UNIVERSITAT POLITECNICA DE CATALUNYA

UNIVERSITY OF NEW SOUTH WHALES

UNDERSTANDTING SOFT MATTER THROUGH THE USE OF NOVEL BIOPHYSICAL TOOLS

Physics Engineering Final Bachelor's Thesis

Author:

Juan Trius Béjar

Supervisors:

Siddharth Rawat

Anna Wang

Co-tutor:

Jaime Oscar Casas Piedrafita

Barcelona, January 25, 2023

1. Abstract: 2
2. Introduction: 3
3. State of the art: 5
4. Methods: 7
    4.1 Scattering theory 7
    4.2 Holography using planar waves 8
    4.3 Holopy 9
    4.4 Machine Learning 10
        4.4.1 CNN 11
        4.4.2 LeNet 5 14
        4.4.3 Yolov5 14
        4.4.4 Data Augmentation 16
    4.5 Sampling methods 16
    4.6 Optical path length 17
5. Experimental Setup 18
    5.1 In-line holography Microscope. 18
    5.2 Chamber for aerosol collection 20
    5.3 Software of the device 21
6. Results and Discussion 24
    6.1 First steps in Neural Networks 24
    6.2 Machine Learning Approach 25
    6.3 Dataset ampliation and Yolov5 implementation 26
    6.4 Final training 29
7. Conclusions and future outlook 32
8. Bibliography 33
9. Appendix A: Codes 36
10. Appendix B: Test Images 40

1. Abstract

Pollen detection is an ongoing topic reviewed nowadays. This substance's impact on human healthcare is high, causing diseases, allergies and asthma. The principal limitation is the necessity of post-processing the laboratory. Also, pollen recognition has been an issue, even with computer vision. The similarity of shapes and sizes of pollen makes it a difficult task if we only have a microscopic image. Taking three-dimension images seems an alternative and can facilitate the recognition of the different pollen grains. However, the cost of reconstructing those images is high and not very viable considering the objective of a live device. In this project, we propose to use a not usual tool in this field. With its high-information images, Holography may give us the difference when trying to recognize pollen grains. Combined with the powerful processing technique of computer vision, this project could be the first step in the in-live recognition of pollen. It can be ampliated to recognize another type of aerosols. As mentioned in this work, we will propose a device capable of performing live pollen detection using Holography and deep learning as the two driving methods to accomplish our objectives. We will achieve a performance of > 90% and make this device mobile and straightforward.

2. Introduction

Whenever you use a coherent source of light that interferes with an object, you can obtain information about the light's intensity and the image's phase. These two parameters, with the proper processing, can lead us to a hologram[1]. That can characterise the speed of an object, the diffusion coefficient, the position of the object, and the 3-Dimensional shape of the object.

At first sight, it can seem complex to obtain all these features and register this information. However, using a laser in an optical microscope and recording the information with a camera is already a digital microscope ready to operate.

This project's **objective** is to classify pollen holograms using Deep Learning. Pollen, as we know, impacts human health; distinct types of pollen can cause allergies and asthma.[2] Studying the concentration and distribution of pollen in the air can benefit human healthcare and prevent allergies inside buildings and rooms. Pollen is an ongoing field of research in human healthcare, and real-time sensing has just been posed.

Holography is a compelling and not complex technology that can give us an insight into the properties and characteristics of very different areas. During this thesis, we will approach various issues that will be solved using holography. As well as, looking at future outlooks and research paths will open as we use holography as a daily technique.

Moreover, we will explore computer vision techniques. We will study different types of architectures and understand how they work, and even design our prototype, even though more efficient techniques will be found later on. Computer vision is a fashion topic explored and applied to the industry extensively in the last decade.

[2]

The study of pollen through holography[3] has never been performed to this extend before, combined with the capacity of real-time sensing, making this project a challenging opportunity that can be the first step into a more complex and complete work in the field of holography and bioaerosol detection. Additionally, the use of Deep Learning adds an innovative technology capable of changing the paradigm inside holography.

This research project was developed over five months at the University of New South Wales in Sydney, Australia. In the school of chemistry, with Anna Wang's research group. The overall project is not finished and it is still on going. It will finish on march 2023 with the publishing of a paper.

3. State of the art

Seventy-five years after the invention of holography by Dennis Gabor[4], there has been a development of different techniques. During these years, humanity has used holography in art, aircraft, and environmental erosion. This technique is only in its early days since the applications, and the theory is in their early phases.

Pollen has several impacts on human health[2]; allergic rhinitis and asthma[5] are the two most common and dangerous triggers pollen can have in our bodies. In Australia and New Zealand, around 18% of the population suffers from Allergenic rhinitis, and 2.7 million habitants (10,7%) in Australia had asthma during 2020-2021.[6] Therefore, controlling pollen concentration in outdoor or indoor environments can be a considerable step when discussing pollen allergies and diseases.

Pollen Sensors typically that are present in the industry involve laboratory personnel. If it is not the case, usually, the sensors recollect the pollen, and then the analysis is performed.[7] Polleno uses digital holography to reconstruct in-focus images of airborne particles. The option of using holography is because the information inside the holographic makes it possible to distinguish the different species of pollens. Alternatively, even the company Pollen Sense has a sensor capable of identifying pollen automatically after recollecting the pollen into a substrate. We can see that the pollen classification field is an explored one. However, there needs to be a consensus on the best principles to obtain a real-life, accurate classification pollen device. Also, a Label-Free Bioaerosol sensing has been proposed, where they use classification through a Convolutional Neural Network[8]. However, before the classification, the image is reconstructed through holography. This work is very similar to the one that we want to perform; the only thing is that we want to use real-life image sensing. The device uses a sticky coverslip. So the sensing has to be performed afterwards and is limited to the capability of the sticky coverslip.

Another approach to classifying pollen was to use proper machine Learning. Therefore, we could extract some parameters that could lead to a quality pollen classification. For example, using the refractive index and radius was enough to detect and classify different contaminants inside the water[9].

After a quick review of the state of the art, we can conclude that similar research has been performed. Therefore, there are others in the field of pollen sensing. During this project, we will surpass the limitations of the actual devices. Hence, we will propose a real-time sensing device, able to recognize different types of pollen, cheap and mobile. We may not surpass some of these limitations, but we will put the first stone in this topic.

4. Methods

This section will review the main tools used during the project development. We will review the basics of the Holography theory. Then we will go deeper into the use of the software able to perform the calculations needed for reconstructing the holograms. Also, we will review the different approaches to machine learning.

4.1 Scattering theory

Interference is the basic principle used in Holography. When recording the object wave (scattered) and the reference wave, we can obtain an interference pattern that can be analysed (Fig.1). Holography can be used with a wide variety of materials and objects. The only requirement is that we have a well-known reference light, for example, a planar wave or a spherical wave.

Typically, the waves are collected on a photographic plate. Nowadays, using cameras, we can also reconstruct objects. This technique is called Computer-generated-Holography (CGH)[10]. For the reconstruction of the hologram, we irradiate the photographic plate with a second reference field.[11] When we illuminate the Holographic plane with the scattered field and the reference field we obtain the following intensity in the plate:

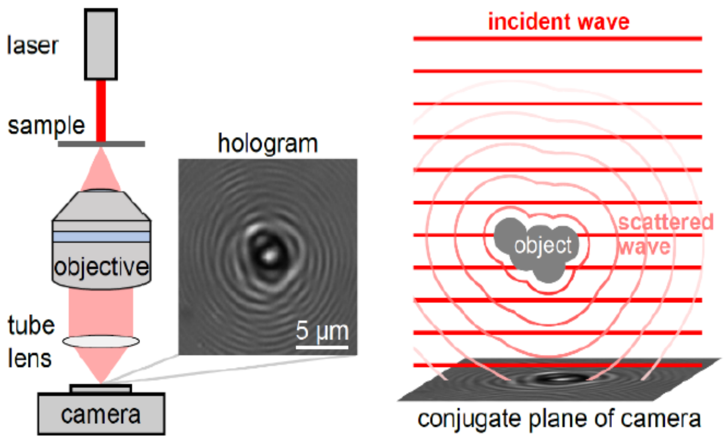


Fig. 1: Summary of a holographic microscope. The laser irradiates the sample that scatters the field and with the help of the objective and tube lens an hologram is formed in the camera

$$I = |E_R + E_O|^2 = |E_R|^2 + |E_O|^2 + E_R E_O{}^* + E_R{}^* E_O \tag{3.1}$$

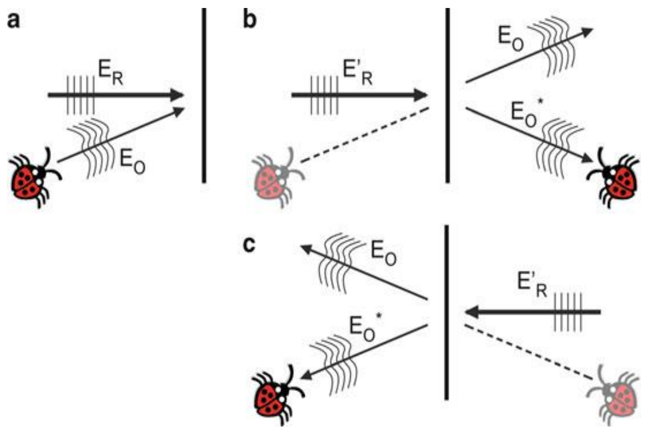


Fig. 2: Timeline of an hologra process (a) Reference field and the object field irradiate to the pothographic plate (b) Images are obtained from each Electric field (c) Reference Field illuminates the photographic plate

If we are trying to reconstruct the hologram, we will have to add another term related to the second reference light field that illuminates the holographic plane (Fig. 2). So, the overall Electric field that we receive is:

$$E = E_{R2}I = E_{R2}|E_R + E_O|^2 = E_{R2}|E_R|^2 + E_{R2}|E_O|^2 + E_{R2}E_R{E_O}^* + E_{R2}{E_R}^*E_O \qquad (3.2)$$

Therefore, by knowing the shape of the incident wave we can reconstruct the scattered field, so that we can understand the shape of the object and the 3D position.

4.2 Holography using planar waves

Since we use a laser as our coherent light source, we use the paraxial approximation. So near the beam centre, the gaussian beam resembles a planar wave. Accepting the paraxial approximation.[11] For the range of microns, it is not far from reality.

Returning to the scattering of planar waves and considering the scattering object a point source, we review the wave equation for the three waves. The difference between them is the incident angle φ. The reference wave without an angle incident, the scattered wave with an angle and the second reference wave with an angle φ'.[11] Therefore, the three equations result in the following:

$$E_1 = \varepsilon_1\, exp(ikz) \qquad (3.3.1)$$
$$E_2 = \varepsilon_2\, exp(ik(xsin(\varphi) + zcos(\varphi))) \qquad (3.3.2)$$
$$E_3 = \varepsilon_3\, exp(ik'(xsin(\varphi') + zcos(\varphi'))) \qquad (3.3.3)$$

Then, the received field is calculated using Franhofer diffraction as:

$$\begin{aligned} E(\theta) &\propto \int_{-\frac{a}{2}}^{\frac{a}{2}} dx I_{12} E_3(z=0)\, exp(ikz) = \cdots = \\ &= [|\varepsilon_1|^2 + |\varepsilon_2|^2]\varepsilon_3 asinc\left[\frac{k'a}{2}(sin\varphi' - sin\,\theta)\right] \\ &+ \varepsilon_1\varepsilon_2\varepsilon_3 asinc\left[\frac{ka}{2}sin\varphi - \frac{k'a}{2}(sin\varphi' - sin\,\theta)\right] \\ &+ \varepsilon_1\varepsilon_2\varepsilon_3 asinc\left[\frac{ka}{2}sin\varphi - \frac{k'a}{2}(sin\varphi' - sin\,\theta)\right] \end{aligned} \tag{3.4}$$

This calculation results in an interference pattern as a sinc function (Fig.3). Our problem, considering point sources, changes substantially since a point source emits a spherical wave instead of a planar wave towards the holographic plane.[11] To calculate this interference, we have to use Lorentz Mie's theory, and the integration gets complex as we go deep inside.

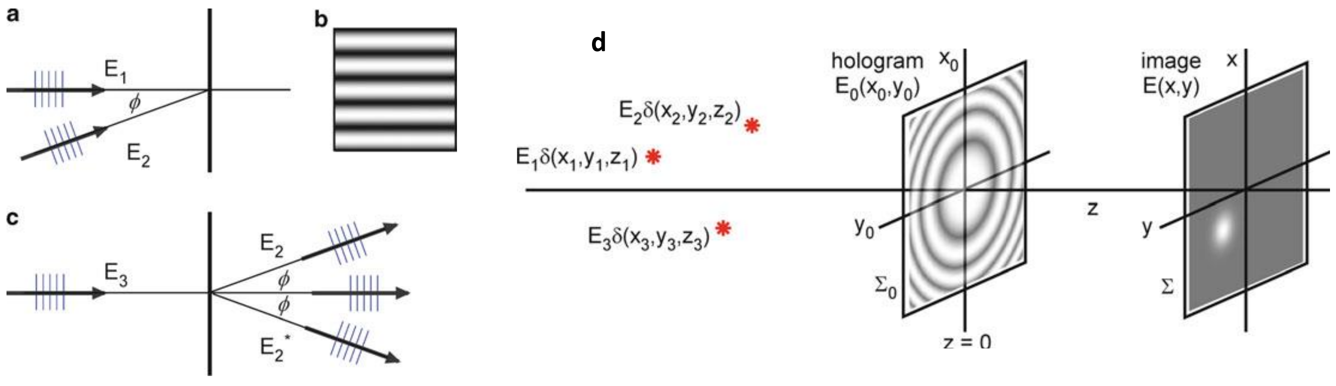


Fig 3: Electric fields of the two different types of interferences. (a) Incident plane waves to the plate (b) characteristic sinc pattern hologram (c) diffraction from the sine hologram (d) interference of point sources that emit spherical waves into the holographic plane.

4.3 Holopy

To process all this theory, Barkley developed a solution called Holopy[12]. Holopy is an open python library used to reconstruct holograms of their experiments. Fortran codes with the corresponding scattering solutions were uploaded and used in this solution. First implemented, Lorenz-Mie theory, which describes the scattering of a homogeneous sphere; from there, more scatterers were implemented as groups of spheres, spheroids, cylinders and even arbitrary particles.

Nevertheless, the most significant package in Holopy is the capability to fit data into scattering models. With the implementation of a Markov chain Monte Carlo simulation. Holopy can fit 15 free parameters (3D position) in a scattering model of five colloidal particles (Fig.4). Also, holopy can infer an object's properties, such as the radius, refractive index, or other parameters and can also simulate noise.

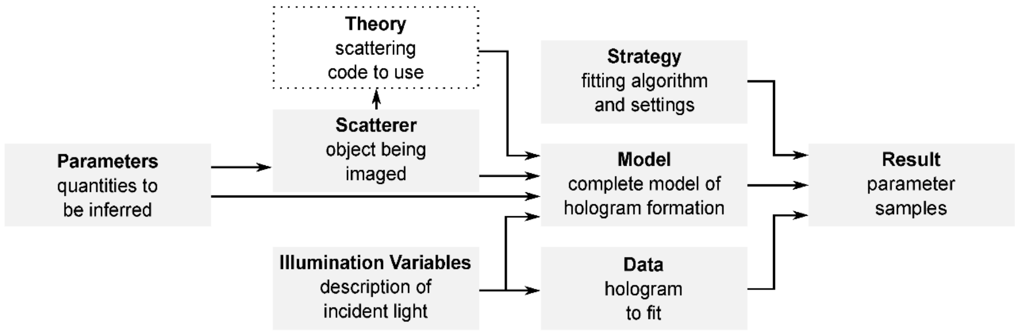


Fig 4: Holopy processing workflow to fit the parameters from a hologram

4.4 Machine Learning

Machine learning has become a modern and powerful data technology to analyze and process data. In computer vision, ML techniques, specifically Deep Learning, are very useful and used in many areas and topics[13]. For this project, we will use computer vision AI to classify holograms of pollen.

To perform machine learning, we need a large dataset for training; these images are classified correctly following a certain pattern. Existing techniques, such as data augmentation, can increment the variety and number of the dataset.[14] For pre-processing data in Machine learning, we should extract some parameters where we can make the classification.

However, in this project, we will use Deep Learning, a variant of Machine Learning where the extraction of parameters that determine the class of an image is determined directly by the algorithm, so we do not have to extract any parameter.[15] Afterwards, in the results section, we will see another approach performed in our experiments using proper parameters and machine learning.

4.4.1 CNN

First, we have to select an architect for our neural network when deciding to use deep learning. Following state of art, we decided that a convolutional neural network (CNN) was the most suitable for this project. CNN consists of three main layers, convolutional layer, pooling layer and fully connected layer, and we have to add the input layer (Fig. 5).[13] Any of these main layers can be very complex or very simple. Now we will see each of the thin layers and the main characteristics:

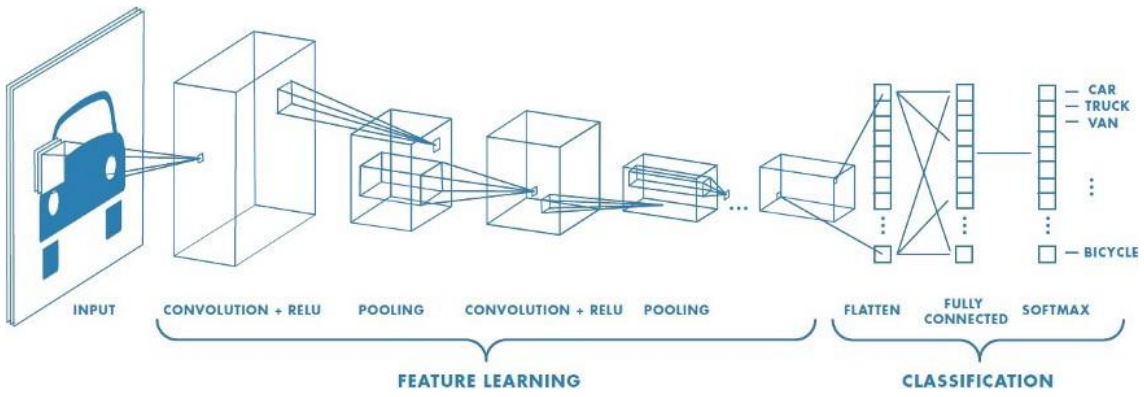


Fig. 5: Diagram with the different parts of the architecture of a convolutional neural network. As the input goes through each later the object take a different size to finally be able to classify it.

- **Convolutional layer**: This layer is in charge of performing the convolution operation (Fig.6). This layer pretends to extract high-performance features that help the algorithm better classify. Examples of feature extraction are the edges of an image, colour, and gradient orientation.

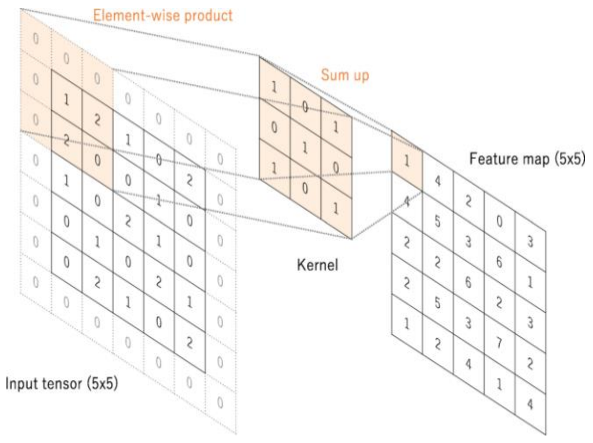


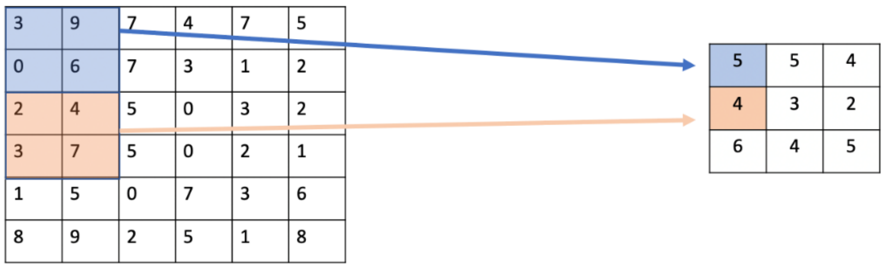
Fig.6: Convolution product explained graphically. This operation is performed through all the image.

Performing convolution gives us several advantages. This technique is invariant to the position and the angle of the features present. For example, pollen spikes will form a feature in the case of pollen. It does not depend on the angle or the position to activate the correspondent node from the neural network. Moreover, this invariance for position and orientation saves us much computational power since the feature of the spike will activate 34 times in different spaces. This means we only have 34 features to work with, only one in different positions.[13] For the activation of a kernel (feature) in a layer, we use an activation function. The most common one is ReLU, rectified linear Unit:

$$ReLU(x) = \max(0, x) \tag{3.5}$$

This activation function is simple to implement because of the low computation. The only issue that has this function is that for activations lower than zero in pieces of training, ReLU kills the node, so there will be no further activation. Nevertheless, this feature could be in further layers or future learning be activated as a high-performance parameter.

- **Pooling layer**: The pooling layer is responsible for delimiting the size of the convolution product resulting from the convolutional layer. By performing this reduction (Fig. 7), we select the most predominant features inside the image. There are different types of pooling, but we can mainly differentiate two of them: MaxPooling, where the maximum value is returned from the piece of the image, we are playing the pooling. Moreover, the Average pooling, which returns as the name indicates the average portion of the image.



3	9	7	4	7	5
0	6	7	3	1	2
2	4	5	0	3	2
3	7	5	0	2	1
1	5	0	7	3	6
8	9	2	5	1	8

5	5	4
4	3	2
6	4	5

Fig.7: Pooling layer explained, in the image we see that they are applying an Average pooling. If the pooling was MaxPooling, the values would be 9 and 6 for the blue and orange square respectivately.

- **Fully connected layer**: In the fully connected layer, each node is connected with each of the outputs of the following layer. These connections are the ones that result in the posterior classification of the image. The nodes are activated through the use of a non-linear function.

Well-known used architecture is LeNet, developed in 1990[16]; that is the first successful application of Convolutional Neural Networks that was first designed to classify hand-written numbers (MINST dataset). Other applications are AlexNet, ZFNet, and GoogleNet (citations). During the development of this project, we tried two main architectures LeNet 5 (5th version of LeNet) and Yolov5, an architecture that uses ResNet.[13]

4.4.2 LeNet 5

LeNet5 was developed as the fifth version of LeNet,[17] a neural network designed by Yan LeCun in 1989. This architecture is designed for the MINST dataset, which aims to classify hand-written numbers from 1 to 9. However, the extent of applications is wide, even for traffic signal recognition[18]. The architecture comprises 8 layers, an input layer with two convolution layers, two pooling layers, two fully connected layers and the output layer (Fig. 8). The input image for this architecture is a 32x32 pixel image. The only issue that we had with applying this architecture in our device is that the size of the image has to be very small, therefore when the pollen grain size is big enough, high-value parameters like the spikes or the surface can be lost when applying a reduction to the size of the image.

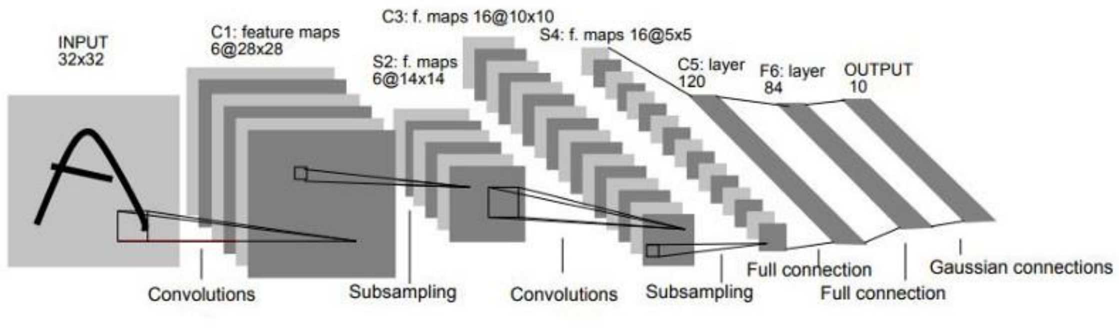


Fig.8: Architecture of LeNet-5.

4.4.3 Yolov5

"Yolo" stands for you only look once; this neural network family uses ResNet as architecture. The structure is much more complex (Fig.9) than the LeNet 5 one, and there are several options to apply: Yolov5n, Yolov5s, Yolov5m, and Yolov5l.[19] Corresponding to nano, small, medium and large. The name stands for the number of layers incorporated in the architecture; this affects the performance of the CNN. However, as we become more accurate, we lose speed. We have been using the Yolov5s during this project because we want a balance between fast detection and an accurate one. In summary, there exist three parts inside Yolov5:

Backbone: A convolutional neural network aggregates and forms image features at different granularities.

Neck: A series of layers to mix and combine image features to pass them forward to prediction.

Head: Consumes features from the neck and takes the box and class prediction steps.

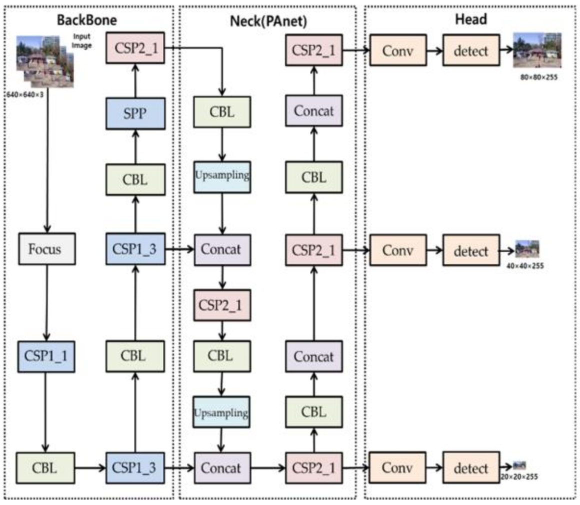


Fig.9: Structure of the Yolov5 with the three characteristic parts: Backbone, Neck, Head

4.4.4 Sidd Architecture

The last architecture we used during this project was designed by Siddharth Rawat, PhD student from Anna Wang's lab. With the knowledge of the different layers of Convolutional Neural Networks, he designed a 9-layer architecture that used Rectified Linear Unit as an activation function and used MaxPool as the pooling layer. The input of the image is 64x64. The details of the architecture are described in Fig. 10.

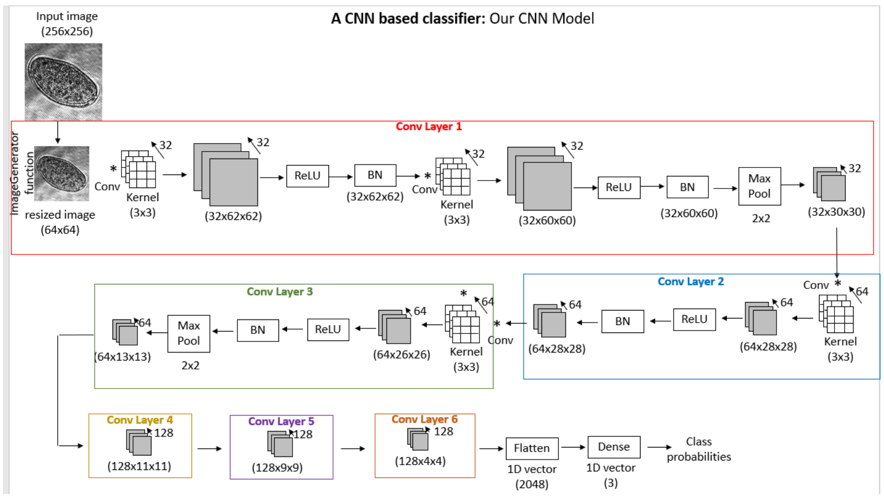


Fig.10: Scheme of the architecture designed by Siddharth. We can see how the different operations act on the image to obtain the final classification. Adding the input layer, flatter layer and the dense layer to the 6 convolutional ones we obtain the 9-layer architecture. Additionally, we can see how the different operations mentioned before are present during all the process

4.4.4 Data Augmentation

Since our time and resources are limited, data augmentation allows us to increase our dataset by introducing variations in our images. For example, by rotating the images randomly in 90º, 180º or 270º, changing the brightness, adding noise… This technique can make us increase our dataset from hundred images to even thousands of them.[14] Even though this technique is an excellent complement, it cannot substitute that the previous fieldwork to obtain the data has to be complete. If we can get a dataset with variety and as many situations as possible, we will assure better training. It is analogous to how our brain trains itself; if we only see one type of cat, for example, and a wide variety of tiger species, the day that we look at a different species of cat, we will probably recognize it as a tiger.

4.5 Sampling methods

To obtain our pollen samples, we extracted the Anther filaments from the flower. Then we put the anthers in PBS and vortex them for 10 seconds. Afterwards, we centrifugate for 15-20 mins. After this process, we throw away the excess filaments, add more PBS and centrifugate again for 10-15 minutes (Fig.11). When we had the solution prepared, we took five micro litters with the micropipette and put it in a slide in a chamber of silicon before taking a look into the microscope.

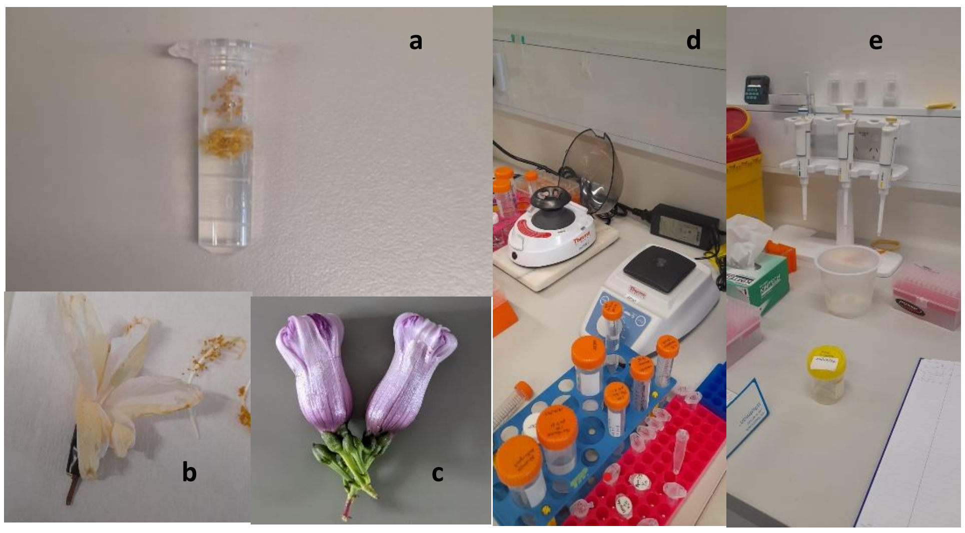


Fig.11: Sampling methods for obtaining the pollen samples in PBS. **(a)** Pollen grains conserved in PBS. **(b & c)** Flower previously to extract the pollen. (**d&e**) Laboratory material to process the pollen.

4.6 Optical path length

The refractive index exhibits the optical properties of the material. This parameter can give us an insight into any information contained in the material. For example, viscosity in phospholipids can be extracted by this property. When studying pollen, we tried to find a parameter that gives us high-quality information about the pollen. Because of the complexity and size of pollen grains, the refractive index was not a possible extraction feature with the techniques we were using. Holopy, could not handle non-homogenous spheres, so the extraction nor simulation of this parameter was not possible. Therefore, we tried a new extraction, the optical path length. To obtain this parameter from an unfocused hologram, we perform a propagation to obtain the focus image. From this propagation, we obtain the phase information and analyse it; the change of phase due to the change of refractive index of the pollen can be measured and extract the difference of phase.

5. Experimental Setup

As in developing the actual device for pollen classification. We design the structure of the hardware (Fig. 12) and how it connects to our software. As we said earlier, this device should be mobile and grant us real-time classification. The overall structure of the device was composed of three main parts:

1. In-line holography Microscope
2. Chamber for aerosol recollection
3. Software of the device

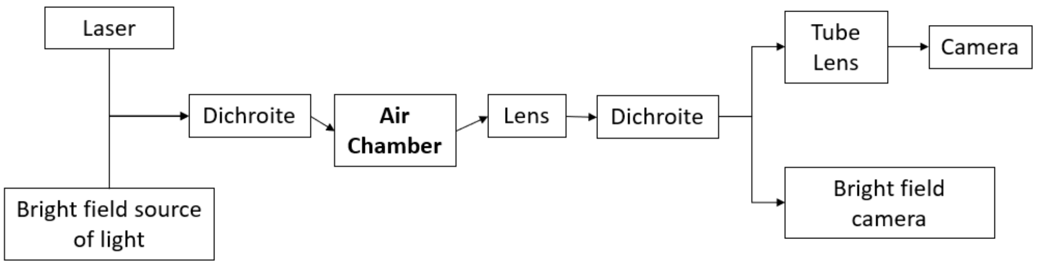


Fig. 12: Workflow of the Hardware of the device. Should correspond to the 1st and 2nd part of the parts of the device. This scheme comprises the holographic microscope and the chamber for aerosol recollection.

5.1 In-line holography Microscope.

This part is the most significant part of the hardware. An In-line holographic microscope comprises a coherent light source, a lens and a receptor (Fig. 13). It was also interesting to add a bright field microscope to make different types of observations if necessary. Therefore, using Dichroic mirrors, we could use the two microscopes simultaneously.

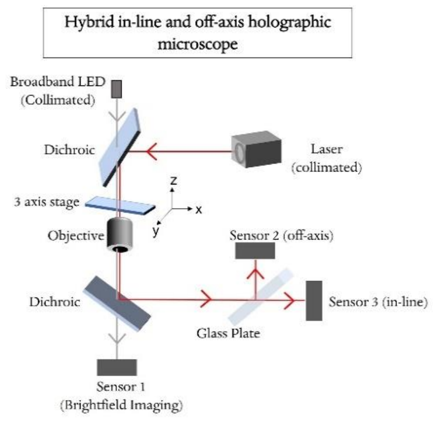
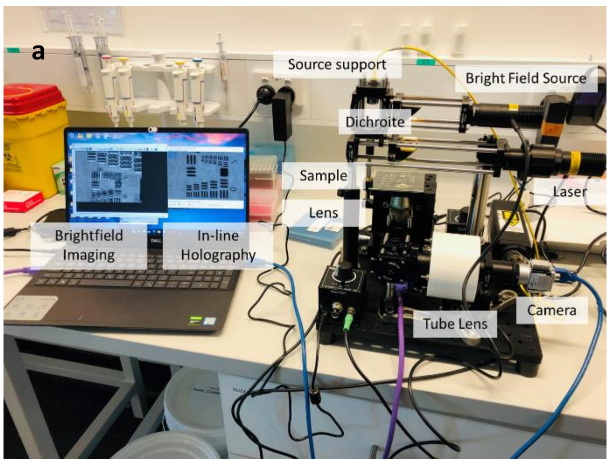

Hybrid in-line and off-axis holographic microscope
Broadband LED (Collimated)
Dichroic
Laser (collimated)
z
x
y
3 axis stage
Objective
Sensor 2 (off-axis)
Dichroic
Sensor 3 (in-line)
Glass Plate
Sensor 1 (Brightfield Imaging)

Fig. 13: In-line holographic microscope scheme

The Dichroic mirror only reflects a small range of wavelengths, and the other goes through the mirror. We used a Dichroic mirror that reflects a small range of around 660 nm to differentiate between the light from the bright field and the coherent light source.

For the Bright field, we used a standard white source LED that gave us the type of image that we wanted. For the coherent light source, we used a Laser of 660 nm. We used a 40x lens in this microscope and two cameras: Basler ace three basic and PCO. Panda 26. Before the holographic camera, we used a Tube Lens to augment the focal distance of the microscope, and in consequence, the walking distance of the microscope was increased. The hardware is showed in Fig. 14.



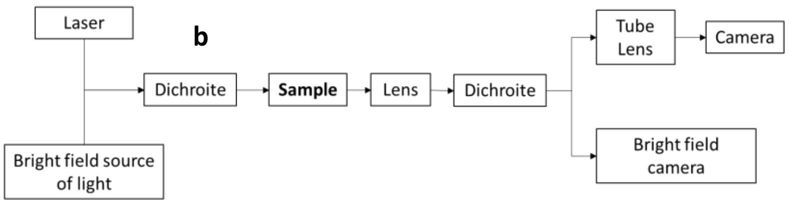


Fig.14: **(a)** Device ready for the detection with the two types of imaging, bright field and holography. **(b)** Schematic workflow of the light through the hardeare.

To obtain the actual magnification, we calculated this parameter by comparing the actual size of the image with the real size of the picture (Fig. 15). We obtained a magnification of 26.26 X as shown in equation 4.6:

1lp= 35 pixels= 7.81microns →0.223 microns/pixel

$$\text{FOV} = \frac{\text{Pixel\_size x no\_of\_pixels}}{\text{Magnification}} \qquad (4.6)$$

$$7.81 = \frac{5{,}86 \times 35}{\text{Magnification}} \rightarrow \text{Magnification} = 26.26\text{X}$$

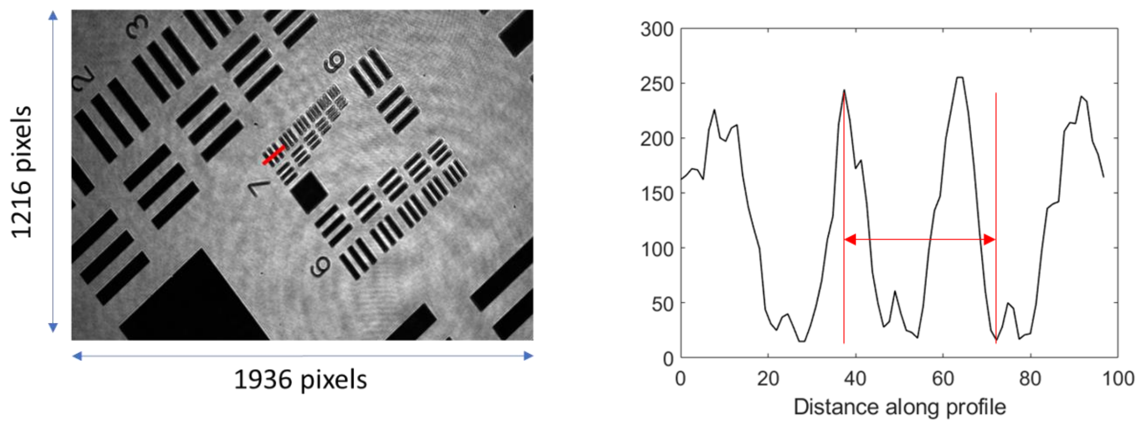


Fig. 15: Image taken to obtain the magnification of the microscope.

5.2 Chamber for aerosol collection

The idea of this project is to accomplish real-time pollen sensing. Therefore, we needed an airflow collector. Following state-of-the-art, these devices typically consist of a pump that pushes air through a chamber containing a sticky coverslip that traps pollen or the desired bioaerosols[20].

If we use this sticky coverslip, several inconveniences arise. For example, we need a rotating coverslip to take different samples over time, and the data we take are measurements in the overall time since we cannot see the airflow.

Another type of collector used, instead of a sticky coverslip, fluids to collect the aerosol, called Impingement, or other techniques as the Filtrators use porous substrate to trap the aerosol and separate the different particles.[21]

The design solution was an impactor composed of a chamber where the air flows by the action of a pump. Inside the chamber, we could use our microscope to capture images and processes them afterwards.

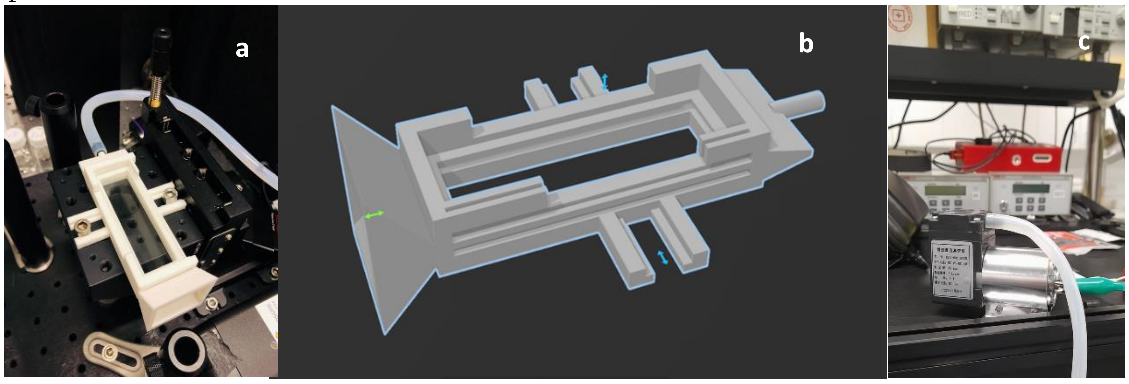


Fig.16: (a) Air chamber connected to the microscope ready to operate. **(b)** Model design in before using the 3-D printer **(c)** Air pump used to create the air flow into the chamber.

During the development of this project, we tried to use the chamber during several experiments, but problems arose because of the incompatibility of batteries with the air pump (Fig.16). Also, we used a 3D printer to build the piece, and the hardware piece presented different malfunctions. For that reason, we decided to discard the use of this piece in the present version of the dispositive.

5.3 Software of the device

The final part was the software. Two cameras were connected to our laptop, so we could obtain the holographic and the bright field Image. Apart from the required drivers to connect the two cameras, we needed a GUI to control our device. We wanted to apply our Neural Network to the images we get in real-time. Therefore, we used python to code the different functionalities of the device.

Pycromanager, cv2 and holopy were the main python libraries used in the project. Pycromanager had the functionality to control the program Micromanager, the main software controller for microscopes. Micromanager is an open source software capable of controlling microscopes. In this project, it was used to manage the exposure of the camera and different properties of the images as the intensity received. Pycromanager is the Python library that can control this program and with the *headless_mode* without actually dealing with the Micromanager GUI. Cv2 was a standard library to display and work with images.

The structure of the GUI is summarized in Fig. 18. Where we can see the different functions that took part in the software. 4 main functionalities were implemented:

1- Programming acquisitions: With this functionality you define that the devices take x images in the spam of time y and save them in the directory that you choose.

2- Live video of the microscope: With this functionality, live video of the microscope is launched. It can be closed and even by pressing ‘w’ you can capture an image and launch the functionality ‘Capture a Snap’.

3- Capture a Snap: Main functionality of the code. With this functionality, you capture and hologram and display it. With this image you can classify by executing the neural network, save the image in a directory or upload another image that you want to visualize.

4- Classify a set of Images. This functionality you can classify a set of files that you select from a directory. In Fig. 17 we can see the different functionalities in action:

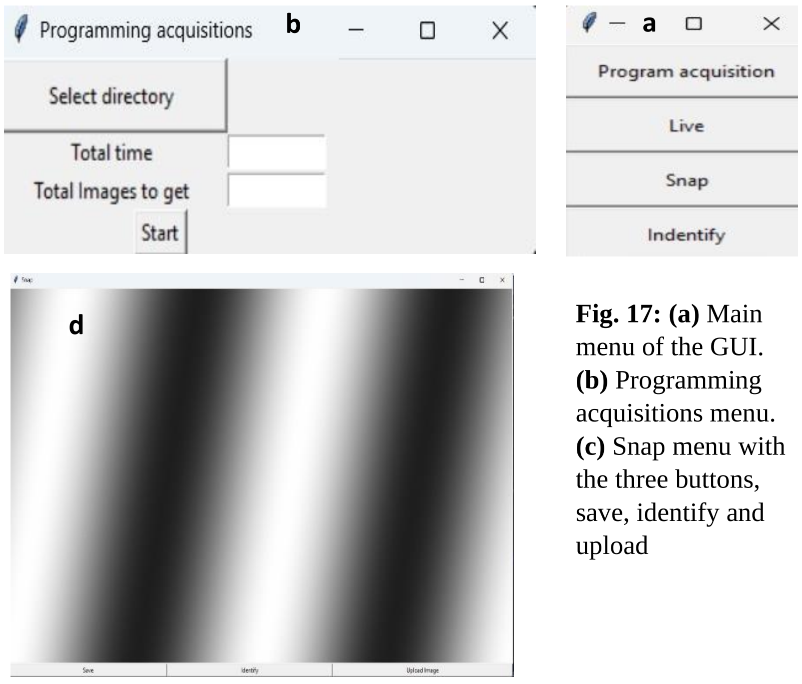


Fig. 17: (a) Main menu of the GUI. **(b)** Programming acquisitions menu. **(c)** Snap menu with the three buttons, save, identify and upload

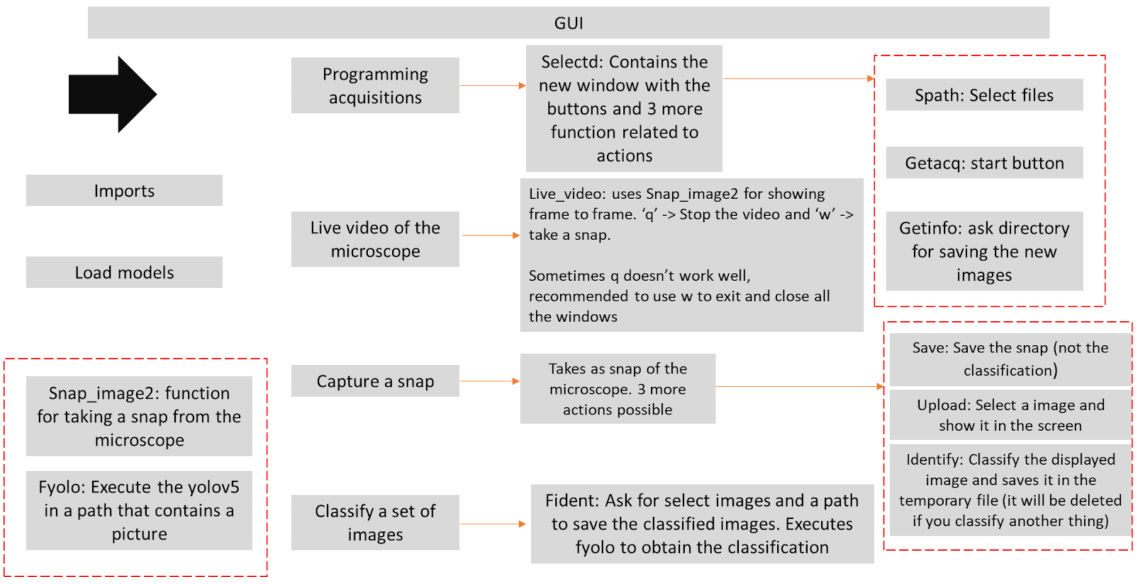

GUI
Imports
Load models
Snap_image2: function for taking a snap from the microscope
Fyolo: Execute the yolov5 in a path that contains a picture
Programming acquisitions
Live video of the microscope
Capture a snap
Classify a set of images
Selectd: Contains the new window with the buttons and 3 more function related to actions
Live_video: uses Snap_image2 for showing frame to frame. 'q' -> Stop the video and 'w' -> take a snap.
Sometimes q doesn't work well, recommended to use w to exit and close all the windows
Takes as snap of the microscope. 3 more actions possible
Fident: Ask for select images and a path to save the classified images. Executes fyolo to obtain the classification
Spath: Select files
Getacq: start button
Getinfo: ask directory for saving the new images
Save: Save the snap (not the classification)
Upload: Select a image and show it in the screen
Identify: Classify the displayed image and saves it in the temporary file (it will be deleted if you classify another thing)

Fig. 18: Main structure of the code for the GUI with each of the buttons explained. First starting with the imports and loading all the models, and then going to the menu with all the possibilities explained. Also there are two recurrence functions that are used through all the code

For training purposes, we used a google collab since we did not have enough computational power in our laptops. This google collab used the GitHub repository of yolov5 and adequately executed the python training code from the GitHub repository.

All the codes used for this project are in Appendix A, based on python because it is an open-source interpreter that anybody can use. Therefore, the software can be easily replicated.

6. Results and Discussion

This section will review all the results obtained from the different experiments developed in this project. We will go from the first test of our neural network to the alternatives we explore. Even though we have not performed extensive data work, we have tried different alternatives to recreate real-life pollen classification.

6.1 First steps in Neural Networks

During the first month of the project, we recollected from 3 flowers about 100 holograms each of them. These holograms were obtained by the sampling method described in section three. After cropping these images manually (Fig.19) to obtain a picture that our first architectures could handle, we started training the two architectures. The architectures were LeNet 5 and the first algorithm designed by Siddharth Rawat (a reference to both of them). The species of the flowers were:

- Morning glory
- Bush lily (red flower)
- Bush lily (white flower)

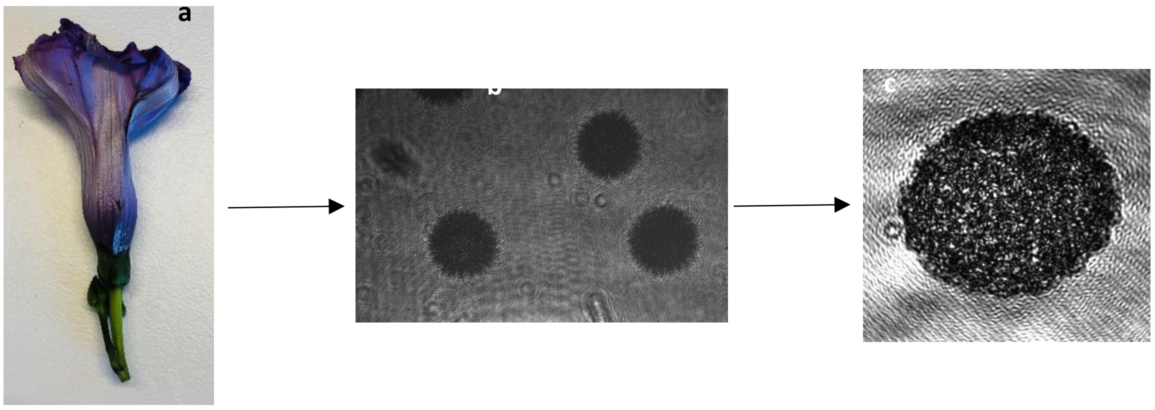


Fig. 19: **(a)** Raw flower from which we will extract the pollen **(b)** Hologram obtained in the microscope with different grain pollen **(c)** Cropped hologram for the neural network

We started to train our first neural network. We trained both algorithms (Talk about training algorithms). The difference between each of them was big, but both of the options were viable (Table 1):

	LeNet 5	Siddharth Architecture
Test Accuracy:	90.48%	90.00%
Train Accuracy:	100.00%	98.18%

Table 1: With the results of both phases of training and testing after the training. Siddharth was trained over 25 epochs with a batch size of 16, and LeNet 5 was trained over 400 epochs with a batch size of 8.

This first run was, unexpectedly, splendid a first view. Additionally, the time of classifying pollen was acceptable for real-life classification, but we did not consider several issues in this training. This training was performed with three very different species that was very easy to recognize the three of them, so that explained that we were capable of obtaining these results with such a small dataset for training our machine learning architecture.

6.2 Machine Learning Approach

After our first training, we took a second review of the state of the art of aerosol detection. Thanks to a second review of state of the art, a new idea came to our mind. This new idea came because we could optimise our work using Machine Learning with parameter extraction instead of Deep Learning. The idea was to use the refractive index and the radius to classify pollen since we could obtain both parameters using Holopy and by sizing the image by associating the actual distance with each pixel.

The refractive index could not be used because of the size and complexity of the pollen grains. Consequently, we changed the strategy and we tried to extract the Optical Path Length from different pollen samples. Therefore, we took a hologram and propagated using the Holopy to different distances to extract the phase of the hologram. For six pollen holograms, we extracted the information on the optical path length. The extraction is shown in Fig. 19 only for one hologram. However, it is the same for the six pollen holograms.

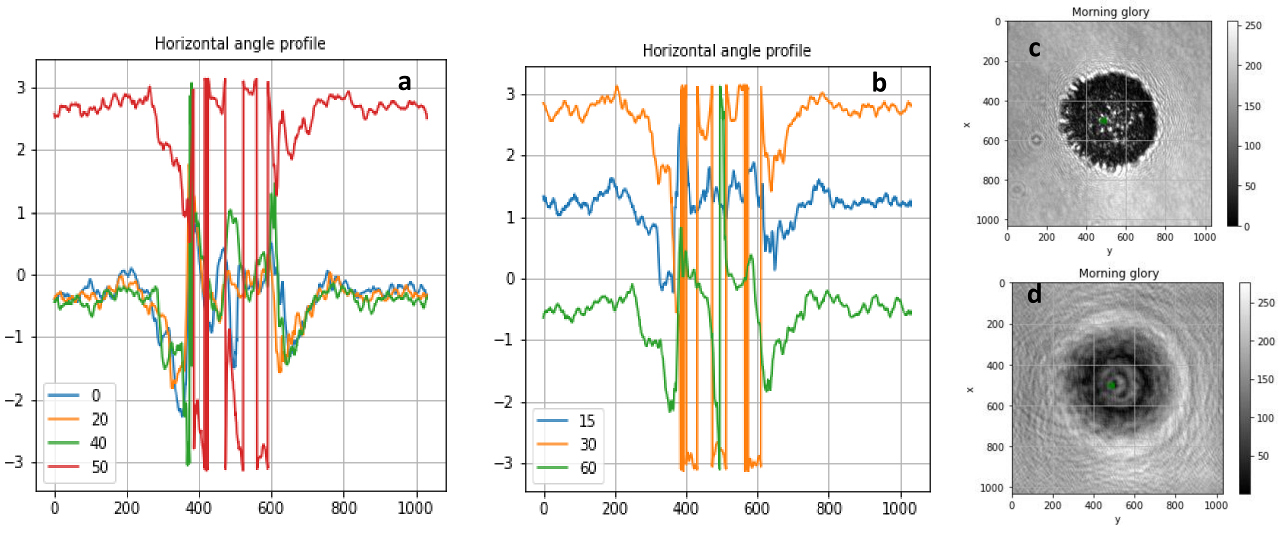


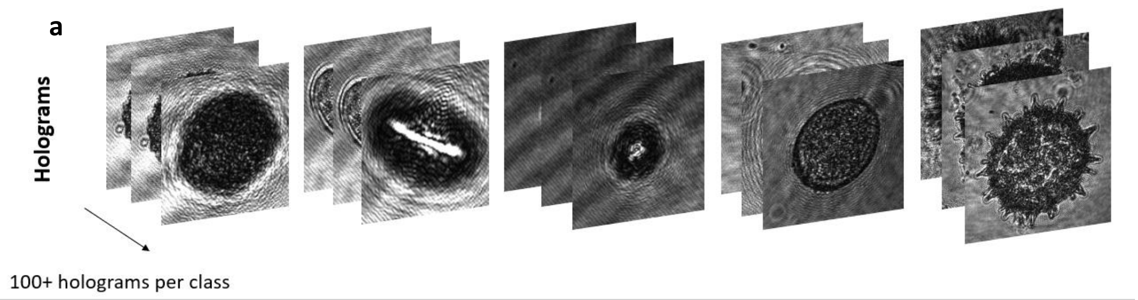
Fig.19: **(a) & (b)** In these images we see the angle information on the y axis and the space that goes through the middle (dot green in (c). The different lines correspond to different propagation distances. **(c)** Raw Morning glory hologram used for obtaining the information **(d)** Propagated morning glory hologram.

As shown in the figure, the phase information is wrapped in the image, but we cannot extract the optical path length. If we go into a more detailed scale of information, we can extract this information and take advantage of it. Therefore, because we needed help to get a deeper understanding of the information behind this approach, we discarded this classification option.

6.3 Dataset ampliation and Yolov5 implementation

After discarding the machine learning approach, we extend our dataset to 500 images by adding two new types of pollen: White Shoeblack pollen and Hibiscus pollen. With this new addition, we trained our two neural networks obtaining an accuracy similar to the prior training (90% in the testing) showed in Fig.20.

With this implementation, we started developing the hardware of the device, so in a parallel way to developing the hardware, we started thinking about the workflow of the device. We needed automatic cropping that allowed us to obtain the pollen and sizes our deep-learning architectures could handle.



b

TP: True Positive	FN: False Negative
FP: False Positive	TN: True Negative

c

Mean Average pollen Precision (mAP) = 98,4 %

Precision (TP/(TP+FP)) = 96,6 %

Recall (TP/(TP+FN))= 97,5 %

Fig.20: **(a)** Dataset with the five types of holograms. From left to right, morning glory, bush lily (red), white shoeblack, bush lily (white) and Hibiscus pollen **(b)** table with the different metrics to count detections in images **(c)** Main metrics obtained from the training, not testing has been perform.

Yolov5 was our solution. This neural network previously used in Anna Wang's group was the optimal path to crop the holograms and then pass them into our neural network's architectures. Therefore, the workflow of the device was defined in Fig. 21(a).

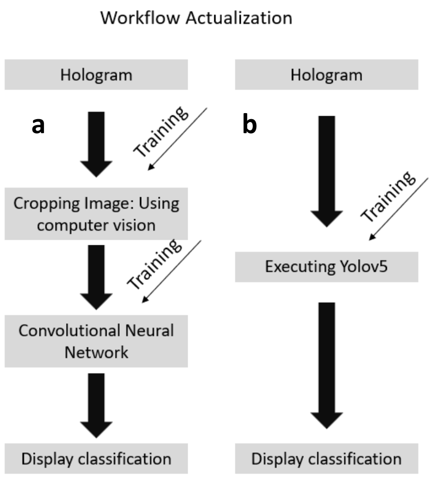


Fig.21: **(a)** Original workflow of the device **(b)** New workflow of the device

However, time was a concerning issue when trying to implement Yolov5 with LeNet 5. The total classification for a simple hologram image was 1 second without adding more features. When trying LeNet 5 for more than one pollen in an image could spend several seconds.

When we tried to implement our architecture, we found incompatibilities between the architecture by the laboratory and the processing image obtained by the microscope. The requirement for our laboratory is that the image size should be 64x64. Nevertheless, we obtained an image of size NxNx3—symbolizing the N number, the cropping size. This size can be reduced, so there is no problem. The thing is that Yolov5 gives a 3-channel image that cannot be reduced because when it is reduced loses part of the Intensity spectrum.

Additionally, when researching more about Yolov5 and understanding that this robust architecture could even differentiate between pedestrians crossing in the street.[22] We decided to optimize our neural network and use Yolov5 as our only neural network, which means that yolov5 was in charge of cropping the different pollen types and classifying them simultaneously.

This decision changed our workflow, as shown in Fig 21 (b). This change allows us to work also with bigger images and more complex holograms pictures. Consequently, new issues also arose. By not taking cropped images, mixing new images was something we should consider in the training of Yolov5. Hence, when we trained our new architecture with dataset 2.0 (Fig. 22), the testing when we tried different mixes was upsetting even though the training results were really good. We could not pass the 20% in the mean average precision.

27 Images in total from 3 mixes

TP	FN
12	10
FP	TN
69	1

Mix 1: Morning Glory I, Morning Glory II, Hibiscus

Mix 2: Morning Glory I, Bush lily, Hibiscus

Mix 3: Bush lily, White Flower

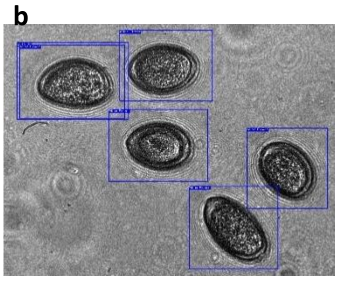
a

Mean Average pollen Precision (mAP) = 13.04 %

Precision (TP/(TP+FP)) = 14.81 %

Recall (TP/(TP+FN))= 54.55 %

b



c

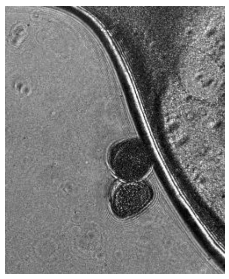


Fig. 22: Summary of the testing images with mixes. **(a)** Measurements taken and results obtained. **(b)** Typical mistake of the neural network, it overlaps two boxes instead of only putting one box. **(c)** Other typical mistake, no detection of pollen grains

Typical mistakes that commit our model, were to detect two times or even three the same pollen and with different classes, not detecting types of pollen was more residual error but there were some cases, and also not being able to distinguish between similar classes. Another factor that could interfere with the training is that the test images with different pollen mixes were taken three weeks after the training dataset was taken. Thus, we thought the mixed pollen was damaged, and the shape and characteristic spikes were modified.

6.4 Final training

With all said, we recreated the training to improve our results. We ampliated the dataset to 6 types of pollens. Moreover, we added new images with different mixes of pollen, so Yolov5 could get an insight into different mixes of pollen. And finally, we used data augmentation to increase our dataset from 300 images with more than 1000 pollen grains to 900 images with 3000 pollen grains (Fig.22). This training gave us much better results.

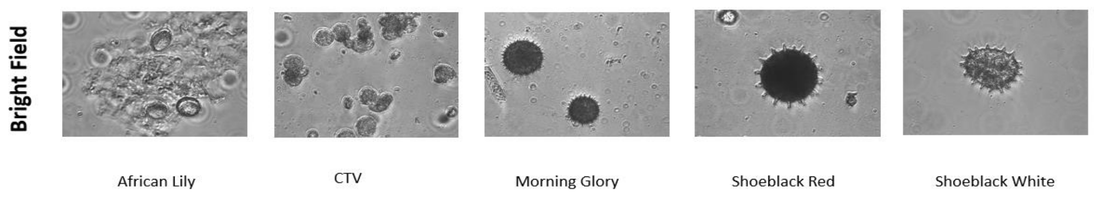


Fig.22: Five out of the six pollens. Images from the bright field, only left the white lily.

We got a Mean Average pollen Precision of 96%, a precision of 89,2% and a recall of 93%. This accuracy is acceptable for the training. Our neural network was ready to operate (Fig. 23). In appendix B we show different test images.

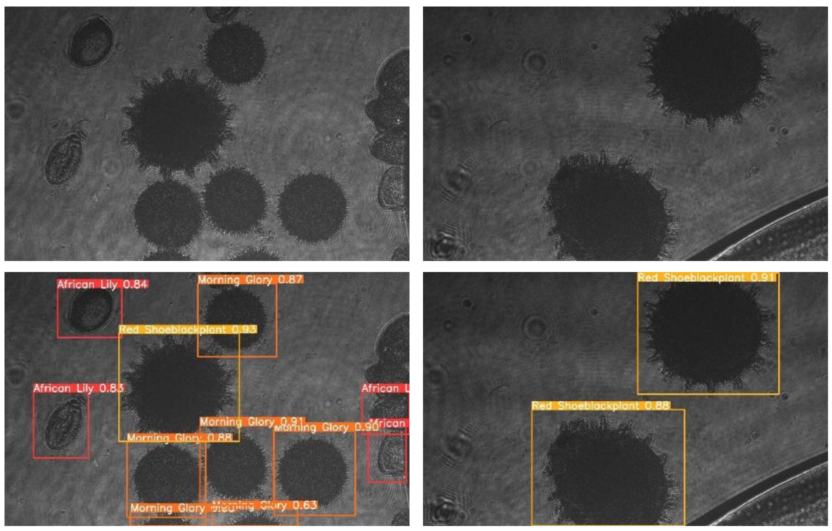
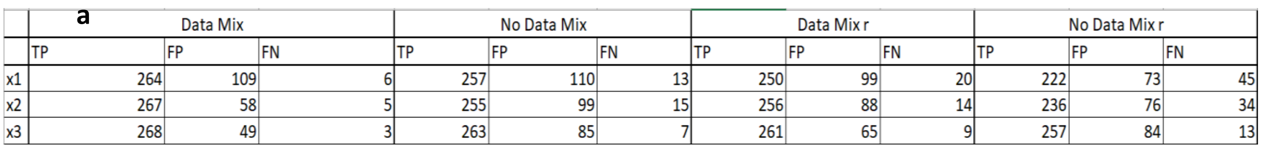


Fig.23: Two examples of pollen classification using our device. The different types of pollen are correctly classified and bounded using the characteristic boxes from Yolov5

It was understandable that the results in our test phase were determinate by the quality of our dataset. Therefore, we experimented with different types of datasets with different qualities to see how the test was affected. For the quality of the dataset, we consider that if we can get more images and the images with the most variety of mix, we get a higher quality dataset.

We created 12 datasets with different characteristics, so we could test how the Yolov5 was able to classify the testing images that remained still for the 12 of the datasets. The results indicated a correlation between the quality and the accuracy of the testing (Fig. 24).



a

	Data Mix			No Data Mix			Data Mix r			No Data Mix r		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
x1	264	109	6	257	110	13	250	99	20	222	73	45
x2	267	58	5	255	99	15	256	88	14	236	76	34
x3	268	49	3	263	85	7	261	65	9	257	84	13

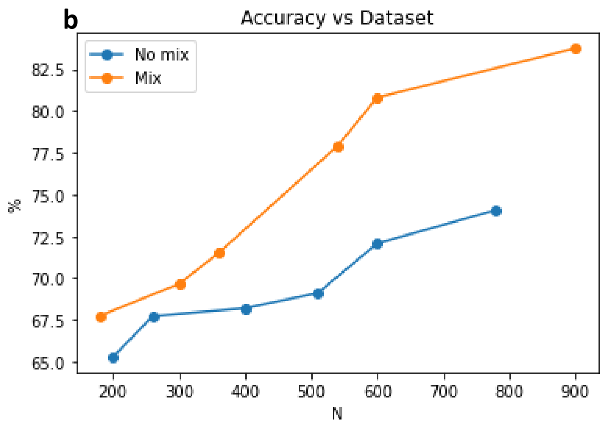


Fig.24: (a) Data counting's for each dataset, the 'r' stands for the different sizes when using data augmentation. **(b)** Final graph with the two lines corresponding to mix data and the no mixed data. We can see that there is a tendency to separate the increment of accuracy. Also, we see that for accuracy in test we have to increase the number of holograms used for the training.

Also, the increment of the True positives detected for each did not increase dramatically. However, the number of false positives differentiates one architecture from another. Another conclusion from the data is that mix and variety take more importance as the data gets higher. Additionally, we understand that we could get higher accuracies in the previous workflow with the two neural networks, but much more training has to be done. Therefore, depending on our objectives' accuracy, returning to the previous workflow would be better.

However, the results obtained for this testing for low-quality datasets are far from the ones obtained previously. This difference can be attributed to the damage to the sample previously explained. The most characteristic parts of the pollen grains are the shape, size and spikes; thus, by modifying some of these characteristics, the pollen seems completely different.

7. Conclusions and future outlook

During this project, we developed a mobile device capable of sensing pollen and classifying it in real-time. We have used the techniques of holography and machine learning to accomplish the objectives that we mentioned at the start of the project. We have identified different types of pollen in real-life using holography. The real-time sensing of pollen has been made partially. We did not need to process the information after the laboratory; we could do it simultaneously as we took the images. However, we still need to process the sample before taking any information. An objective before publishing the paper will be to obtain real-time sensing. Therefore, we will need to improve our chamber and make it functional. By implementing the chamber, since our device is already mobile, we will surpass the limitation of the mobile device. Finally, being as cheap as we can make a microscope, a holographic microscope is relatively inexpensive. It is a straightforward and powerful method that can give us much information.

The following limitation on this topic to surpass, will be making the device cheaper and automating its functioning. If this is accomplished, several applications can be possible. For example, it detects pollen in rooms and buildings to alert for possible allergies or measure pollen concentration outside. Other general future outlooks concerning the pollen are the modifications using optical traps.[23] During our studies, we found pollen kits in the pollen grains. This substance is responsible for sticking the pollen to the bees so pollination can occur. The pollen kits can be modified as we please, and in the future, we can modify the properties of the pollen grains. Modifying these properties can benefit the pollination and optimise this natural process. Finally, we will perform changes to the device before the end of this project. We plan to add an in-live classification video and improve our dataset with more types of pollen, especially those critical to human healthcare. Also, improve the neural network results to get a better classification.

8. References

[1] Barkley, S., Dimiduk, T.G., Fung, J., Kaz, D.M., Manoharan, V.N., McGorty, R.J., Perry, R.W., & Wang, A. (2018). *Holographic Microscopy with Python and HoloPy. Computing in Science & Engineering*, 22, 72-82.

[2] Kim KH, Kabir E, Jahan SA. (2018) *Airborne bioaerosols and their impact on human health*. J Environ Sci (China), 67, 23-35.

[3] Benjamin Swanson, Maxwell Freeman, Samir Rezgui, J. Alex Huffman. (2023) *Pollen classification using a single particle fluorescence spectroscopy technique.* Aerosol Science and Technology 57:2, 112-133.

[4] Johnston, Sean. (2003). *Reconstructing the history of holography*. Proceedings of SPIE - The International Society for Optical Engineering. 5005. 455-464.

[5] Canova C, Heinrich J, Anto JM, Leynaert B, Smith M, Kuenzli N, Zock JP, Janson C, Cerveri I, de Marco R, Toren K, Gislason T, Nowak D, Pin I, Wjst M, Manfreda J, Svanes C, Crane J, Abramson M, Burr M, Burney P, Jarvis D. (2013) *The influence of sensitisation to pollens and moulds on seasonal variations in asthma attacks.* Eur Respir J. 42(4):935-45.

[6] Allergy, A. s. (2022). *Pollen Alergy*. Ascia.1.1-3

[7] Benjamin Swanson, Maxwell Freeman, Samir Rezgui, J. Alex Huffman. (2023) *Pollen classification using a single particle fluorescence spectroscopy technique.* Aerosol Science and Technology 57:2, 112-133.
[8] Wu, Y., Calis, A., Luo, Y., Chen, C., Lutton, M., Rivenson, Y., … Ozcan, A. (2018). *Label-Free Bioaerosol Sensing Using Mobile Microscopy and Deep Learning.* ACS Photonics, 5:11, 4617–4627.

[9] McCutcheon JR, Grier DG. (2017). *Holographic characterization of contaminants in water: Differentiation of suspended particles in heterogeneous dispersions*. Water Res. 1; 122:431-439

[10] Aaron Yevick, Mark Hannel, and David G. Grier,(2014) *Machine-learning approach to holographic particle characterization*, Opt. Express 22, 26884-26890.

[11] Kim, M. K., & Kim, M. K. (2011). *Digital holographic microscopy Chapter 3: Principles of Holography*, Springer New York, 149-190.

[12] Barkley, S., Dimiduk, T.G., Fung, J., Kaz, D.M., Manoharan, V.N., McGorty, R.J., Perry, R.W., & Wang, A. (2018). *Holographic Microscopy with Python and HoloPy*. Computing in Science & Engineering, 22, 72-82.

[13] Ertam, F., & Aydin, G. (2017). *Data classification with deep learning using Tensorflow*. 2017 International Conference on Computer Science and Engineering (UBMK), 755-758.

[14]. O’Shea, K., & Nash R. (2015). *An Introduction to Convolutional Neural Networks.* Computing Research Repository (CoRR). 1-11.

[15] M. Hannel, A. Abdulali, M. O’Brien, and D. Grier. (2018). *Machine-learning techniques for fast and accurate feature localization in holograms of colloidal particles,* Opt. Express 26, 15221-15231.

[16] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proc. IEEE, 86, 2278-2324.

[17] Wenchao Cui, Qiong Lu, Asif Moin Qureshi, Wei Li & Kehe Wu (2021) *An adaptive LeNet-5 model for anomaly detection,* Information Security Journal: A Global Perspective, 30:1, 19-29

[18] Wenlong Li, Xingguang Li, Yueya Qin, Wenjun Song, and Wei Cui. (2020). *Application of Improved LeNet-5 Network in Traffic Sign Recognition. In Proceedings of the 3rd International Conference on Video and Image Processing* (ICVIP 2019). Association for Computing Machinery, New York, NY, USA, 13–18.

[19] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, (2016) *You Only Look Once: Unified, Real-Time Object Detection.* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016 pp. 779-788.

[20] Kevan, Peter G.; DiGiovanni, Franco; Ho, Rong H.; Taki, Hisatomo; Ferguson, Kristyn A.; Pawlowski, Agata K. (2006). *A Simple Method for Collecting Airborne Pollen*. Journal of Biological Education, v40 n4 p181-183.

[21] Shipe El, Tyler Me, Chapman Dn. (1959) *Bacterial aerosol samplers. II. Development and evaluation of the Shipe sampler*. Appl Microbiol.;7(6):349-54.

[22] Montenegro, Bryan and Flores-Calero, Marco. *Pedestrian detection at daytime and night time conditions based on YOLO-v5.* Ingenius. 2022, n.27, pp.85-95.

[23] Keir C. Neuman and Steven M. Block. (2004). *Optical trapping*, Review of Scientific Instruments 75, 2787.

9. Appendix A: Codes

GUI Code

```
################################################################################
##########################  CENTRAL CODE V.2  #################################
################################################################################
# ==============================================================================
# IMPORTS
# ==============================================================================
import tkinter as tk
from tkinter.filedialog import askdirectory, askopenfilename, asksaveasfilename, askopenfilenames
import os
from PIL import Image, ImageTk
import numpy as np
from pycromanager import Bridge, start_headless
import cv2
import time
from roboflow import Roboflow
import torch
# ==============================================================================
# SET UP
# ==============================================================================
# Here we could made a process were we check if there exists a temporal file and
# the different configurations that exist in the microscope (scanning all the
# cfg configurations)

# SELECT CFG FILE
# p1=askopenfilename(title='Select the configuration file',filetypes=[("Configuration file","*.cfg")],
#                  initialdir=r"C:\Program Files\Micro-Manager-2.0")
# # With a try to open always a path.
#
# ==============================================================================
# STARTING MICROMANAGER
# ==============================================================================
mm_app_path = r"C:\Program Files\Micro-Manager-2.0" # Path to micromanager
config_file = mm_app_path+"\MMConfig_demo.cfg" # Path to the configuration file
# Start the without entering the application
start_headless(mm_app_path, config_file, timeout=5000)
# What is the Bridge - Basically is executing the code as a class in order
# to be able to change exposure...
bridge = Bridge()
core = bridge.get_core()
core.get_version_info()
core.set_exposure(90)
# ==============================================================================
# LOADING CNN
# ==============================================================================
Ymodel = torch.hub.load(r'C:/Users/Juan Trius/yolov5','custom', r'C:/Users/Juan Trius/yolov5/best_DrNMx3', source='local')
# Temporal path 1
ptmp1=r'C:\Users\Juan Trius\Desktop\FDP-UNSW\Python Codes\CentralCoding\tmp\1.jpg'
# Temporal path 2
ptmp2=r'C:\Users\Juan Trius\Desktop\FDP-UNSW\Python Codes\CentralCoding\tmp2'
ptmp2n=r'C:\Users\Juan Trius\Desktop\FDP-UNSW\Python Codes\CentralCoding\tmp2\1.jpg'

# ==============================================================================
# DISPLAY THE FRAMES
# ==============================================================================
window =tk.Tk()
window.title("Pollen Classifier")
# ==============================================================================
# RECURRENCE FUNCTIONS
# ==============================================================================
# Function to get a snap

def snap_image2():
    # acquire an image and display it
    core.snap_image() # Acqure an image
    tagged_image = core.get_tagged_image() # Fetch the image data (buscar)
    # get the pixels in numpy array and reshape it according to its height and width
    image_array = np.reshape(
        tagged_image.pix,
        newshape=[-1, tagged_image.tags["Height"], tagged_image.tags["Width"]],
    )
    # for display, we can scale the image into the range of 0~255
    image_array = (image_array / image_array.max() * 255).astype("uint8")
    # return the first channel if multiple exists
    return image_array[0, :, :]

def fyolo(path):
   path2=ptmp2n
   # Ypred=Ymodel.predict(path, confidence=40, overlap=30)#.save(path2)
   Ypred=Ymodel(path)
   Ypred.save(exist_ok=True,save_dir=ptmp2)

   return path2
```

```
# =============================================================================
# LIVE VIDEO
# =============================================================================
# Function to start video
def live_video(event):
    cv2.namedWindow('Live, Press q to end the video',cv2.WINDOW_NORMAL)
    cv2.resizeWindow('Live, Press q to end the video', 1162,730)
    while(True):
        # Capture the video frame
        # by frame
        frame = snap_image2()
        # Display the resulting frame
        cv2.imshow('Live, Press q to end the video', frame)
        # the 'q' button is set as the
        # quitting button you may use any
        # desired button of your choice
        if cv2.waitKey(1) & 0xFF == ord('w'):
            snap_image(event)
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    # cv2.destroyWindow('Live, Press q to end the video')
    # Destroy all the windows
# =============================================================================
# FUNCTIONS THAT CREATES WINDOWS
# =============================================================================
# PROGRAMMING ACQUISITONS
def selectd(event):
    def spath(event):
        global path
        for widget in frame1.winfo_children():
            widget.destroy()
        path = askdirectory(title='Select directory to save the files')
        greeting = tk.Label(
                        master=frame1,
                        text='Path:'+path,
                        wraplength=500,
                        justify="left")
        greeting.pack()

    def getacq(N,T,path):
        # Here we should be able to get the different acquisitions
        print('Getting acquisitions')
        for i in range(N):
            im=snap_image2()

            cv2.imwrite(path+'\%d.jpg'%(i+1),im)
            a=time.time()
            time.sleep(float(T/N))
            b=time.time()
            print(b-a)
            print('Supposed time: %d'%(float(T/N)))
            # Classify them?
        print('Ok! Just finished :)')

    def getinfo(event):
        global N,T,path
        T=ntime.get()
        N=nimages.get()
        # Error handling
        try:
            T=int(T)
            N=int(N)
            if os.path.exists(path) == False:
                raise ValueError('No directory')
            winSnap.destroy()
            getacq(N,T,path)
        except ValueError:
            tk.messagebox.showerror('Error', 'Error: Invalid input data')
        except NameError:
            tk.messagebox.showerror('Error', 'No directory defined')
    winSnap =tk.Toplevel()
    winSnap.title("Programming acquisitions")
    # Frame to reload
    frame1 = tk.Frame(master=winSnap,relief=tk.RAISED,borderwidth=1)
    frame1.grid(row=0,column=1)

    # Button to select path
    button = tk.Button(master=winSnap,text="Select directory",height=2,width=20)
    button.bind("<Button-1>", spath)
    button.grid(row=0,column=0)

    # Time
    ntime = tk.Entry(master=winSnap,width=10)
    ntime.grid(row=1,column=1)

    Times = tk.Label(master=winSnap,text='Total time',wraplength=500,justify="left")
    Times.grid(row=1,column=0)
    # Number of pictures
    nimages = tk.Entry(master=winSnap,width=10)
```

```
# ==============================================================================
# FUNCTION FOR EXECUTING OUR NEURAL NETWORKS TO DIFFERENT ACQUISITIONS
# ==============================================================================
# Ask for open different different files askopenfiles save all the images in
# list, and every one of them separate them and execute them. Also ask where
# save the images
def findent(event):
    print('Identifying')
    try:
        paths=askopenfilenames(title='Select the files Identify',filetypes=[("Jpg","*.jpg"),("Tif",'*.tif')])
        savepath=askdirectory(title='Directory where to save the images')
    except:
        tk.messagebox.showerror('Error', 'No files or path detected')
        return
    i=1
    for path in paths:
        p2=fyolo(path)
        ident=cv2.imread(p2)
        cv2.imwrite(savepath+'\%d.jpg'% i,ident)
        i+=1
    print('Process Finished')
    return

# ==============================================================================
# SNAP A NEW IMAGE
# ==============================================================================
def snap_image(event):

    global imgp
    def Iupload(event):
        global frame, imgp
        try:
            path=askopenfilename()
            frame=cv2.imread(path)
            cv2.imwrite(ptmp1,frame)
            # Made the code autosustainable / No users/Juan Trius - Use r'.\Whatever\You\Want'

            imgp = Image.fromarray(frame)
            # img = ImageTk.PhotoImage(imgp.resize((968,608)))
            img = ImageTk.PhotoImage(imgp.resize((1162,730)))
            label1.configure(image=img)
            label1.image = img
            label1.grid(row=0,column=0,columnspan=3,rowspan =3)
        except:
            tk.messagebox.showerror('Error', 'No hologram detected')

    def Isave(event):
        try:
            path = asksaveasfilename(title='Save as',defaultextension='.jpg')
            cv2.imwrite(path, frame)
        except:
            tk.messagebox.showerror('Error', 'Wrong directory')
        # Save the image here

    def Identify(event):
         global pred
         print("Identifying")
         pathfile=ptmp1
         p2=fyolo(pathfile)
         ident=cv2.imread(p2)
         cv2.namedWindow("Identification", cv2.WINDOW_NORMAL)
         cv2.resizeWindow("Identification", 1162,730)
         cv2.imshow("Identification",ident)
         cv2.waitKey(0)
         cv2.destroyAllWindows()

    # Create a new window
    frame=snap_image2()
    cv2.imwrite(ptmp1,frame)
    # k=IWindow(frame)
    winSnap =tk.Toplevel()
    winSnap.title("Snap")
    # winSnap.geometry("700x700")
    # y=np.shape(frame)[1] + 20
    # x=np.shape(frame)[0] + 20
    # winSnap.geometry("%dx%d" %(x,y))
    imgp = Image.fromarray(frame)
    img = ImageTk.PhotoImage(imgp.resize((1162,730)))
    label1 = tk.Label(master=winSnap,image=img)
    label1.image = img
    label1.grid(row=0,column=0,columnspan=3,rowspan=3)
    # 1 Save
    b1 = tk.Button(master=winSnap,text="Save")
    b1.bind("<Button-1>", Isave)
    b1.grid(row=3,column=0, sticky="news")

    # Identify image
    b2 = tk.Button(master=winSnap,text="Identify")
    b2.bind("<Button-1>", Identify)
```

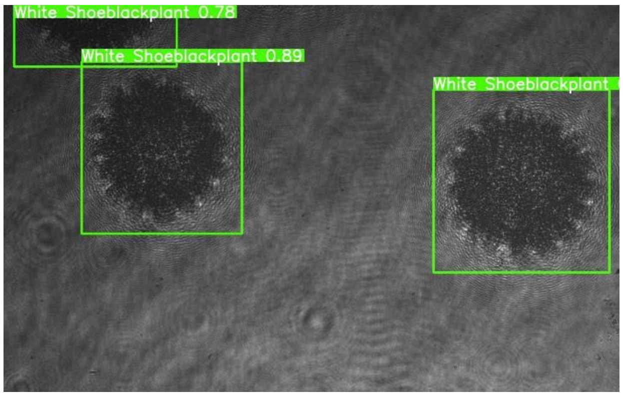
```
# ==============================================================================
# MAIN MENU
# ==============================================================================
a1=os.getcwd()
button = tk.Button(master=window,text="Program acquisition",height=2,width=20)
button.bind("<Button-1>", selectd)
button.grid(row=0,column=0)
# The only problem here is that I cannot have the two types of acquisition at
# the same time. The good thing is that when taking a snap we can go to another
# screen so we can actually do things in that context. Add to the sanp image
# window the different buttons related to the identification of the holograms

live = tk.Button(master=window,text="Live",height=2,width=20)
live.bind("<Button-1>", live_video)
live.grid(row=1,column=0)

snap = tk.Button(master=window,text="Snap",height=2,width=20)
snap.bind("<Button-1>", snap_image)
snap.grid(row=2,column=0)

Identify = tk.Button(master=window,text="Indentify",height=2,width=20)
Identify.bind("<Button-1>", findent)
Identify.grid(row=3,column=0)

# ==============================================================================
window.mainloop()
# ==============================================================================
```

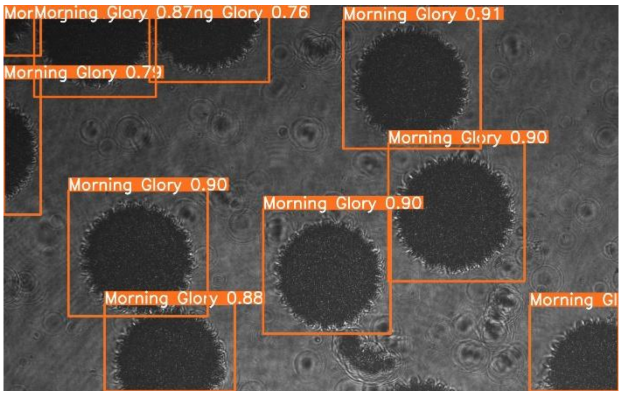
JupyterNotebook for the trainning:

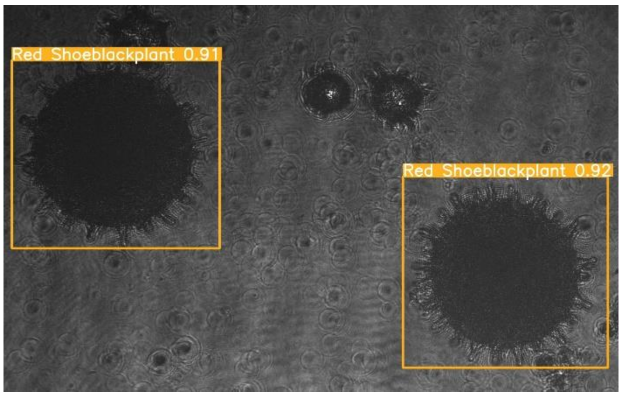
```
# -*- coding: utf-8 -*-
Automatically generated by Colaboratory.
Original file is located at
   https://colab.research.google.com/drive/1KHVkoZFPFKLB37EhWt1v6XKjG535nRXP
# Custom Training with YOLOv5

# Step 1: Install Requirements
"""
# Commented out IPython magic to ensure Python compatibility.
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output  # to display images

print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'})")

from roboflow import Roboflow
rf = Roboflow(model_format="yolov5", notebook="ultralytics")

# set up environment
os.environ["DATASET_DIRECTORY"] = "/content/datasets"

#after following the link above, recieve python code with these fields filled in
from roboflow import Roboflow
rf = Roboflow(api_key="7bljrtvTZ41FEWa6jjL0")
project = rf.workspace("pollen-classification").project("pollen6")
dataset = project.version(4).download("yolov5")

"""# Step 3: Train Our Custom YOLOv5 model
!python train.py --img 1216 --batch 16 --epochs 150 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache
"""#Run Inference  With Trained Weights
Run inference with a pretrained checkpoint on contents of `test/images` folder downloaded from Roboflow.
"""
!python detect.py --weights runs/train/exp/weights/best.pt --img 1936 --conf 0.1 --source {dataset.location}/test/images
#display inference on ALL test images
import glob
from IPython.display import Image, display
for imageName in glob.glob('/content/yolov5/runs/detect/exp3/*.jpg'): #assuming JPG
   display(Image(filename=imageName))
   print("\n")


#export your model's weights for future use
from google.colab import files
files.download('./runs/train/exp7/weights/best.pt')
```

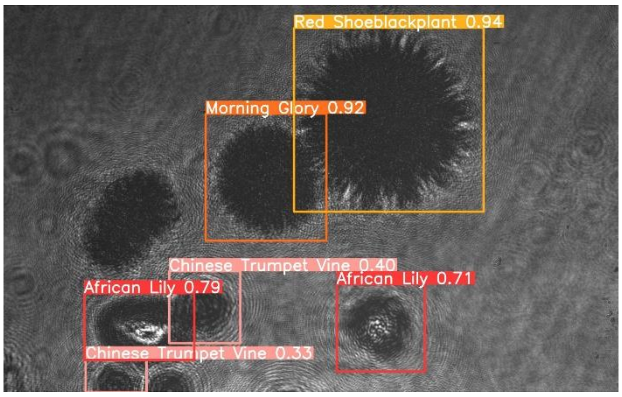
10. Appendix B: Test Images

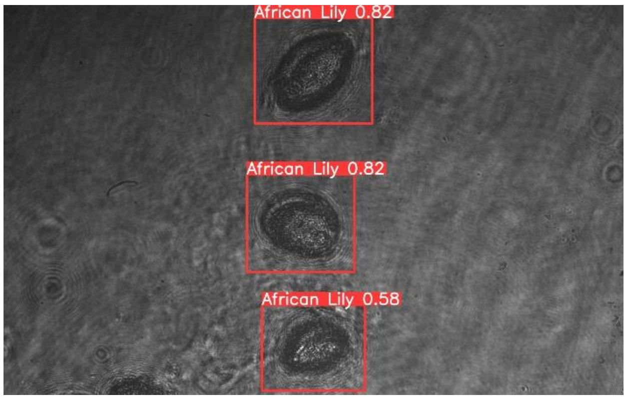
In this Section we share different images from the test results. These images aim to show the results obtained in the last training. We can see different results, images that are recognized perfectly, others with mistakes and images with pollen grains not recognized.

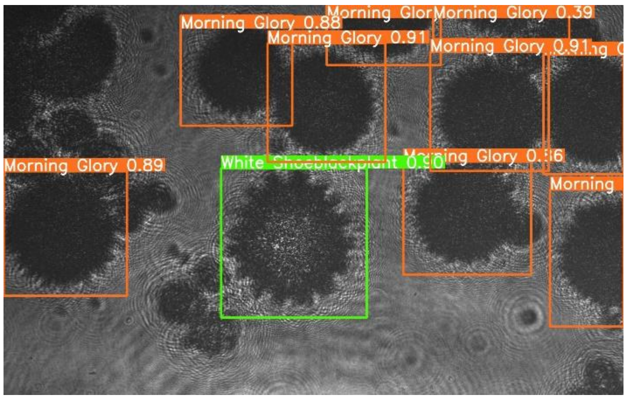


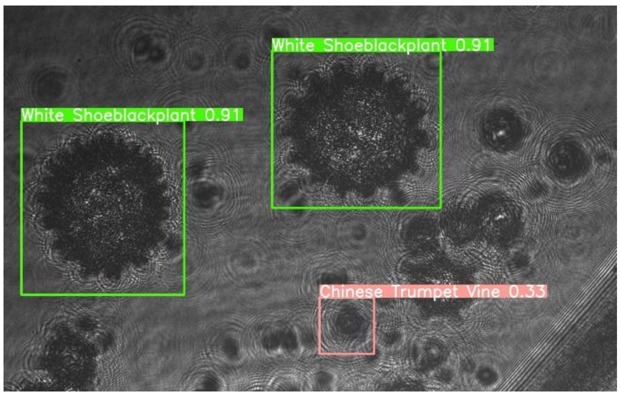


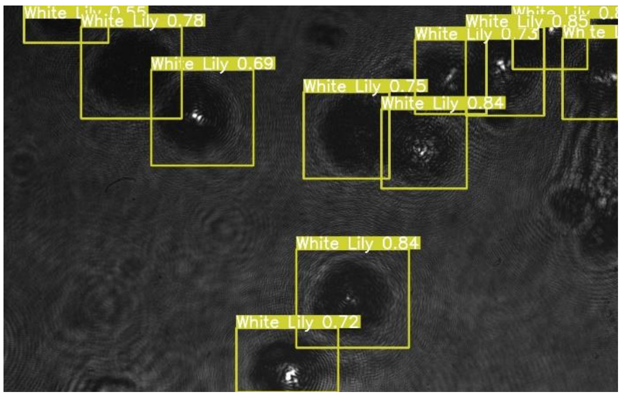


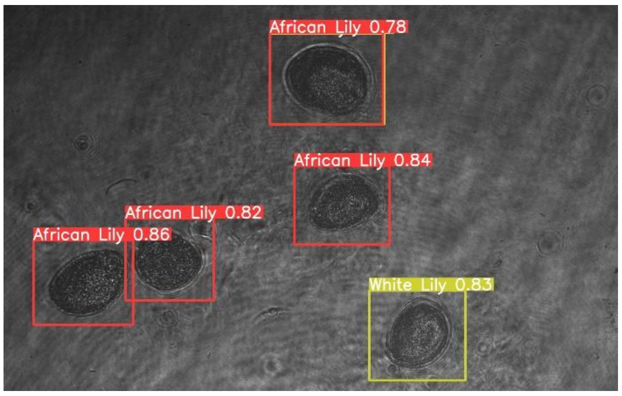


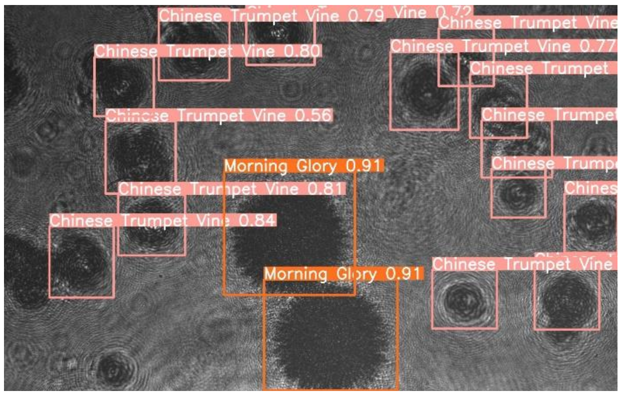


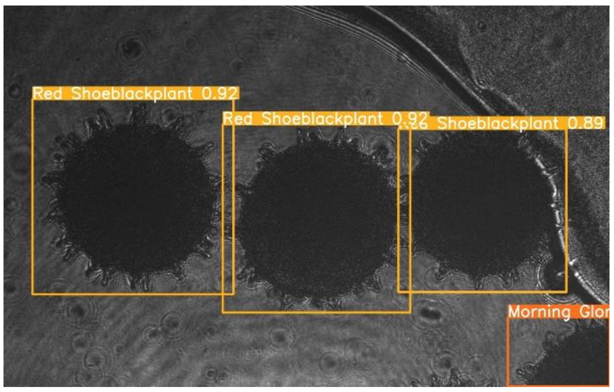


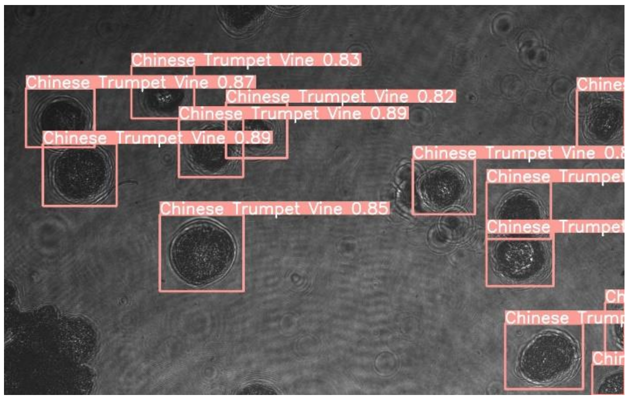

White Shoeblackplant 0.91
White Shoeblackplant 0.91
Chinese Trumpet Vine 0.33


White Lily 0.55
White Lily 0.78
White Lily 0.69
White Lily 0.75
White Lily 0.84
White Lily 0.73
White Lily 0.85
White Lily 0.84
White Lily 0.72


African Lily 0.78
African Lily 0.84
African Lily 0.82
African Lily 0.86
White Lily 0.83


Chinese Trumpet Vine 0.79
Chinese Trumpet Vine 0.80
Chinese Trumpet Vine 0.77
Chinese Trumpet Vine 0.56
Morning Glory 0.91
Chinese Trumpet Vine 0.81
Chinese Trumpet Vine 0.84
Morning Glory 0.91


Red Shoeblackplant 0.92
Red Shoeblackplant 0.92
Shoeblackplant 0.89


Chinese Trumpet Vine 0.83
Chinese Trumpet Vine 0.87
Chinese Trumpet Vine 0.82
Chinese Trumpet Vine 0.89
Chinese Trumpet Vine 0.89
Chinese Trumpet Vine 0.85