Graz University of Technology

Software program for data evaluations based on Python language

Internship report

From 22/04 to 26/07 2024

Academic supervisor:
Basile Dufay
Professor at ESIX Caen

Zaria Ferté
Student in second year at ESIX Normandie in Embedded systems

Supervisor:Torsten Mayr
Associate Professor at TU Graz University

Table of contents

01 List of abbreviations

- FLIM : Fluorescence Lifetime Imaging Microscopy
- ROI: Region Of Interest

02 List of figures

03 Acknowledgements

I would like to thank Torsten Mayr for entrusting me with this project. I thank Bernhard Müller for facilitating my integration into the institute and for answering all my questions about the internship. I thank everyone at the institute of analytical chemistry and food chemisty for their warm welcome and their assistance whenever I needed it.

04 Introduction

From April 22 to July 26, 2024, I completed an internship as an assistant engineer at the Institute of Analytical Chemistry and Food Chemistry located at the University of Graz in Austria.

Professors, PhD students, master's students, researchers, and technicians work at this institute. The research topics are diverse: the staff studies sensors based on optical transduction, the synthesis and characterization of new luminescent indicators, the fabrication of high-performance detection materials, and optical sensors, among others. Within this dynamic university, I was able to discover the luminescence of materials, a field I had never encountered before.

This report presents how this internship allowed me to become familiar with a real engineering project, particularly in the field of luminescence. I will describe the missions that were assigned to me, the different steps to complete my tasks and the problems encountered, the technical skills I was able to develop, and the lessons learned from this professional experience.

05 Roles and responsibilities

Within the institute, many people work or study the luminescence of certain materials to develop optical sensors. Knowing the lifetime of a material's luminescence allows the characterization of its physical properties such as pH, temperature, glucose level, etc. This improves sensor reliability and performance.

To facilitate the study of luminescence lifetime, the university purchased a camera from the brand “PCO”, specialized for this study. However, using this camera is quite complex, and the only software that offers an interface to use the camera is a software developed by Nikon: ‘Nis Elements’. This software offers many features, notably for managing electronic microscopes, and therefore the license for this software is very expensive for the university each year. The researchers and PhD students who use this camera only utilize a small part of this software and do not need the other features. Thus, my task during this internship is to develop software similar to the Nikon software, focusing only on the necessary features and adding those missing from the Nikon software.

I have broken down my work into four main steps:

1. Information
2. Data acquisition
3. Data processing
4. User interface

Goal:

I have listed each of the features to be developed and/or included in the interface in order of priority:

- Configure the camera
- Capture images
- Display graphs (Intensity, phase, modulation)
- Obtain and display graph data (minimum, maximum, average)
- Live visualization
- Save the last entered parameters to display them on startup
- Select a region of interest (ROI) on each graph
- Obtain and display data for this specific region only
- Background correction (calibration and correction)
- Apply filters to improve the image (Wiener and Blur)
- Reduce the graph display scale so that the colors are more indicative around the value of interest
- Save graphs as images
- Export data table in Excel format
- Display the luminescence lifetime for the selected area
- Calibration of image processing using a known component
- Fitting: obtain certain parameters (pH, temperature, oxygen level, etc.) based on the luminescence lifetime (establish the characteristic curve)
- Create an executable

06 Management

I was autonomous in managing the organization of my work. I had regular meetings with Torsten (my supervisor) and Bernhard to explain the progress of the project. These meetings were an opportunity for me to present the points where I was stuck, ask my questions, and also for them to indicate the tasks to add gradually. After discussions with Torsten, we decided that I would start by implementing the basic features to complete the four steps for just a few functionalities. This was intended to provide a functional software, even if there wasn't enough time to implement all the desired features.

Outside of these meetings, I was free to organize myself as I wished to advance on this project. I used the Trello application to plan my tasks for each stage of the project. I noted the tasks in progress, those that were completed, and those that remained to be done for each step. I also indicated the steps that were blocking me and the questions I needed to ask. This allowed me to clearly explain to my supervisor where I was in the project.

I worked on my personal computer, used Spyder on Anaconda for programming, and went to the lab to test my programs where the camera was located. I used GitHub to keep a record of my work in case I needed to go back or encountered a bug.

07 Display of 3 graphs

1-Informations

The first step was quite long because I had to learn about a concept I had never encountered before: luminescence.

- Understanding the Basic Aspects of Luminescence

Luminescence is the emission of light by a material without it being heated. Atoms or molecules of the component absorb energy and re-emit it in the form of light. This reaction can occur after different phenomena: light, electricity, or a chemical reaction. The luminescence lifetime is the duration during which the component emits photons after being exposed to one of the phenomena aforementioned.
In my case, the component is illuminated by a pulsed light for which we can choose the wavelength, signal shape, and frequency using an Omicron LED device. In response to this excitation, the component emits photons at a different wavelength. This response is what we observe and allows us to measure certain quantities such as phase shift, modulation, or intensity as represented in Figure 01:

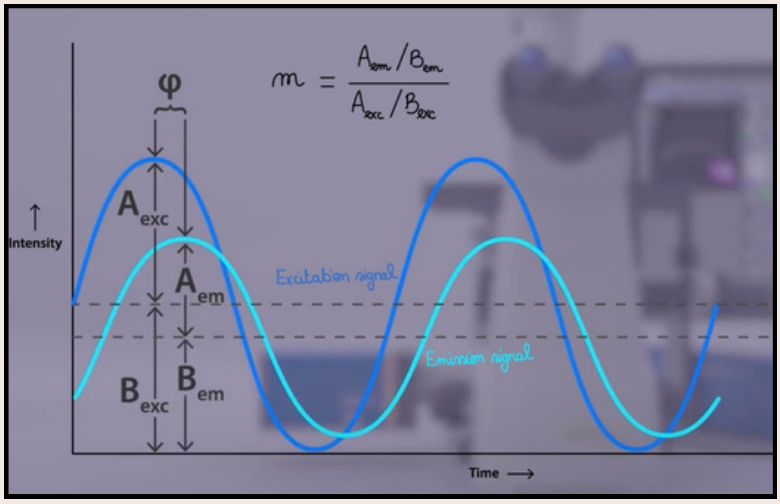


FIGURE 01- DATA EXPLANATION GRAPH

The quantities Φ (phase angle) and m (modulation) are related to the luminescence lifetime (τ) by the following formulas:

$$tan(\Phi) = 2\pi f\tau$$
$$m = \frac{1}{\sqrt{1 + (2\pi fm\tau)^2}}$$

Thus, to determine the luminescence lifetime of a component, it is crucial to accurately determine its phase, intensity, and modulation.

- Analysis of Nikon Software

The software I had to develop needed to replicate what the Nikon software already did. Therefore, I was shown how to use the software and the functionalities that were important for my software to implement.

Key Functionalities to Implement:

- Parameter Selection
- Live Visualization to position the component under the camera
- Display of Three Graphs: Intensity, phase, and modulation with a color scale
- Display of Minimum, Maximum, and Average Values for each graph
- Ability to Select a Region of Interest (ROI) on the graph to obtain data

- Reading and Analyzing Technical Documentation

Finally, I gathered information about the camera I was going to use. The PCO camera is specialized in capturing the luminescence lifetime and has essential features for studying luminescence. I informed myself by reading all the documentation provided by the manufacturer.

The camera uses a CMOS (Complementary Metal Oxide Semiconductor) sensor with 2 taps, which means it can capture images more quickly. The camera can capture 45 double images per second with a resolution of 1008 pixels. It sends a modulation signal to the Omicron LED device and also receives a signal to synchronize its image capture with the modulation signal.

FIGURE 03 : CAPTURE

There are several ways to capture images, influencing the precision of the results:

Either the two taps are used alternately, or only one of the two is used to capture images. The two taps can have different sensitivities, allowing us to choose to correct image symmetry by capturing each image with both taps.

We also choose the number of images that will be captured during a modulation cycle: the more images we capture, the more precise the received signal will be. For example, if we choose to capture 4 phases for each cycle of the modulation signal, it means we will capture images at moments when the modulation signal is at 0, 90, 180, and 360 degrees. The modulation signal is a sinusoidal signal. Figure 03 illustrates when images are captured.

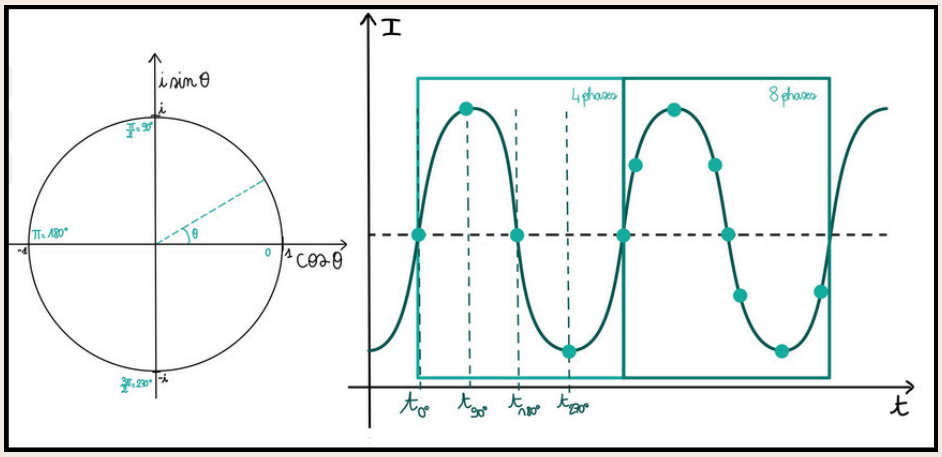
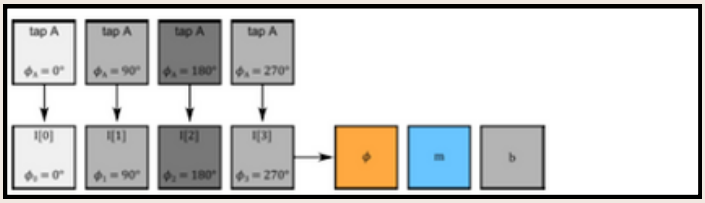


FIGURE 03- RELATION BETWEEN MODULATION SIGNAL AND PHASE

Figure 04 shows two examples of image processing based on the desired parameters. In the first case, only one tap is used to capture 4 phases, so the phases will be captured in the order of 0, 90, 180, and 270 degrees. In the second case, both taps are used, and we want to correct the asymmetry of the two taps by capturing each image twice with each tap. Thus, during the first modulation period, tap A will capture the 0-degree phase and tap B the 180-degree phase, and so on. Finally, we take an average of each image captured at 0 degrees and similarly for the other phases to obtain images corresponding to the 4 phases.

Parameter	Value
Phase Number	[4 phases]
Phase Symmetry	[twice]
Phase Order	[ascending]
Tap Select	[tap A]
Asymmetry Correction	[off]



Parameter	Value
Phase Number	[4 phases]
Phase Symmetry	[twice]
Phase Order	[opposite]
Tap Select	[both]
Asymmetry Correction	[off]

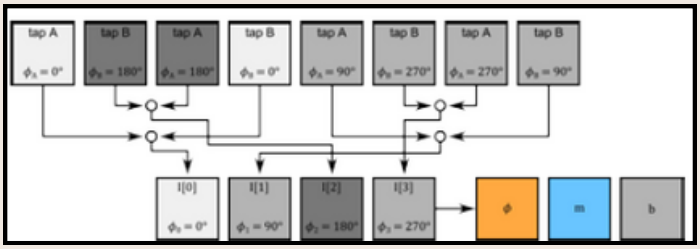


FIGURE 04- WAY OF ACQUISITION ACCORDING TO DIFFERENT PARAMETERS

Ultimately, each of these parameters influences the number of images to capture and the order of capture.

I also searched for libraries to facilitate the use of the PCO camera. Indeed, there was a library provided by the manufacturer, so I analyzed it to be able to use it effectively.

It offers two main classes:

- Camera Class: This is primarily used to initialize the camera parameters mentioned earlier, set the modulation signal parameters, start an acquisition, and retrieve all captured images.
- Flim Class: This class allows obtaining data arrays corresponding to phase, intensity, modulation, and phasor data for each pixel of the image.

The camera is connected to the computer via USB, and the communication protocol between the two is established within this library, so I did not need to implement it in my code.

Once I had all the information about the general operation of the camera, I had a comprehensive idea of what needed to be done and could begin the second step.

2- Data Acquisition

The second step involved communicating with the camera to retrieve data for analysis. This was a complex part because I encountered difficulties in finding the necessary information, often due to the very brief documentation of the Python library.

Problems Encountered :

- Alternation of Taps
- Synchronization with the Modulation Signal
- Camera Settings
- Frame Management

The documentation indicated that depending on the chosen parameters, the sensor's taps must alternate to capture images and that each tap must capture each phase. However, I didn't find any function in the libraries to alternate these taps within the program.

Moreover, my program had to be perfectly synchronized with the modulation signal to capture images exactly at the correct phases. I initially thought that the synchronization between the modulation signal and the moment the image was captured had to be managed within the program, meaning the execution time of each function had to be minimized to maintain this synchronization. Ultimately, this synchronization is directly managed by the camera, which sends a signal to the module generating the modulation signal at the start of each acquisition. Therefore, synchronization is handled by a hardware cable between the two devices.

There were sometimes inconsistencies between the documentation and the actual behavior of the code, likely due to code updates. The documentation clearly stated that to capture 4 phases while correcting for asymmetry, 8 images were needed (one for each tap for each phase as shown in Figure 05). However, it did not specify whether the averaging was done internally or if I had to capture 8 images and average them in the correct order myself.

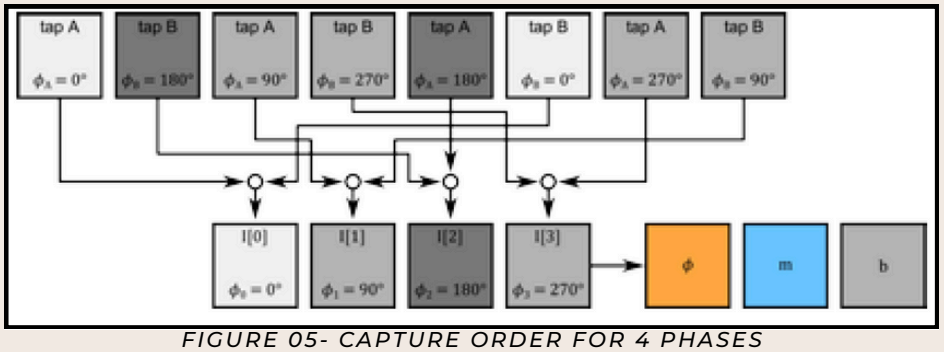


FIGURE 05- CAPTURE ORDER FOR 4 PHASES

The code I developed at this stage allows for the configuration of the modulation signal by specifying the frequency, signal shape, and the camera's exposure time. The exposure time corresponds to the duration during which the modulation light pulses and the camera acquires images. The longer the exposure time, the more light the camera receives during the capture of an image. I was ultimately able to configure the camera and retrieve the correct number of images according to the parameters entered by the user.

3- Data Processing

Once the images were acquired correctly according to the documentation, they needed to be processed to determine the luminescence lifetime. My goal was to display:

- A graph showing intensity
- A graph showing the phase shift between the modulation sinusoid and the emitted sinusoid
- A graph showing modulation

(for each pixel of the image)

I then started using the class "flim" from the pco.py library, which allows image analysis. There was no documentation for this part of the library, making the way of working of each function, the type of data to enter as function arguments, and how to interpret the returned data difficult to understand. There is a method called “calculate” that provides arrays corresponding to intensity, phase, magnitude, and phasor for each pixel.
The phasor is calculated using the discrete Fourier transform. We have a certain number of images after acquisition, and each was captured at a specific phase. The phasor is an image of the same size as the captured ones where each pixel (x, y) is defined as follows:

$$Phasor\,(x,y) = \sum_{k=0}^{N-1} I_k\,(x,y) \times e^{\frac{-2i\pi k}{N}}$$

Where N is the number of images captured and the complex exponential is the complex Fourier coefficient corresponding to the phase of image k. Figure 06 provides an example of the Fourier coefficients for 8 phases. We ultimately obtain a complex image that provides information about the phase and modulation of the signal. The phase is the argument of the phasor in the complex plane, while the modulation is the magnitude of the phasor.

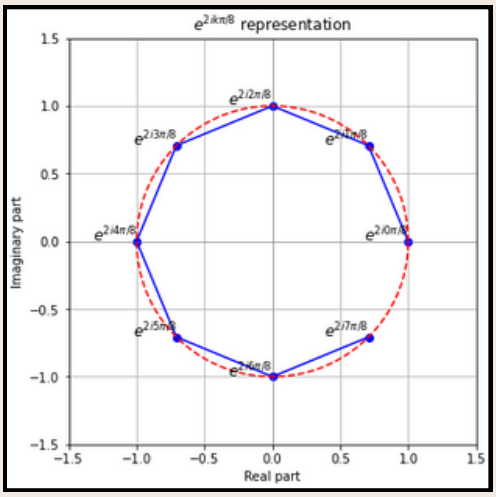


FIGURE 06- FOURIER COEFICIENTS FOR 8 PHASES

Displaying these three arrays with the addition of a legend allows us to visualize the graphs. I was thus able to create a program that retrieves the images and processes them to acquire the phase and intensity at every point of the image. Figure 7 shows the images I was able to display after this processing, including the display of maximum, minimum, and average values across the entire image.

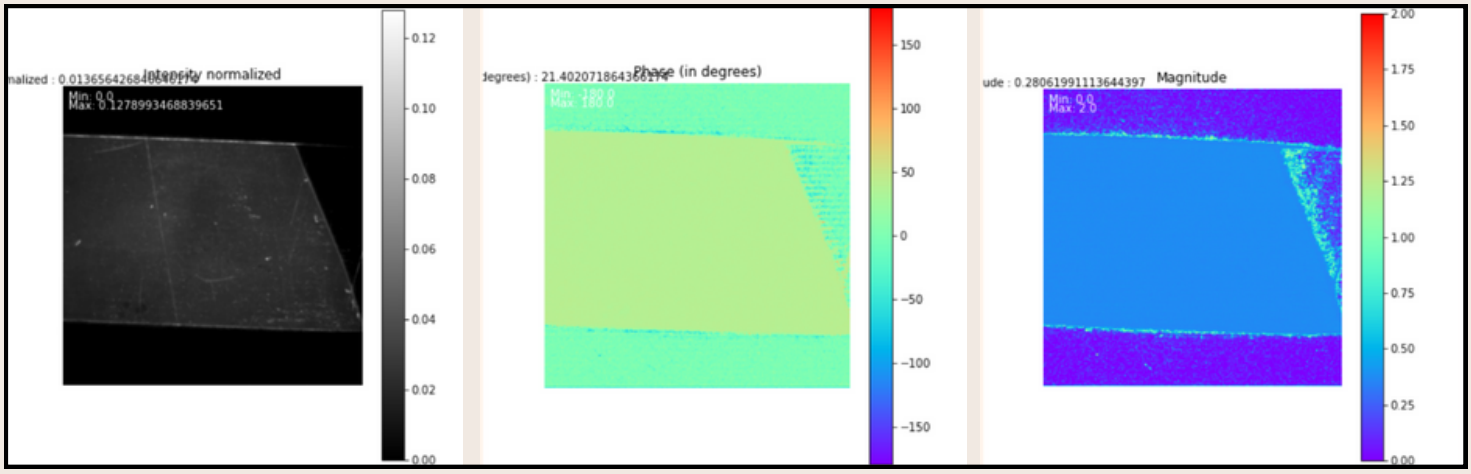


FIGURE 07- DISPLAY OF 3 GRAPHS

4- User Interface

Finally, I needed to create an easy-to-use interface for everyone at the university who uses this camera for their work. The interface had to allow the user to enter the same parameters that could be specified in the Nikon software.

I used the "Tkinter" library in Python, which offers many features for creating interactive graphical windows. I had freedom in designing the interface. My goal was to have a clean interface that would be easy to understand and use. Figure 8 shows a screenshot of the interface after starting an acquisition. It is divided into four main areas:
Zone 1 : A right sidebar to enter all the useful parameters for camera initialization and processing, as well as buttons to start the acquisition.
Zone 2 : A section at the bottom of the screen displays the minimum, maximum, and average values in a table based on the desired data. It also shows the luminescence lifetime and an option to export the data in Excel format.
Zone 3 : In the center of the screen, the three graphs are displayed with the option to save them and to select an area on each.
Zone 4 : There is also a menu at the top of the window that provides access to other features such as calibration tools or the software documentation.

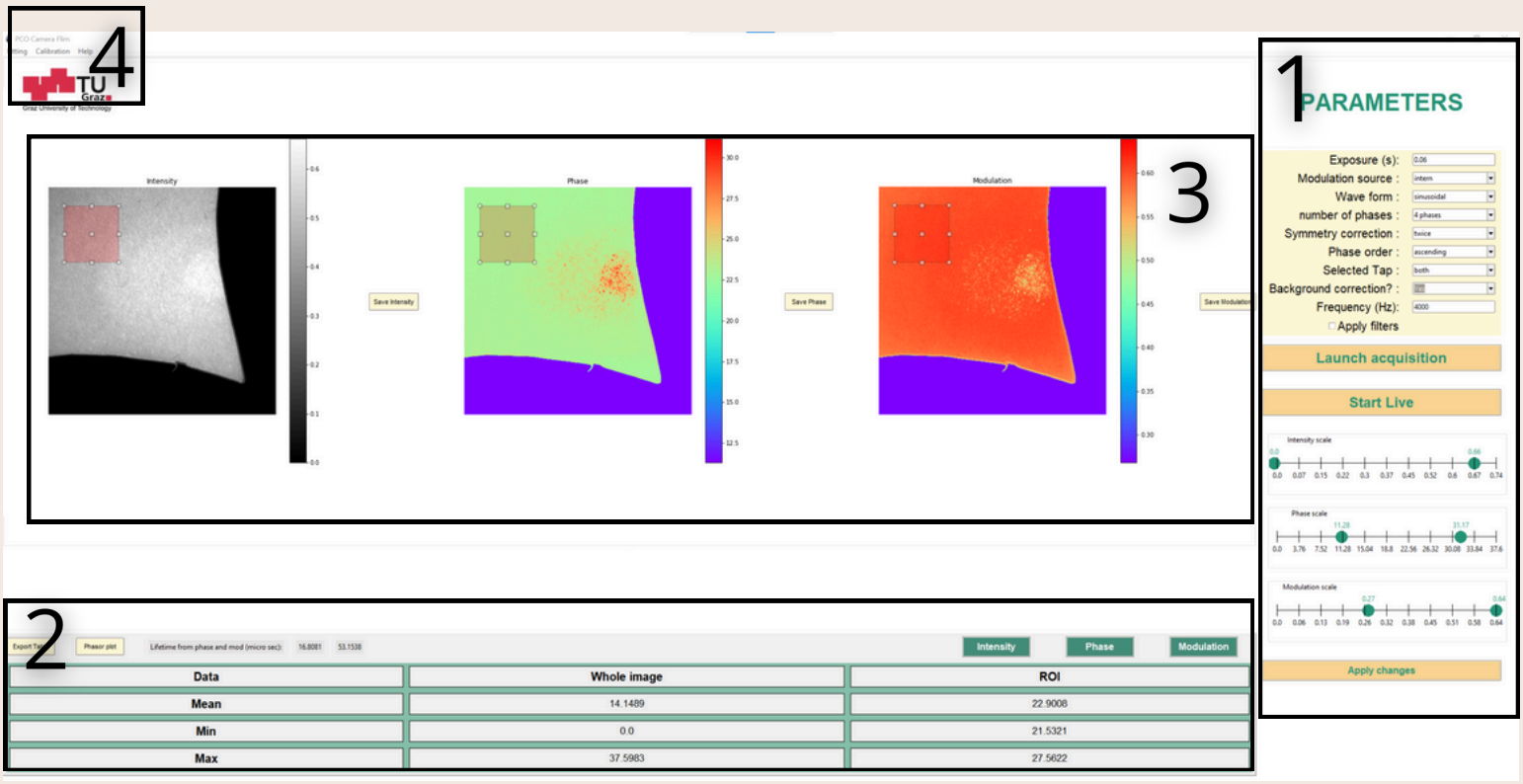


FIGURE 08- VISUAL OF THE INTERFACE

08 Secondary tasks

There were several features to add in addition to the main one, which is displaying the three graphs. Since I already had a functional interface, each new task was directly added to the interface, allowing it to be linked to the existing ones.

Live :

It is an option that lets you view the image captured by the live camera, so you can place the component correctly under the camera before starting the acquisition.

Background Correction :

Background correction aims to eliminate noise caused by the camera and the background and to standardize the background, giving it a coherent value. To perform calibration, several images are acquired, and an average is taken pixel by pixel. If the background correction option is checked, the value of each pixel in the image acquired during calibration is subtracted from each pixel in the captured images. If a pixel becomes negative, it is set to o. Thus, if there was no noise present, the background was probably black and the pixel value would be o. If there was noise or a non-uniform background, it will be removed from the image. This helps eliminate defective pixels. To standardize the background, I used a thresholding function from the OpenCV library to create a mask. I create a mask from the intensity image and then perform a logical "OR" operation between each of my images and my mask. This sets the background value to o in all graphs, including the luminescence lifetime, allowing us to focus on the data of the component.

Filtering :

Filters are applied to the image to remove noise.

I applied the Wiener filter using the "wiener" function from the "scipy" library, which uses the Fourier transform. By applying the Fourier transform, we can estimate the signal frequencies that correspond to noise using the signal's variance, for example. We can then attenuate these frequencies without modifying the others, thus improving the signal-on-noise ratio. I also used the Blur filter with the "medianblur" function from the "OpenCV" library. This filter replaces a pixel with the median value of the surrounding pixels, thereby removing defective pixels.

Calibration of Lifetime :

This option aims to adjust the calculations during image processing to obtain phase and modulation values closer to reality.

We place a component with a known luminescence lifetime under the camera and start the acquisition with the correct parameters. Then a window opens (Figure 09), prompting us to enter the known lifetime and select an area on the image where we can observe only the component, excluding the background values from the calculation.

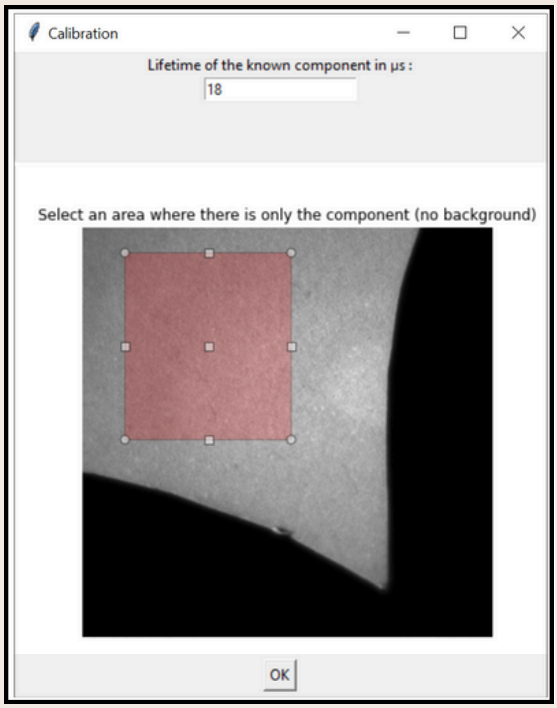


FIGURE 09- LIFETIME CALIBRATION WINDOW

We have formulas to obtain phase and modulation from the lifetime, so we get theoretical values for phase and modulation.

Thus, we can define coefficients for phi and for m such that:

$$coef\Phi = \frac{\Phi_{theorical}}{\Phi_{calculated}} \quad coefm = \frac{m_{theorical}}{m_{calculated}}$$

Thus, with each acquisition, the calculated phase and modulation data will be respectively multiplied by the phase and modulation coefficients.

Scale Adaptation :

Some components have low brightness, making it difficult to distinguish between the background and the component if they have similar colors on the graph. I created sliders that allow adjusting the scale of each graph. We determine the maximum and minimum values of the scale, and the display adapts accordingly.

For example, if the phase scale is initially between -180° and 180°, but the component has a phase around 25°, we set the sliders around 25°. This allows us to visualize variations in the component that were not observable before. Note that all pixels above the "max" slider will have the maximum color, and all those below the "min" slider will have the minimum color. Figure 10 illustrates this example:

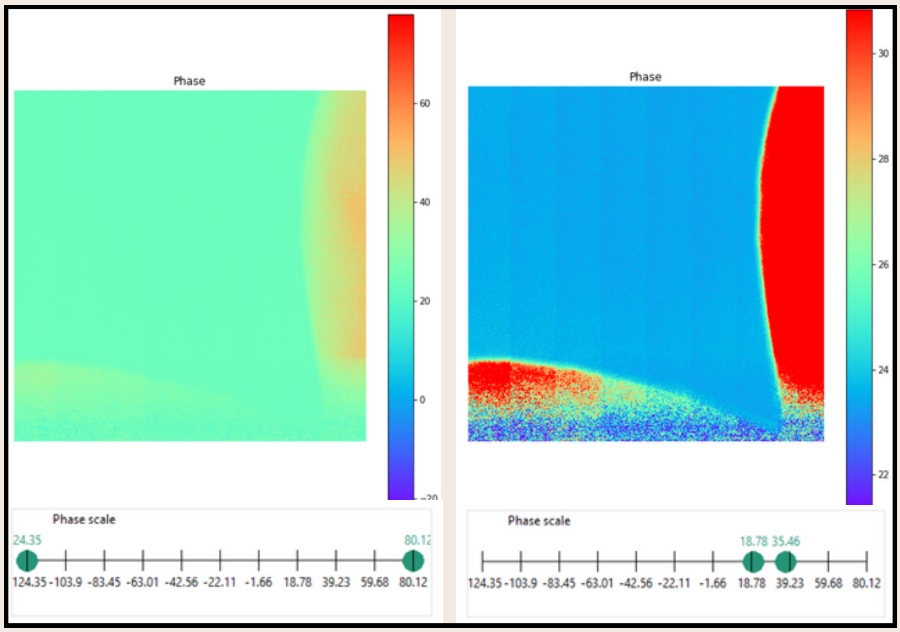


FIGURE 10- DIFFERENCE WITHOUT AND WITH THE SCALE ADAPTATION

ROI :

Once the acquisition is started and the three graphs are displayed, we obtain the minimum, maximum, and average values for the entire image. These values can be useful for detecting potential issues but do not allow us to establish the luminescence lifetime of a component. I used a tool from the Python library "matplotlib" that allows creating a region on a graph. With this option, we can select a rectangular area on any of the three graphs. This area will be automatically reflected on the others in the same location, and we obtain the desired values, calculated only from the selected area. We can then change the size and location of the area or delete it.

Save :

For each of the graphs, there is a button to export the graph and save it in PNG format. Similarly, we can save the data table in Excel format. It summarizes the minimum, maximum, and average values for each of the graphs, both for the entire image and for the selected region of interest.

I've also added a function to save the parameters entered by the user as well as the fitting equations and other calibration parameters when the window is closed. I used the JSON python library. When the window is closed, all parameters are saved in a ".json" file which will be read the next time the window is opened; if no file exists, parameters are set to default values.

Phasor Plot :

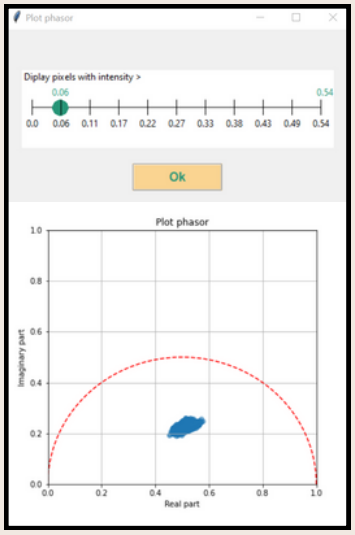


FIGURE 11- PHASOR PLOT WINDOW

The phasor plot is an important measure as it combines phase and modulation. It helps in detecting potential issues and visualizing if multiple luminescence lifetimes are present in a component. Each point on the plot represents a pixel of the image. If the points lie outside the unit circle, it indicates an error. If the points are all in the same area, it suggests that these pixels have the same luminescence lifetime.

Fitting :

The idea is to be able to display a parameter of interest such as pH, temperature or oxygen level according to its location on the image. The option is available in the top-left menu. A window appears (figure 12), allowing you to establish a curve equation based on a model. I used the Python lmfit library and created a file in which all the necessary models are defined (Boltzmann, Ster Volmer, Exponential...). The established equation is then saved, and in the "use fitting" menu, we can choose to perform an acquisition using a pre-established equation. For example, if we choose to acquire images to determine the pH using the Boltzmann model, an image of the intensity and one of the pH will be displayed, with a color scale and an option to select only a region of the image.

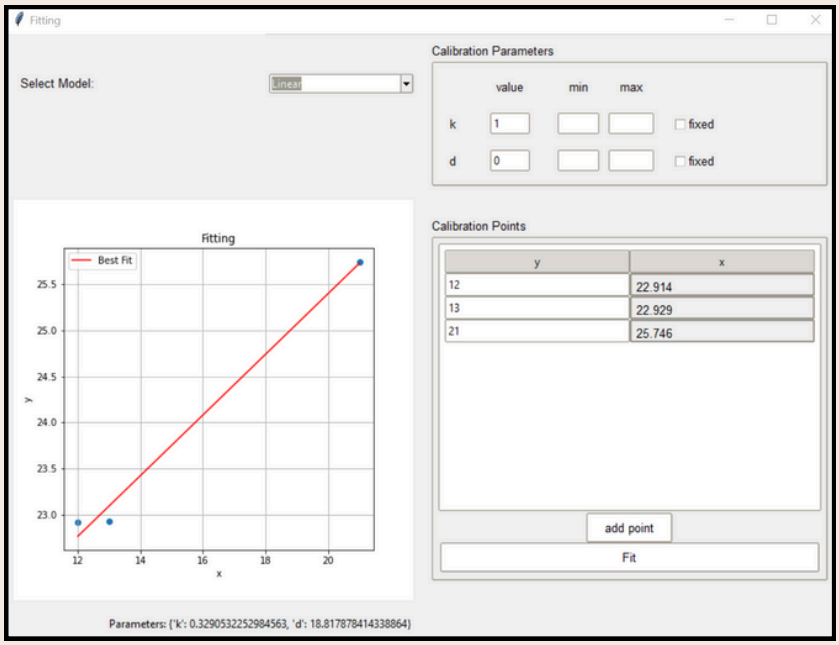


FIGURE 12- FITTING WINDOW

Documentation :

In the menu at the top right of the interface, in the "Help" section, we can view the user documentation. This document explains the purpose of each of the options, and gives hints on how to use them.

09 Conclusion

Technical assessement

During this internship, I encountered obstacles, but they did not prevent me from achieving my objectives. What caused me the most problems and took the most time during this project was getting to grips with the Python libraries. Indeed, I struggled to understand what could be managed via the libraries and what needed to be done directly in the code.

The tasks assigned to me were partially completed. My main task was to create software to use the PCO FLIM camera. This task was accomplished since I was able to provide software with basic functions. Some objectives were added during my internship, given my progress on the interface. These were not primary tasks. We are now able to use the PCO FLIM camera at the institute to acquire data and make measurements without using the Nikon Nis Elements software.

I consider my objective achieved.

In annex [blabla], you will find a screenshot of the software provided to the University of Graz at the end of this internship.

With more time, we could have completed this software and added features to improve the user experience and the practicality of the software. The only objectif that could not be achieved is the creation of an executable. It was'nt a necessary task so I preferred to concentrate on other more important tasks.

Personal Assessment

This internship provided me with the opportunity to develop and deepen certain skills:

- Adaptation
- Organisation
- Autonomy
- Communication

I had to perfect my organization since I was working autonomously. I had to adapt to each situation that arose and respond to the problems I encountered.

This internship was also very enriching from a personal point of view. Indeed, I completed this internship abroad, which allowed me to meet new people, discover a new culture, and experience a different way of working. Moreover, I had to speak English throughout this internship, which helped me improve my language skills and made it an unforgettable experience.

10 Appendix 1 -Screenshot of the final interface