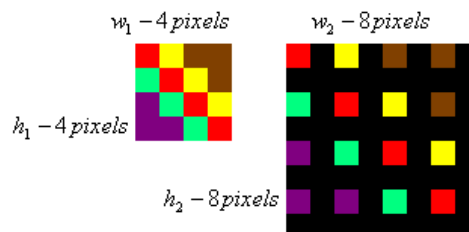


Fig 26. An image with dimension (4x4) is to be enlarged to (8x8).

- b. Now, we have all pictures of the same size. However, the pumping process produces unclear pictures.

To do this, we may use different techniques. One of them is Bilinear Interpolation.

Considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixels Takes a weighted average of these 4 pixels to arrive at the final interpolated values Results in smoother looking images than nearest neighborhood Needs of more processing time, Fig. 27.

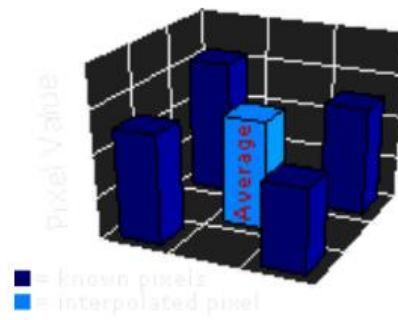


Fig 27. An average of 4 pixels

Example:

Original image:



Fig 28. An original sub-picture

After Bilinear Interpolation:



Fig 29. After Bicubic interpolation

Another one is FFT Based Scaling

Transform the images into a Fourier representation (2D frequency). For upscaling enlarge this Fourier representation by adding zeroes around the spectrum, for downscaling reduce the Fourier representation by removing lines and columns from the spectrum's outer edges. Transferring the image back into the 2D space domain will yield an image that was enlarged/downscaled accordingly - with the addition of new frequencies or removal of frequencies other than those which were too high for the new resolution anyway.

We will implement both of them and chose the best one for our particular case.



Fig 30. The drone



Fig 31. A balloon located one meter from the drone



Fig 32. A Cluster of balloons located one meter from the drone



Fig 33. A balloon located two meters from the drone



Fig 34. A cluster of balloons located two meters from the drone

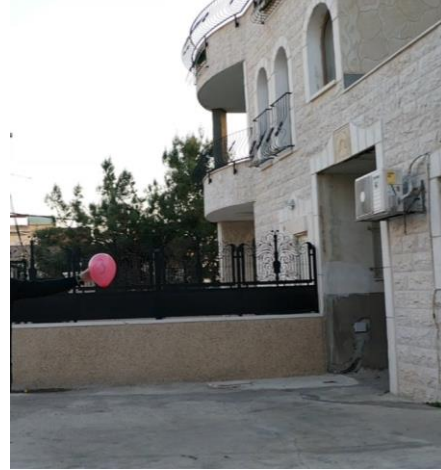


Fig 35. A Balloon located three meters from the drone



Fig 36. A Cluster of balloons located three meters from the drone



Fig 37. A Balloon located four meters from the drone



Fig 38. A cluster of balloons located four meters from the drone



Fig 39. A balloon located five meters from the drone



Fig 40. A Cluster of balloons located five meters from the drone



Fig 41. A balloon located six meters from the drone



Fig 42. A cluster of balloons located six meters from the drone

In order to identify the balloons, we should first go through several stages, let us explain them:

At the beginning of this process, an input image is broken down into pixels. If our image dimensions are $n \times m \times 3$ (n and m are the length and width of the CNN training pictures, the 3 refers to RGB values) it will represent in a form of a matrix ($n \times m$). Then, we take the $k \times k$ ($k < n$ and $k < m$) filter matrix (which we got from the training phase of CNN) and slide it over the complete image and along the way performs a mathematical operations that the end result of all this calculation is a feature map.

Since we trained our CNN algorithm on many images of balloons it already aware of the features of the balloons. So, it will be able to identify the balloons in the image that is going to be sent to it as input.

2.3 Target Liquidation

2.3.1 Laser

A laser is a device that emits light through a process of optical amplification based on the stimulated emission of electromagnetic radiation. The term "laser" originated as an acronym for "light amplification by stimulated emission of radiation". The first laser was built in 1960 by Theodore H. Maiman at Hughes Research Laboratories, based on theoretical work by Charles Hard Townes and Arthur Leonard Schawlow.

A laser differs from other sources of light in that it emits light which is coherent. Spatial coherence allows a laser to be focused to a tight spot, enabling applications such as laser cutting and lithography. Spatial coherence also allows a laser beam to stay narrow over great distances (collimation), enabling applications such as laser pointers and lidar. Lasers can also have high temporal coherence, which allows them to emit light with a very narrow spectrum.

Lasers are used in optical disk drives, laser printers, barcode scanners, DNA sequencing instruments, fiber-optic, semiconducting chip manufacturing (photolithography), and so on. They have been used for car headlamps on luxury cars, by using a blue laser and a phosphor to produce highly directional white light, cf. [19].



Fig 43: Red, blue, and green lasers, cf. [44].

- Green Light Laser:

Green light lasers are the most common and popular today, green laser light is ten to 50 times brighter than the red-light lasers. This means improved visibility of laser lines or dots, even during broad daylight and in direct sunlight. It can travel longer distances of up to three miles, which is why it is an essential in various industrial worksites. Its wavelength is 532nm and its power is 5mW to 130mW, cf. [20].

- Red Light Laser:

Red light lasers were once the standard but lost its supremacy down the road. Red lasers lack the power and distance possible in newer models. Red lasers use diodes, optics, and various electronic components. Its wavelength is 630-680nm and its power is 5mW and below, cf. [20].

- Blue Light Laser:

Blue light lasers have a shorter wavelength compared to red lasers, allowing for stronger focus or resolution in very fine structures in imaging applications. They are used in interferometers like compact discs, laser printers, digital photofinishing, data recording, and CD and DVD players. Its wavelength is 445nm and its power is 500mW, cf. [20].

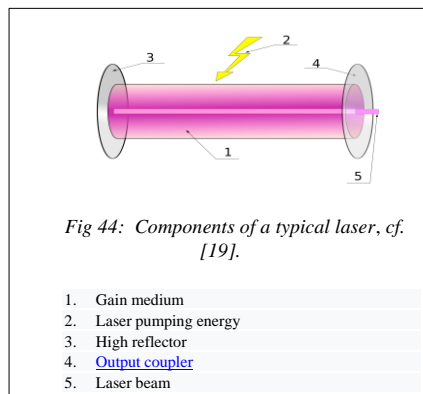
2.3.2 Design

A laser consists of a gain medium, a mechanism to energize it, and something to provide optical feedback. The gain medium is a material with properties that allow it to amplify light by way of stimulated emission. Light of a specific wavelength that passes through the gain medium is amplified (increases in power).

For the gain medium to amplify light, it needs to be supplied with energy in a process called pumping. The energy is typically supplied as an electric current or as light at a different wavelength. Pump light may be provided by a flash lamp or by another laser.

The most common type of laser uses feedback from an optical cavity—a pair of mirrors on either end of the gain medium. Light bounces back and forth between the mirrors, passing through the gain medium and being amplified each time. Typically, one of the two mirrors, the output coupler, is partially transparent. Some of the light escapes through this mirror. Depending on the design of the cavity (whether the mirrors are flat or curved), the light coming out of the laser may spread out or form a narrow beam. In analogy to electronic oscillators, this device is sometimes called a laser oscillator.

Some practical lasers contain additional elements that affect properties of the emitted light, such as the polarization, wavelength, and shape of the beam.



2.3.3 Types and operating principles

Lasers are classified into 4 types based on the type of laser medium used:

1. Solid-state laser

A solid-state laser is a laser that uses solid as a laser medium. In these lasers, glass or crystalline materials are used.

Ions are introduced as impurities into host material which can be a glass or crystalline. The process of adding impurities to the substance is called doping. Rare earth elements such as cerium (Ce), erbium (Eu), terbium (Tb) etc are most used as dopants.

Materials such as sapphire (Al_2O_3), neodymium-doped yttrium aluminum garnet (Nd:YAG), Neodymium-doped glass (Nd:glass) and ytterbium-doped glass are used as host materials for laser medium. Out of these, neodymium-doped yttrium aluminum garnet (Nd:YAG) is most commonly used.

The first solid-state laser was a ruby laser. It is still used in some applications. In this laser, a ruby crystal is used as a laser medium.

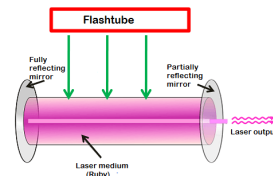


Fig 45: Global Solid-State Laser, cf. [22].

In solid-state lasers, light energy is used as pumping source. Light sources such as flashtube, flash lamps, arc lamps, or laser diodes are used to achieve pumping.

Semiconductor lasers do not belong to this category because these lasers are usually electrically pumped and involve different physical processes. cf. [22].

2. Gas laser

A gas laser is a laser in which an electric current is discharged through a gas inside the laser medium to produce laser light. In gas lasers, the laser medium is in the gaseous state.

Gas lasers are used in applications that require laser light with very high beam quality and long coherence lengths.

In gas laser, the laser medium or gain medium is made up of the mixture of gases. This mixture is packed up into a glass tube. The glass tube filled with the mixture of gases acts as an active medium or laser medium.

A gas laser is the first laser that works on the principle of converting electrical energy into light energy. It produces a laser light beam in the infrared region of the spectrum at $1.15\text{ }\mu\text{m}$.

Gas lasers are of different types: they are, Helium (He) – Neon (Ne) lasers, argon ion lasers, carbon dioxide lasers (CO₂ lasers), carbon monoxide lasers (CO lasers), excimer lasers, nitrogen lasers, hydrogen lasers, etc. The type of gas used to construct the laser medium can determine the lasers wavelength or efficiency, cf. [22].

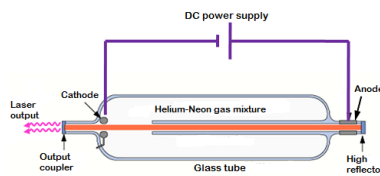


Fig 46: Gas laser, cf. [22].

3. Liquid laser

A liquid laser is a laser that uses the liquid as laser medium. In liquid lasers, light supplies energy to the laser medium.

A dye laser is an example of the liquid laser. A dye laser is a laser that uses an organic dye (liquid solution) as the laser medium.

A dye laser is made up of an organic dye mixed with a solvent. These lasers generate laser light from the excited energy states of organic dyes dissolved in liquid solvents. It produces laser light beam in the near ultraviolet (UV) to the near infrared (IR) region of the spectrum, cf. [22].

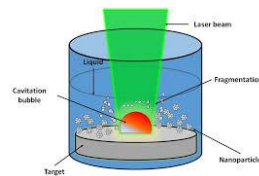


Fig 47: Laser ablation in liquid medium, cf. [42].

4. Semiconductor laser

Semiconductor lasers play an important role in our everyday life. These lasers are very cheap, compact size and consume low power. Semiconductor lasers are also known as laser diodes.

Semiconductor lasers are different from solid-state lasers. In solid-state lasers, light energy is used as the pump source whereas, in semiconductor lasers, electrical energy is used as the pump source.

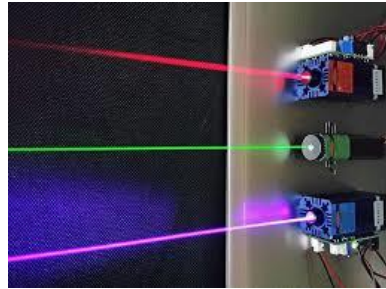


Fig 48: Semiconductor laser, cf. [43].

2.3.4 Uses

When lasers were invented in 1960, they were called "a solution looking for a problem". Since then, they have become ubiquitous, finding utility in thousands of highly varied applications in every section of modern society, including consumer electronics, information technology, science, medicine, industry, law enforcement, entertainment, and the military. Fiber-optic communication using lasers is a key technology in modern communications, allowing services such as the Internet.

Some uses are:

1. Communications: besides fiber-optic communication, lasers are used for free-space optical communication, including laser communication in space.
2. In medicine: Lasers have many uses in medicine, including laser surgery (particularly eye surgery), laser healing, kidney stone treatment, ophthalmoscopy, and cosmetic skin treatments such as acne treatment, cellulite and striae reduction, and hair removal
3. Industry: cutting including converting thin materials, welding, material heat treatment, marking parts (engraving and bonding), additive manufacturing or 3D printing processes such as selective laser sintering and selective laser melting, non-contact measurement of parts and 3D scanning, and Laser cleaning.
 - 1) Military: marking targets, guiding munitions, missile defense, electro-optical countermeasures (EOCM), lidar, blinding troops.
 - 2) Research: spectroscopy, laser ablation, laser annealing, laser scattering, laser interferometry, lidar, laser capture microdissection, fluorescence microscopy, metrology, laser cooling.
 - 3) Commercial products: laser printers, barcode scanners, thermometers, laser pointers, holograms, bubble grams.
 - 4) Entertainment: optical discs, laser lighting displays, laser turntables.

2.3.5 Power

Different applications need lasers with different output powers. Lasers that produce a continuous beam or a series of short pulses can be compared on the basis of their average power. Lasers that produce pulses can also be characterized based on the peak power of each pulse. The peak power of a pulsed laser is many orders of magnitude greater than its average power. The average output power is always less than the power consumed.

The continuous or average power required for some uses in the following Table 6:

Power	Use
1–5mW	Laser pointers
5mW	CD-ROM drive
5–10mW	DVD player or DVD-ROM drive
1W	Green laser in Holographic Versatile Disc prototype development

Table 6. Average power required.

2.3.6 Market survey





- 1) We are limited with the drone weight, the maximum weight that our drone able to suffer is 300g (we tested it physically). So, we are looking for a suitable laser with lightweight.
- 2) The laser must be in enough power that can be able to destroy a specific point that we will get from the parallel project.





Market survey is the survey research and analysis of the market for a product/service which includes the investigation into customer inclinations. Market surveys are tools to directly collect feedback from the target audience to understand their characteristics, expectations, and requirements, cf. [21].

The purpose is to gather data on customers and potential customers. The collected data aids business decision making. This reduces the risks involved in making these decisions.

The requirements for the project:

We made a market survey to decide which laser is suitable to our project, in the following Table 7 we will introduce the laser types we found and if it suitable to the project according to its specification:

Laser name	Speciation
Green laser pen cf.[35].	<p>Power: 100mW. Having sub-head shapes. Wavelength: 532nm. Weight: 75 gr. Max range: 5000m. Price:130₪.</p>  <p><i>Fig 49: Green laser pen cf. [35].</i></p>
A super strong red laser cf. [36].	<p>Power: 200mW. Weight: 182 gr. Max range: 5000m. Price:199₪.</p>  <p><i>Fig 50: A super strong red laser cf. [36].</i></p>
Green laser head effects cf. [37].	<p>Power: 50mW. Weight: 75 gr. Max range: 5000m. Price: 159₪.</p>  <p><i>Fig 51: Green laser head laser head effects cf. [37].</i></p>
Green Laser Module 3V cf. [39].	<p>Power: 100mW. Size:3.5cm x 1.2cm x 1.2cm. Weight: 18gr. Price: 27\$.</p>  <p><i>Fig 52: Green Laser Module 3V cf. [39].</i></p>

<p>Green Laser Diode Module Line cf. [40].</p>	<p>Wavelength: 532nm Power: 50mW Cross Section size: 21x21mm. Price: 25.37\$.</p>  <p><i>Fig 53: Green Laser Diode Module Line cf. [40].</i></p>
<p>Green laser cf.[16]</p>	<p>Power: 2.5mW. Wright: 110 gr. Max range: 2000m. Price: 11\$.</p>  <p><i>Fig 54: Green Laser cf. [16].</i></p>
<p>EASYIDEA laser cf.[17]</p>	<p>Power: 5mW. Weight: 100 gr. Max range: 5000m. Price: 7.72\$</p> <p><i>laser</i></p>  <p><i>Fig 55: EASYDEA cf. [17].</i></p>
<p>Pointer Pen Laser cf.[18]</p>	<p>Power: 0.5mW. Weight: 90gr. Max range: 2000m. Price: 11,99\$.</p>  <p><i>Fig 56: pointer laser cf. [18].</i></p>


<p>KELUSHI laser cf.[29]</p>	<p>Power: 5mW. Weight: 47gr. (without battery) With 47+11=58gr. Max range: 5000m. Price: 14.9\$.</p>	 <p><i>Fig 57: KELUSHI laser cf. [29].</i></p>
----------------------------------	--	---

Table 7. Lasers Specification.

Initially, we thought that 5mW is enough power to destroy an object like a balloon. So, we chose a laser-based on this thought. Then, we decided to choose KELUSHI laser (see, Fig.57) as the official laser that we will use in the project because the KELUSHI laser has more advantages than the others. In fact, its weight is suitable to what the drone can suffer, it is powerful, and the maximum range can be up to 5000m. So, this laser is the perfect and the most suitable to our project.

However, after we made many types of research about lasers we found that 5mW isn't enough, and the minimum power that the laser should be is 50mW to destroy an object. So, we searched more lasers with minimum 50mW power, and we added them to Table 2. Now, we have to decide which laser is the most suitable to this project. We found that the green laser module 3v is the most suitable one, according to its specification. In fact, one can make the laser turn on/off without a button. So, we can apply the second solution that we mention on item 10 in 1.6 section. Moreover, also the green laser module 3v power is 100mW that more than the minimum power that laser can destroy a balloon, (the minimum is 50mW), and this laser has 100mW. In addition, its weight is completely suitable because it has lightweight. Finally, this laser has a reasonable price. So, according to all this laser specification, we can assume that all our project requirements will succeed without any doubt.

2.3.7 A mechanism to connect a laser to the drone

According to the criteria, which we mentioned above, we must have a mechanism that connects the laser to the drone. Also, this laser should rotate about two axes: vertical and horizontal.

Our mechanism, described below, allows rotation of a laser about those two axes. The chosen mechanism is presented on Fig 58.

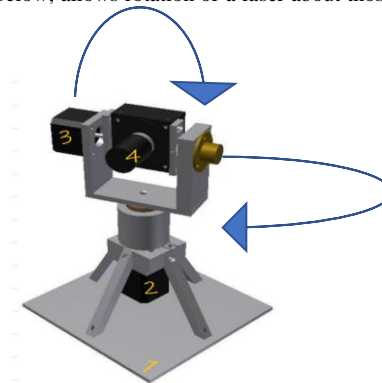


Fig 58: The mechanism description.

Description for the numbered parts of Fig.58:

1. Describes the place where we connect this mechanism to the drone, we can connect in various ways (such as welding, fasteners, etc...)
2. Describes the first servo motor that is responsible for the rotation of the laser about a vertical axis.

3. Describes the second servo motor that is responsible for the rotation of the laser about the horizontal axis.
4. Describes the place where we connect the laser to the mechanism.

2.3.8 System description

- Laser movement system that made up of two servo motors and the laser itself.
 - Arduino microcontroller.
 - Radio waves receiver sensor.
 - Radio waves transmitter sensor.

2.3.9 Motor selection

According to what we explained above, the two motors are responsible for the laser movement. So, we choose to use servo motors (described in the next section in detail), because they allow accurate control of angular position because they made up of sensors for the locations.

Servo motors demand related microcontroller that has been planned to receive sensors signals and commands the engine to make the demands movement. An extended explanation will be explained in the next section, cf. [23].

2.3.10 Servo motor

A servo motor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

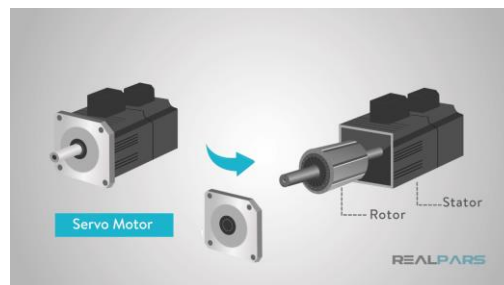


Fig 59: Servo motor, cf. [25].

2.3.11 Servo motor Mechanism

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.

More sophisticated servomotors use optical rotary encoders to measure the speed of the output shaft and a variable-speed drive to control the motor speed. Both enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting, cf. [24].

2.3.12 Servo Motor advantages

- 1) High output power relative to motor size and weight.
- 2) Encoder determines accuracy and resolution.
- 3) High efficiency. It can approach 90% at light loads.
- 4) High torque to inertia ratio. Servo Motors can rapidly accelerate loads.
- 5) Has 2-3 times more continuous power for short periods.
- 6) Has 5-10 times more rated torque for short periods.
- 7) Servo motors achieve high speed at high torque values.
- 8) Quiet at high speeds.
- 9) Encoder utilization provides higher accuracy and resolution with closed-loop control, cf. [25].

2.3.13 Microcontrollers

A microcontroller is a small computer on a single metal-oxide-semiconductor integrated circuit chip. In modern terminology, it is like but less sophisticated than, a system on a chip, a SoC (System On a Chip) may include a microcontroller as one of its components. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR is flash, or PROM (Programmable Read Only Memory) also often included on-chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general-purpose applications consisting of various discrete chips.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys, and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed-signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. In the context of the internet of things, microcontrollers are an economical and popular means of data collection, sensing and actuating the physical world as edge devices, cf. [46].

2.3.14 Communication between our mechanisms and the remote controller

The drone works among the 100-200-meter range. So, we demand communication that works well with this range. There are many communication methods found in the market, wither it Bluetooth, or via radio waves. The following Table 8 listed the wavelength range of the Bluetooth, cf. [27]. The following Table 9 presents the wavelength range of the radio frequency, cf. [28]. After many market reviews we did about the communication method, we choose to use radio communication because radio waves allow work with this range.

Level	Band	wavelength range	examples
1	High	Up to 100m	USB adapter
1.5	Medium\high	Up to 30m	Beacon
2	Medium	Up to 10m	Mobile device

3	Low	Up to 1m	Bluetooth device
---	-----	----------	------------------

Table 8. Wavelength range of the Bluetooth.

The following Table 9 presents the wavelength range of the radio frequency.cf. [28]:

Frequency	Band	wavelength range
30–300 kHz	Low	10-1 km
300–3000 kHz	Medium	100-1000 m
3–30 MHz	High	10-100 m
30–300 MHz	Very High	1-10 m

Table 9. Wavelength range of the radio.

3. THE MAINSTREAM OF THE FULL PROJECT

In this project our goal is to combine all the products of the previous projects, made by another teams, into one complete system which will be a drone that is capable of detecting camouflaged objects in the field, using the device's camera and multiple additions, like Optical Flow device (PX4FLOW) and a special micro-controller (PIXHAWK). The system will also be capable of detecting aerostats that are flying nearby (like Incendiary balloons or other drones). After detection, the system will be capable to notify the operator in his controller PC, and allow him to liquidate the detected targets using a built in laser component.

In the first phase of our project, we got the following products:

1. Laser Housing Mechanism:
The mechanism got reconstructed because of a 'dead area' it couldn't aim to. But in the end of previous groups project, it did work as intended and aimed correctly to every direction it was given.
2. Camouflaged objects and Aerostats detection algorithm:
The algorithms that were implemented by the previous teams weren't tested against real-time video stream. It was tested and successfully detected its targets in a static images and short videos. The camouflage detection worked well and in a good rate of 24 fps.
The aerostat detection worked in 5 fps, significantly less than the camouflage detection, and in order to work at this rate it needs a high end GPU.
3. Drone, PIXHAWK, P4XFLOW, Radio Transmitter and Receiver.
We had to install all the equipment on the drone in such a way that it won't disable it from flying or landing, and the laser pen could be freely pointed to any location within the drone's sight. The drone is a compact device that wasn't intended by DJI to hold third-party accessories, so it's a challenge that we had to solve.

4. PC – Ground Station

We needed to make a solid communication channel between the microcontroller, which is installed on the drone, and make sure that it's reliable and work in real time. This connection is yet to be made by previous teams, so it's one of our goals.

5. Detected Object Tracking algorithm

The purpose of the Optical Flow device in our system is its ability to compute the direction and velocity of the drone, so it will be able to follow the detected object even if the drone is not static in the air. The algorithm isn't yet implemented and tested.

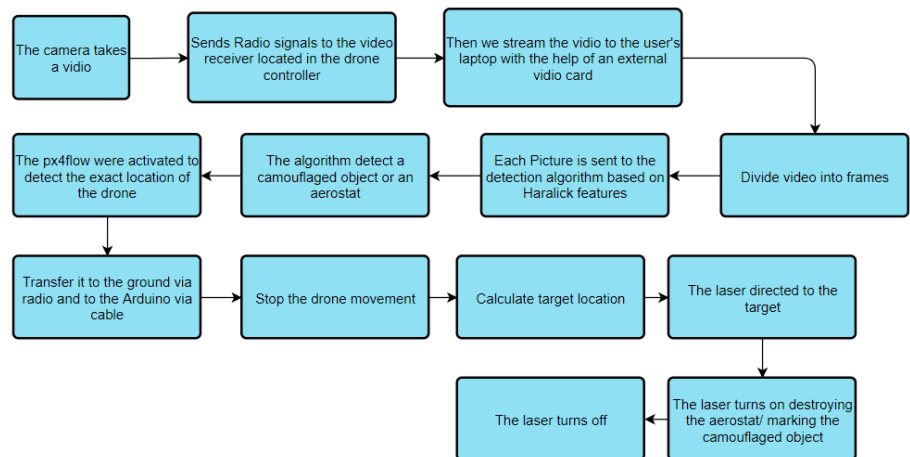


Fig. 60: The full process description

In this, the second phase we tried to work from where the previous groups manage to get it. The laser aiming mechanism worked well and both detection algorithms from groups did give coordinates. Connecting the detection with aiming was therefore simple enough. It worked and did aim to coordinates that the Arduino controller received from detection algorithms.

Due to logistic issues, we didn't receive much needed parts like the communication device. Therefore, we couldn't work on assembling the parts on the drone. This is a hard issue because of this drone model. It has sensors that shouldn't be block or else it will refuse to take off from the ground. Those sensors are crucial for its stability in flight.

So, we worked on the algorithm and tried to make them work well on video streams. The aerostat detection was slower and seemed like the harder part to work on. The algorithm from previous group was based on "detectron2" which is an open-source detection algorithm developed in Python by Facebook. It is stated that this algorithm works in real time.

After a lot of work, we managed to improve the previous team's algorithm and make it work in a higher rate than 5 fps on video recording and 2.5 fps on video stream. The previous algorithm worked on Mask RCNN model which also threw errors while processing video stream. We deleted the Mask, left the FasterRCNN and retrained the algorithm. It did work in a higher rate of 10 fps of video recording and 6 fps on video stream but with 1 second delay. It was better but 6fps is not enough and the delay making the results irrelevant in real time perspective.

The previous group had better results after testing on a computer with a high end GPU. And we thought this was the main reason for this delay but after testing we realize the source of the problem.

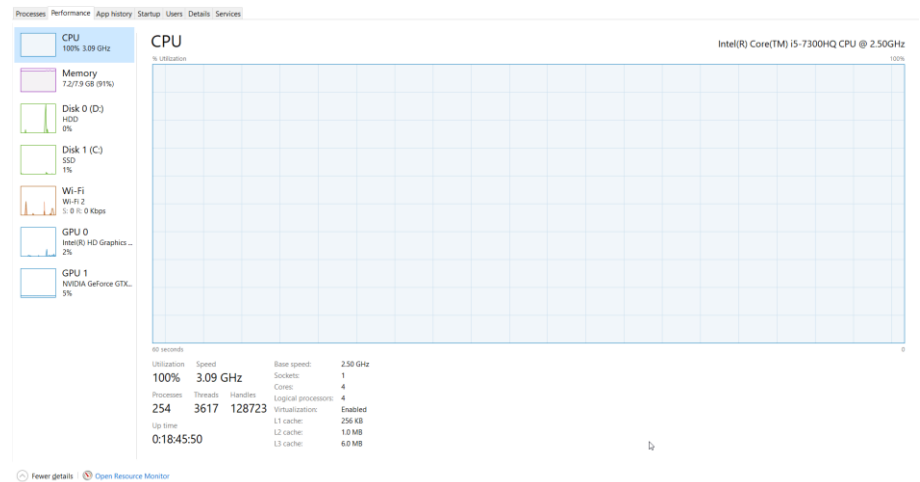


Fig. 61: Load on the CPU

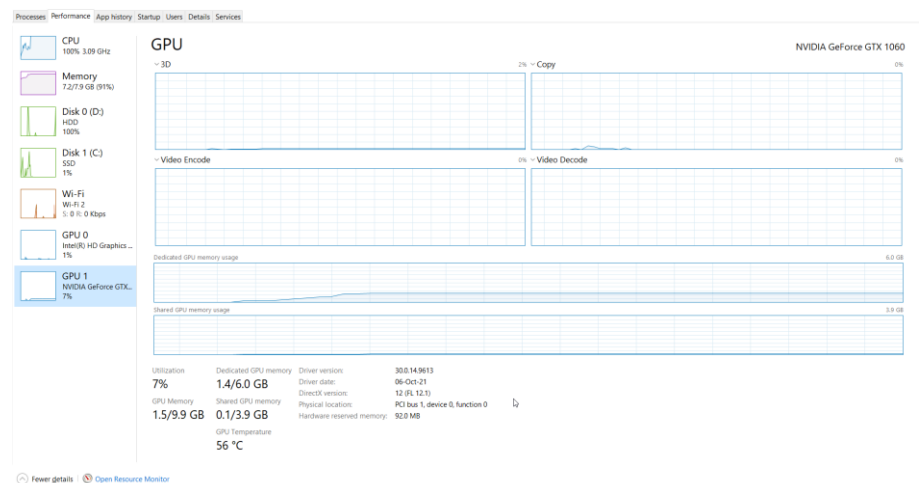


Fig. 62: Load on the GPU

While the CPU works really hard to process the video stream, the GPU that was older and weaker than the high end GPU that previous group had, almost wasn't used.

With no success or hope of improving algorithm based on "detectron2" we decided to look for other detection algorithm.

We found algorithm named "YoloV4" that is also written in Python. This algorithm tries to balance between the speed of detection and correctness. It worked in 60 fps on a video recording but with misscategorization. We continued to train it till it worked well in a rate of 15 fps.

In 15 fps it's success rate, the rate of successful and correct detection, was as well as the "detectron2" based algorithm.

After some more work and training we managed to get it to work in 30 fps with the same success rate.

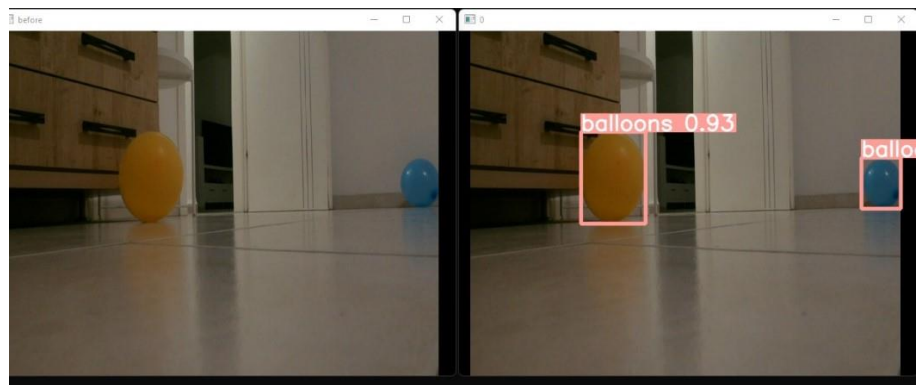


Fig. 63: Successful Aerostat Detection

And it gives the coordinates.

NOTE: the video stream is received in 30 fps. Also it shows a higher rate, it works in the rate of the receiving video stream. Although two targets were detected, the laser will aim on the first one in the queue: the one with the lowest X coordinate. In this example the orange balloon's X coordinate is lower than the blue balloon. The orange balloon will be the first in the queue and the laser will aim for the orange balloon.

```
[ 214.0 203.5 90.0 127.0 ]  
X152Y165  
[ 588.5 211.5 55.0 71.0 ]  
X321Y141  
  
0: 480x640 2 balloonss, Done. (0.019s) FPS: 52.63
```

Fig. 64: Detected aerostats' coordinates

With all the work with aerostat detection we left the camouflage detection behind.

All the algorithms work with a library called OpenCV. In order for the camouflage detection to receive the video stream it need a function from this library. Although, it uses OpenCV for the detection algorithm without a problem, it refuses to cooperate and give access to video stream.

4. EXPECTED RESULTS AND THE REALITY

The final expected result of the project was implementation of the full detection and destruction flow, described above for a Mavic Pro 2 drone, carrying a micro-controller, radio module, laser module with positioning mechanism and a software installed on a computer on the ground.

We expected to successfully detect the object, be it a sniper on the ground or an aerostat in the air, and liquidate it in real time by a laser in the following way:

First, we hoped to successfully detect the targets (camouflaged objects and aerostats) in real time video streaming from the drone to the ground station. For this purpose, we wanted to implement the radio based connection between the micro-controller and the ground station to be stable and sufficient for real time streaming.

Second, we expected to locate the objects accurately enough for its liquidation. We hoped to successfully calculate its exact coordinate related to the drone.

Third, we expected the laser module to be successfully directed towards the target's coordinates and track its position. So, the beam would still be aimed towards the target and liquidating it even if the drone isn't static.

We didn't achieve the wanted results completely.

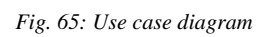
The detection of aerostats from real time video stream does work. It detects aerostats and does not miss-categorize. But the most cheering achievement is that we succeeded in making the detection work in 30 fps from the 60 fps transmitted video. It's a significant improvement from the 5 fps of a saved video. Our supervisors told us that the detection should be at least visually with no delay between received video stream and the output video.

The detection of camouflaged object: we failed to make it work on a video stream from the drone. The detection algorithm of camouflage works with an open-source library. We tried to connect it with the algorithm, but it didn't cooperate well. The detection worked well but it refused to get the input from the camera. We searched for a solution but to no avail.

Both algorithms do provide coordinates. The camouflage detection provide X,Y,Z coordinates of one target and the aerostat detection provides X,Y coordinates of all recognized targets which after words convert to X,Y,Z that the aiming algorithm use. The aiming algorithm does recognize these coordinates, and the mechanism aims the laser to the target. In case of multiple targets in aerostat detection, the laser will aim for the first target in a queue, The one with smallest X coordinate in the initial X,Y coordinates. However, the communication device that would transmit between the aiming mechanism and computer was not supplied on time due to logistic issues.

We wanted to develop to a state of a finished product but succeed only to develop further and closer to the wanted end.

5.1 Use Case



Actor	Use case	Description
Micro-controller	Undergoing set of mathematical operations	The micro-controller does mathematical operations to calculate the distance

		between laser and desired object
	Sending signals	The micro-controller sends signals to do so many operations such as controlling the motors (turning them on or off) or making the laser returns to the default point.
	Coordinates translation	It does this by interpreting data it receives from its I/O using its central processor, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data.
Px4low	Sending data to the receiver sensor	Px4flow processing optical flow algorithm
	Transmission data to Pixhawk	Transmission data to Pixhawk via UART
Login	Login	System checks the logging fields and grants/denies the user access.
Start the detection	Start the detection	The program starts to run to identify camouflage.
-		Algorithm starts to divide the video record into images from the connected camera.
-		Algorithm receives frames (images).
-		Algorithm converts frames from RGB to grayscale.
-		Algorithm uses clustering algorithm.
-		Algorithm divides the frame (image) into sub-images.
-		Algorithm uses Texture analysis.
-		Algorithm checks for similarity between the photo and the texture from the DB.
-		Algorithm warns the user if detection were found.

View results		Present the result to the user.
--------------	--	---------------------------------

Table 10. Use case description

5.3 Class Diagram:

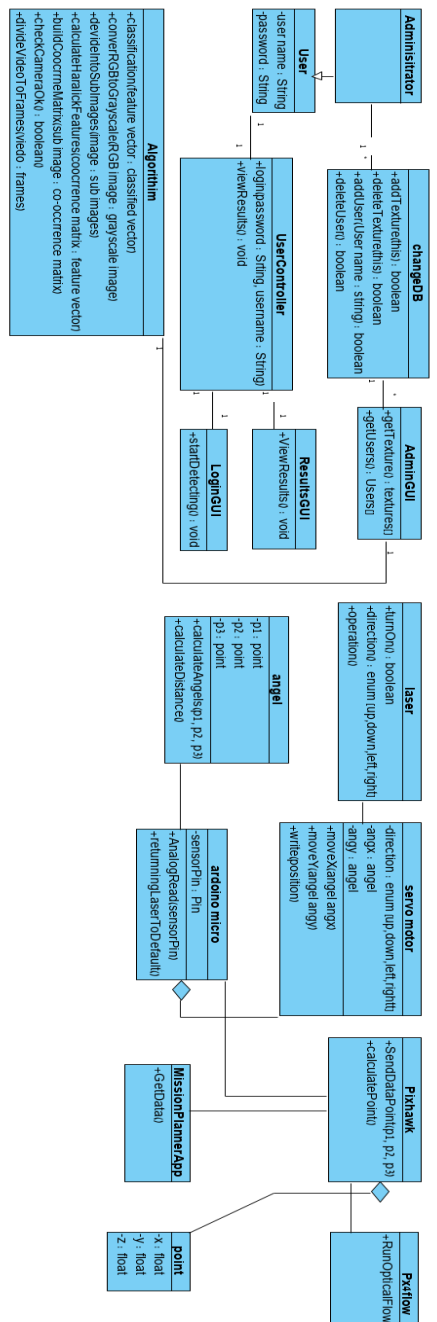


Fig. 66 Class diagram

5.4 Testing Plan (two kinds of testing : algorithm detection and real time mode)

5.4.1 Testing the Detection algorithm.

No.	Test subject	Expected result	Result
1.	Camouflaged object with the exact same texture, like the background.	System would not detect the camouflage.	Object was not detected.
2.	Camouflaged object with a texture different from the background.	System would detect the camouflage.	The object was detected successfully.
3.	Camouflaged object far distant from the camera.	System <u>probably</u> will not detect the camouflage.	Maximal detection distance: approx. 10-15 meters
4.	Camouflaged object close to the camera.	System will detect the camouflage.	The object was detected successfully.
5.	Capturing pictures using low resolution camera.	System <u>probably</u> will not detect the camouflage.	The object was detected successfully.
6.	Capturing pictures using high resolution camera.	System will detect the camouflage.	The object was detected successfully.

Table 11. Testing the model

5.4.2 Sanity test of the code.

In order to check out the system performance we will run the program on some significant input:

No.	Test subject	Expected result	Result
1.	User log-in with incorrect details.	System denies the user access.	Login denied.
2.	User log-in with correct details.	System grants the user access.	Login accepted.
3.	Administrator tries to delete textures from database	The textures deleted from the database	The textures deleted.
4.	Administrator tries to add textures to the database	The textures successfully added to the database.	The textures successfully added.
5.	User starts the algorithm without camera connected.	System notifies the user to connect the camera.	Aerostats algorithm won't run without webcam connected.
6.	User starts the algorithm.	System ready and start searching for camouflaged objects in photos received from the camera.	Frames analyzed successfully by the detection algorithm.

Table 12. Testing the model

a. Testing Plan of Real Mode:

As we already explained, our solution must effectively work in real time mode. That is why we must profoundly investigate the ability of the chosen hardware and the implemented software to provide the expected results. Moreover, we must investigate the application boundaries by multiple static and dynamic experiments.

b. GUI

Main menu GUI.

The main menu GUI will be very simple for the user. The user will first chose what to search for, aerostats or camouflaged objects.

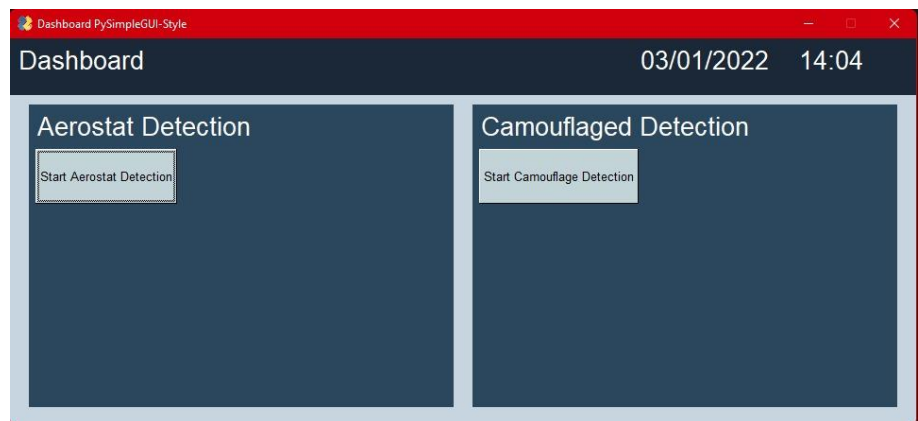


Fig. 67 Main menu GUI

GUI for the results.

Also, this GUI will be very simple, once the system detects a camouflage or an aerostat, it will notice the user. The implementation will visually mark the target.

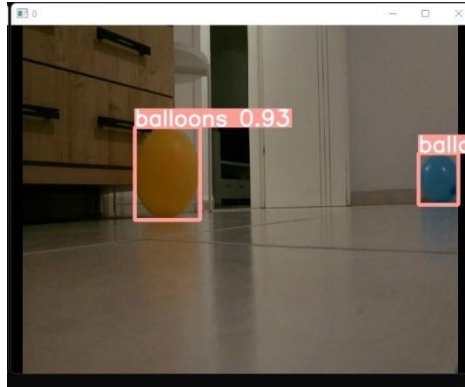


Fig. 68 The result in GUI for aerostat detection

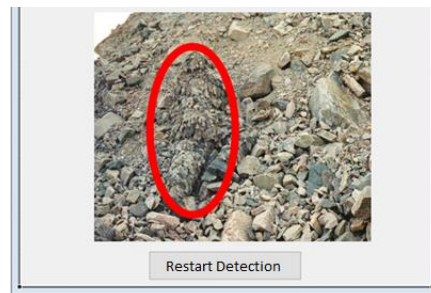


Fig. 69 The expected result in GUI for camouflage detection

6. Operation Instructions

6.1 General Description

Aerostats and camouflaged object detection system consist of 2 separate and totally different object detection and tracking algorithms:

6.1.1 Aerostats object detection algorithm

The aerostats object detection algorithm is a Real-Time Machine Learning object detection algorithm, based on YoloV5 Pytorch implementation of the YoloV5 algorithm family.

The algorithm is capable of analyzing and inference from multiple input sources, like local image files, video files and webcam (Using an open-source library could OpenCV).

The objects that are predicted by the algorithm are visualized in a Real-Time stream by bounding boxes that are drawn inside the frame.

6.1.2 Camouflaged object detection algorithm

The camouflaged object detection algorithm is based on 13 Haralick Features, which Robert Haralick suggested in his article from 1973, and in a SVM (Support Vector Machine) model.

The algorithm is capable of analyzing and inference from images and videos. The objects that are predicted by the algorithm are visualized in a stream by mask object that is drawn inside the frame, and over the detected object.

6.2 How to use

Inside the project's root folder, 'Balloon-Camouflage-Detection', there is the main GUI component, called 'Demo_Dashboard'.

Edit 'Demo_Dashboard' as follows:

- 1) In line 56, change first argument of 'subprocess.call' to the following command, after changing the red paths to the correct paths on your local machine:

```
[PYTHON_PATH]/python.exe [PATH_TO_FOLDER]/Balloon-Camouflage-Detection/yolov51/detect.py --source 0 --weights [PATH_TO_FOLDER]/Balloon-Camouflage-Detection/yolov51/best.pt --conf-thres 0.7 --half --dnn
```

Fig. 70: Connecting to detection algorithm

- 2) In 'detect.py' inside 'yolov51' subfolder, change line 51 and define the correct COM number for the Arduino USB interface:

```
ArduinoSerial=serial.Serial('COM6',9600,timeout=0.1) #  
Define COM number for arduino interface
```

Fig. 71: Connecting to Arduino

- 3) Install dependencies:

Make sure 'PIP' package is installed.

Open PowerShell from 'yolov51' folder and execute:

```
pip install -r requirements.txt
```

Fig. 72: Installing dependencies

- 4) Run 'Demo_Dashboard.py', and choose the algorithm you want to fire:

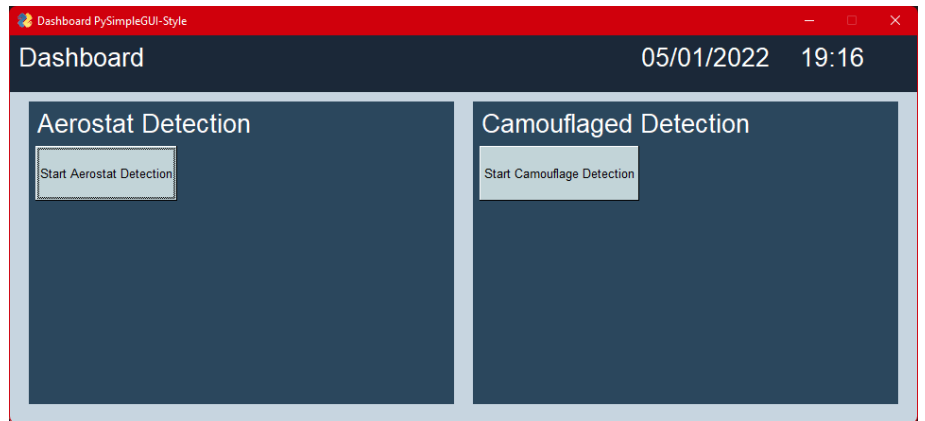


Fig. 73: First Screen

7. Maintenance Guide

7.1 Software Structure

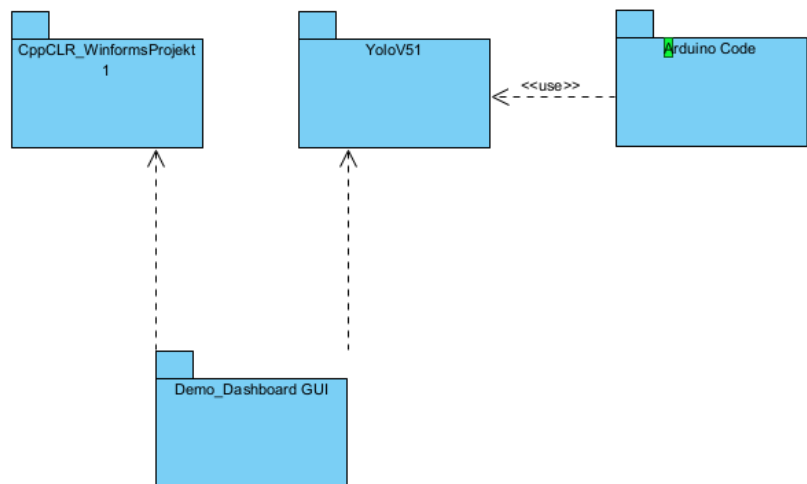


Fig. 74: Package Diagram

7.2 Requirements:

```
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.41.0
tensorboard>=2.4.1
#wandb
pandas>=1.1.4
seaborn>=0.11.0
#coremltools>=4.1 # CoreML export
#onnx>=1.9.0 # ONNX export
#onnx-simplifier>=0.3.6 # ONNX simplifier
#scikit-learn==0.19.2 # CoreML quantization
#tensorflow>=2.4.1 # TFLite export
#tensorflowjs>=3.9.0 # TF.js export
#openvino-dev # OpenVINO export
#albumentations>=1.0.3
#Cython # for pycocotools
#pycocotools>=2.0 # COCO mAP
#roboflow
```

Fig. 75: Required Libraries

8. REFERENCES:

- [1]. Real Time Camouflage Detection with A Special Computation Unit. Rani Halabi, Samer Kinaan, BSC final project by Dr. Elena Ravve, Dr. Dan Lemberg supervisor, 2018.
- [2]. Real time dynamic camouflage detection. Eden Kisliankov and Nael Halabi. BSC final project by Dr. Elena Ravve, Dr. Dan Lemberg supervisor, 2019.
- [3]. https://profs.info.uaic.ro/~ancai/DIP/lab/Lab_2_DIP.pdf
- [4]. <https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/>
- [5]. Textural Features for Image classification, Robert Haralick, K. Shanmugam and itshak dinstein 1973
- [6]. 3D EXTENSION OF HARALICK TEXTURE FEATURES FOR MEDICAL IMAGE ANALYSIS, Ludvik Teser, Daniel Smutek, Akinobu and Hidfunne Kobatake, 2007
- [7]. https://en.wikipedia.org/wiki/Center_of_mass
- [8]. https://en.wikipedia.org/wiki/Optical_flow
- [9]. https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle
- [10]. https://en.wikipedia.org/wiki/Support-vector_machine
- [15] <https://www.epiphan.com/store/avio-hd/>
- [17] <https://www.epiphan.com/products/avio-hd/>
- [18] <https://www.startech.com/en-eu/audio-video-products/usb32hdes>
- [19] <https://www.epiphan.com/products/avio-4k/tech-specs/>
- [20] https://www.researchgate.net/publication/261503873_An_open_source_and_open_hardware_embedded_metric_optical_flow_CMOS_camera_for_indoor_and_outdoor_applications
- [21] SiK Telemetry Radio — Copter documentation (ardupilot.org)
- [22] Pozyx for Non-GPS Navigation — Copter documentation (ardupilot.org)
- [23] [Amazon.com: GH1.25 Cables and connectors Wire for Pixhawk2 Pixhawk 4 Pixracer PXFmini Edge CUAV V5: Camera & Photo](https://www.amazon.com/GH1.25-Cables-and-connectors-Wire-for-Pixhawk2-Pixhawk-4-Pixracer-PXFmini-Edge-CUAV-V5-Camera-Photo/dp/B075333333)
- [24] <https://ardupilot.org/planner/>
- [25] [RFD900 \(SiK\) Telemetry Radio · PX4 v1.9.0 User Guide](https://ardupilot.org/RF900/SiK/TelemetryRadio/PX4v1.9.0/UserGuide/)