UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND BIOMEDICAL INFORMATICS

Monitoring of nasal mucosa blood supply alterations by means of a computer vision system

EVANGELOS PSYLLAKIS

THESIS Supervisors

Dimitrios Iakovidis

Professor

DCSBI - UTH

Vassilis Papadakis

Researcher

IMBB - FORTH

Lamia, 2022

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND BIOMEDICAL INFORMATICS

Monitoring of nasal mucosa blood supply alterations by means of a computer vision system

EVANGELOS PSYLLAKIS

THESIS Supervisors

Dimitrios Iakovidis

Professor

Department of Computer Science and Biomedical Informatics- UTH

Vassilis Papadakis

Researcher

Institute of Molecular Biology and Biotechnology of the Foundation for Research and Technology Hellas

Lamia, 2022

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις (1), που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Καταγραφή των μεταβολών της αιμάτωσης της κάτω ρινικής κόγχης, με την χρήση τεχνικών υπολογιστικής όρασης

ΨΥΛΛΑΚΗΣ ΕΥΑΓΓΕΛΟΣ

Acknowledgements

First supervisor Dimitrios Iakovidis, Professor.
Second supervisor Vassilis Papadakis, Researcher.
I am particularly grateful for the assistance given by Prof. Emmanuel Prokopakis.
I would like to express my very great appreciation to Dr Dimitrios Iakovidis, Profesor, and Dr Vassilis Papadakis, Researcher, supervisors of this thesis, for their valuable and constructive suggestions during the planning and development of this research work. Their willingness to give their time so generously has been very much appreciated.
I would also like to thank the following organizations:
1: Department of Otorhinolaryngology-Head and Neck Surgery, University of Crete School of Medicine, Heraklion, Crete, Greece.
2: Institute of Molecular Biology & Biotechnology, Foundation for Research and Technology – Hellas, N. Plastira 100, GR-70013, Heraklion, Crete, Greece.
3: Department of Computer Science and Biomedical Informatics, Papasiopoulou 2-4, 35131, Galaneika – Lamia, Greece.

Abstract

Blood supply alterations in the nasal mucosa are associated with multiple clinicopathologic conditions. However, there is no objective method on evaluating the effectiveness of pharmaceutical agents used in the aforementioned multiple clinic-pathological conditions. Moreover, there is a necessity of improvement in the effectiveness of current clinical diagnosis towards an objective characterization. Therefore, the need of an improvement in diagnostic potential and the evaluation of the pharmaceutical agents' effectiveness is deemed necessary. In this study, the topic of non-destructive detection and staging of nasal mucosa blood supply alterations is addressed with a unique approach. The method is based on computer vision approaches and in particular multispectral imaging and advanced signal processing. The pilot clinical trials have shown that the proposed method can characterize the mucosa tissue changes under the application of a pharmaceutical agent in a quantitative way. Both clinicians and researchers may find this approach to be extremely useful in the evaluation of the different pharmaceutical agents' efficiency.

Contents

1 Introduction

Blood supply alterations in the nasal mucosa are associated with multiple clinicopathologic conditions, such as allergic rhinitis, vasomotor rhinitis, and inflammatory disorders of the paranasal sinuses [1]. Moreover, they are related with the topical or systematic application of different pharmaceutical agents [2][3]. In comparison to other mucosa membranes, the nasal mucosa is easily accessible and serves as a convenient entry point for small and big molecules [3].

The diagnosis related to nasal pathology is done by a non-objective manner today is far from objective. Physicians use qualitative scales to assess related pathology [2]. In this study, a novel system and methodology based on computer vision techniques to improve the effectiveness of current clinical diagnosis was introduced. The project had two main focus areas. The first one was to collect a number of pilot measurements in normal subjects. Measurements were performed with the use of a multispectral imaging camera coupled to an endoscope to better characterize blood spectral features. The second goal was to analyze data and correlate them with the tissue characteristic changes under the effect of a pharmaceutical agents.

Data analysis required the application of advanced signal and image processing algorithms. In particular, pattern recognition techniques were used to compare images acquired in different time points from a moving object. The oxygenation of hemoglobin was calculated by dividing the two wavelength bands, 642 nm and 870 nm. The lower the amount of blood in the nasal mucosa, the lower the oxygenation of hemoglobin is expected to be, under this assumption, that the size of the nasal mucosa is related with the amount of blood in the nasal mucosa. The outcome of the work performed in this study is a 2-Dimensional Graph that is called the Hemoglobin Oxygenation Dynamic Graph. This 2D graph is going to depict the changes that happen in the nasal mucosa after the effect of a pharmaceutical agent.

1.1 Aim and Contributions

This thesis had 2 aims, the first one being the collection of pilot measurements in normal subjects using a snapshot camera and the second one being the creation of a method that can characterize the mucosa tissue changes under the application of a pharmaceutical agent in a quantitative way. The pilot clinical trials have shown that the proposed method is capable of characterizing the mucosa tissue changes under the application of a pharmaceutical agent in a quantitative way.

The contributions of this study are as follows:

- The use of multispectral and computer vision techniques and algorithms for the data processing and analysis lead to the representation of the changes that happen in the nasal mucosa after the effect of a pharmaceutical agent in 2 dimensional graphs. These Hemoglobin Oxygenation Dynamic graphs provide the information needed to conclude in a legitimate outcome.

- This thesis with the preliminary data it provided presented a 2D Hemoglobin Oxygenation Dynamic model graph to depict the changes that happen in the nasal mucosa after the effect of a pharmaceutical agent.
- The Graphical user interface, that was created for this thesis, made the whole process of data processing and analysis easier and user-friendly, and it is provided in the appendix section.

1.2 Thesis Structure

The structure of this work is classified into nine sections: Section 1, includes the Introduction; Section 2 and 3, provides background information about the image processing and multispectral imaging and the anatomy and physiology of the nasal cavity. Section 4 represents the related work that has been developed around this subject. Section 5 illustrates the materials that were used and the methodology of our algorithm, Section 6, analyses the results we found from our research. In the Section 7 takes place a discussion and a few suggestions that could lead this research to the next level and in Section 8 is presented the conclusion of our work. In the Section 9 concludes the Bibliography, Finally, in the last Section there is the appendix which contains the Python code, that was used to perform image processing and the analysis, as well as the user's manual, in which are contained the necessary information for the use of the Graphical User Interface (GUI) that was created in this research.

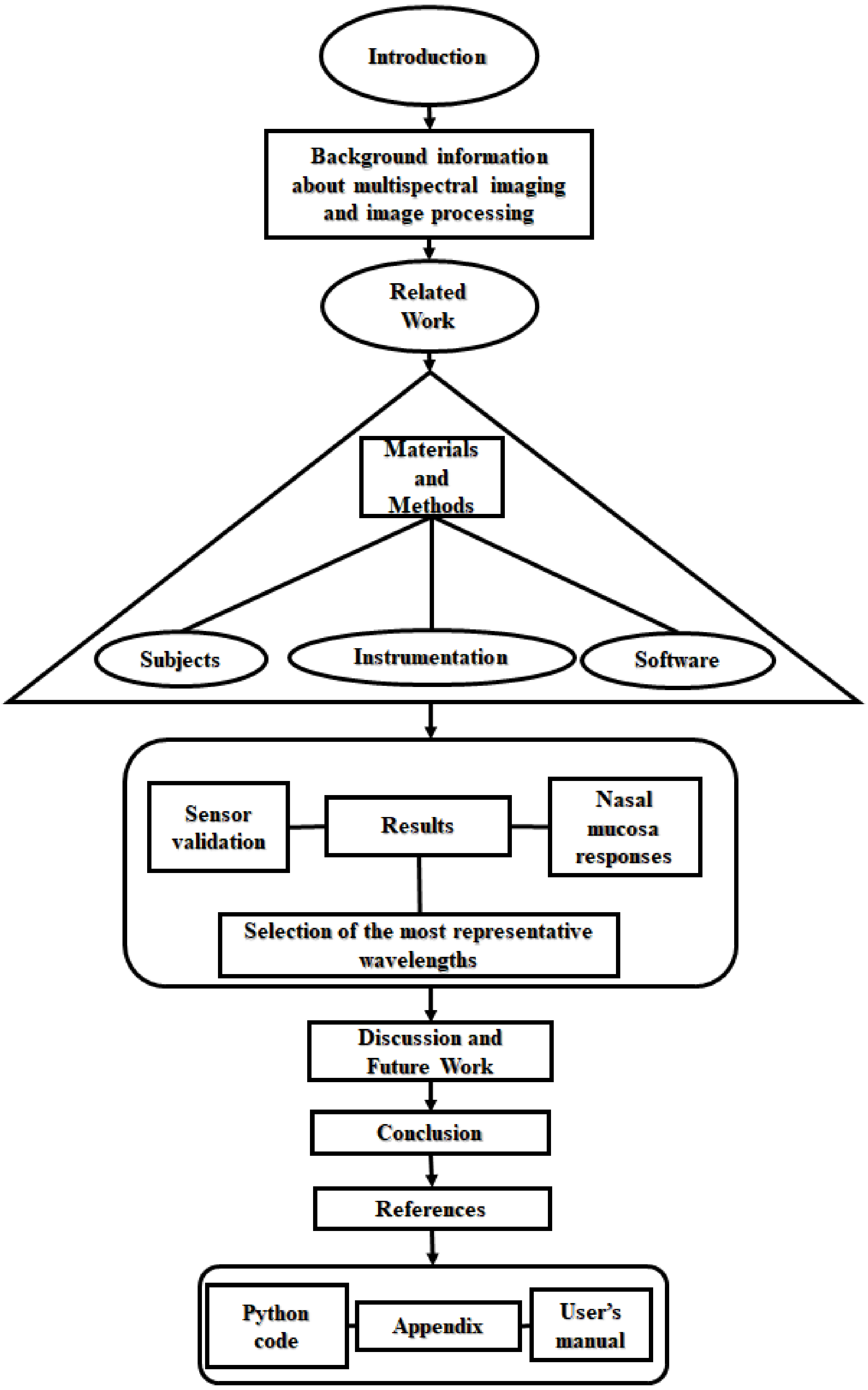


Figure 1. Thesis Structure

2 Multispectral Imaging and Image Processing

A brief summary of some of the essential components that characterize the domains of Multispectral Imaging and Image Processing is presented in this section, with the goal of providing enough clarity on the background that sets the framework for this thesis.

2.1 Digital images

We must first establish how images are recorded in digital form before we can study some of the approaches included in the field of image analysis. If we sample a continuous image of the type A(x, y) with x, y continuous variables, we may acquire the equivalent digital image by performing sampling and quantization procedures to convert it to a discrete and finite form that can be processed digitally. As a result, images are represented as two-dimensional arrays A(i, j), where (i, j) are discrete spatial coordinates and A() records the intensity value at any (i, j) position as specified by its bit depth.

The image's height and width are determined by M rows and N columns, respectively. Each image piece f(x, y) is also known as a pixel, as it contains an intensity value and, when combined with its neighbors, provides the appearance of a continuous tone image.

The numerical representation of a digital image is expressed as follows:

A(i, j)=

A(0,0)	A(0,1)	A(0,2)	A(0,N-1)
A(1,0)	A(1,1)	…	A(0, N-1)
…	…	…	A(0, N-1)
A(M-1,0)	A(M-1,1)	…	A(M-1, N-1)

(i)

2.2 Multispectral Imaging

Multispectral imaging (MSI), also known as imaging spectroscopy, is a technique that uses optical elements to disperse incoming radiation into discrete wavelengths, allowing it to sample hundreds of small spectral bands over the electromagnetic spectrum. This method combines the best qualities of two current technologies: imaging and spectroscopy, allowing researchers to use both morphological and chemical properties of objects acquired by a camera. Because the interaction between electromagnetic radiation and matter differs depending on the material, this technique can be used to distinguish between them. [4]

Although MSI has traditionally been used in remote sensing, it has recently been a hot issue in a variety of research domains, including food quality assessments, military and security applications, and precision agriculture. In the healthcare field, MSI is also a growing imaging technique. The study of light propagation through biological tissues can help diagnose a variety of illnesses. These aspects of light's interaction with biological tissue encourage the development of instruments for diagnostic assistance based on technologies that utilize the information of light propagation through tissues. One of MSI's advantages as an alternative diagnostic technique is that it is fully non-invasive and label-free. Multispectral/hyperspectral (MS/HS) imaging (MSI/HSI) has grown in popularity in recent decades, with applications in a wide range of fields. This sort of imaging gathers data across the electromagnetic (EM) spectrum, which includes a wide variety of wavelengths spanning the visible, near-infrared, and midinfrared regions.[4]

MS/HS images are a three-dimensional (3D) data structure with two dimensions for spatial information and a third for spectral information (figure 2). This technique combines the best aspects of two current technologies: imaging and spectroscopy, allowing users to leverage both the morphological and chemical characteristics of objects acquired by a camera. Because all materials reflect, absorb, or emit EM radiation at specified wavelengths, this feature of HS images allows for the reconstruction of the radiance spectrum of each image pixel and, as a result, the identification of distinct materials based on their spectral form. As a result of this quality, HS data may be used in a wide range of applications.[4]

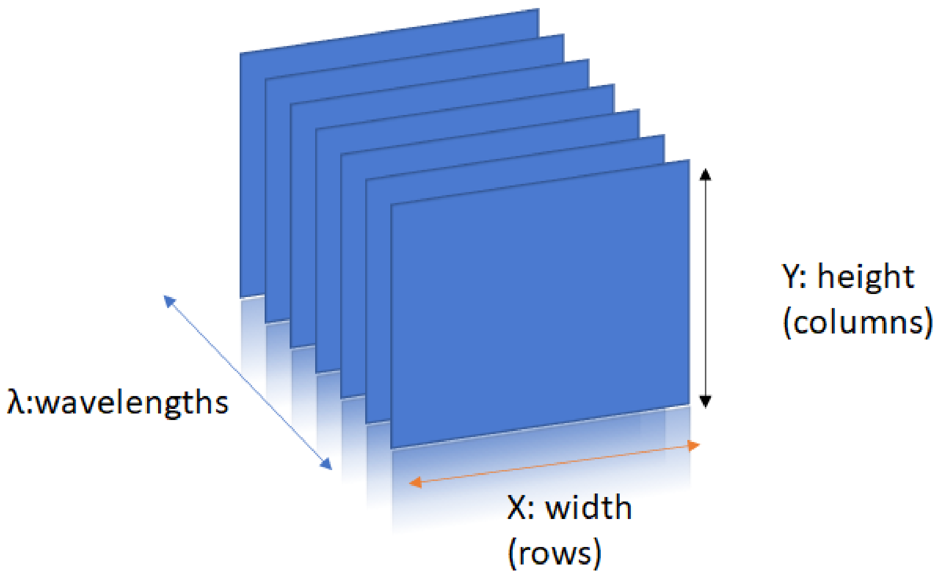


Figure 2. 3-Dimensional Multispectral (MS) cube

2.3 Imaging Spectroscopy

Imaging Spectroscopy, also known as HSI and MSI, is a method that may overcome the imaging limits of human eyesight based on white light (WL). In reality, HSI combines the benefits of two technologies that have been utilized separately for decades, namely digital imaging and spectroscopy. Digital imaging allows for the capture of morphological elements of a scene, as well as the extraction of information about the shape and texture of various objects. Spectroscopy studies the interaction of electromagnetic radiation with matter.[4]

While human eyesight is limited to a certain area of the EM spectrum (EMS), the visible spectrum, spanning 400 to 700 nm, most typical MS commercial systems extend this spectral range to 400 to 2500 nm. Although MS cameras exist that can cover the EM down to 12 microns, they are limited to certain applications that are outside the scope of this research.[4]

MSI provides knowledge in parts of the EMS that the human eye cannot see, exposing qualities of substances that are otherwise unavailable to humans. Furthermore, the human eye can only discriminate between three distinct wavelengths linked with the retina's opsins. An MS image is saved in a data structure known as an MS cube, which stores both spatial and spectral information about the image. The data contained in the MS cube may be viewed in a variety of ways. A single pixel from an MS image can be picked to investigate the spectrum associated with that pixel.[4]

Similarly, the whole spatial information for a particular wavelength may be seen. The information perceived at different wavelengths represents distinct characteristics of materials.

Many researchers have used MSI technology for a variety of purposes, including non-invasive food quality inspection, enhancing recycling operations, and studying artworks for exact pigment identification in order to enhance their restoration. MSI is a technique used by geologists to locate various minerals. Furthermore, this method has been utilized in agriculture to quantitatively assess soil and determine plant stress levels.[4]

2.3.1 Spatial scanning systems

In most spatial scanning systems, an optical element performs diffraction of incoming light into fixed places of a sensor, where the spectral channels are sampled. There are two types of multispectral spatial scanning systems, Whiskbroom systems and Pushbroom systems. Whiskbroom systems can collect an one-dimensional data array including the spectrum of the object on which the device is focusing. To capture a multispectral cube in this fashion, spatial scanning is required, in which the camera or the captured object(s) shift position while the camera captures frames. Scanning can also be done with a mirror in front of the fore optic, which is moved to image the entire object. Although the use of mirrors enables for the development of more compact instruments, the geometric distortion that mirrors might cause in the collected image must be considered. [4]

Pushbroom HS systems, unlike Whiskbroom multispectral systems, utilize optical gratings that can capture the spatial response of objects over the entire spatial dimension. In other words, pushbroom acquisition devices can capture a single spatial dimension (a narrow line in an image) as well as the entire spectrum information for a specific scene at the same time. To obtain a multispectral cube in pushbroom systems, a spatial scan of the remaining spatial dimension is necessary. The optical element that separates the incoming radiation into various wavelengths values is at the heart of these cameras. These optical devices have a high spectral resolution and are sensitive to a larger range of wavelengths (from 300 nm to 2500 nm).[4]

The spatial resolution of spatial scanning systems is limited, both by the sensors and by the spatial scanning process itself. Furthermore, the requirement of spatial scanning prevents spatial scanning systems from acquiring moving objects. These systems, on the other hand, are well suited to linear movement environments, such as a production line chain or satellites for remote sensing applications. As a result, these cameras are employed in open surgical procedures, in-vivo surface inspection, and ex-vivo tissue investigation in the medical industry.[4]

Due to its incapacity to do spatial scanning, this type of camera cannot be directly attached to medical devices such as laparoscopes or intraoperative microscopes. In terms of acquisition time, it is limited by a tradeoff between the scanning system's speed and the camera frame rate. However, when compared to other multispectral systems, spatial scanning requires longer acquisition periods.[4]

Both methods are based on the vibrational state of the molecules, however RS is more sensitive to scattering changes than FTIR spectroscopy. As a result, both technologies are mutually beneficial. Furthermore, Spatial Frequency Domain Imaging (SFDI) uses modulated light sources and light transport models to extract information on tissue absorption and scattering, which may then be used for diagnostics.[4]

Finally, fluorescence spectroscopy techniques may detect the fluorescence spectra of the specimen following light excitation after the application of certain fluorescent agents to the sample. These fluorescence spectra can be linked to various illness states, leading to biomedical diagnostic applications.[4]

2.3.2 Spectral scanning systems

Another way to make a multispectral cube is to use spectral scanning, which captures both spectral dimensions for a single wavelength each time. This type of camera uses an optical element to filter the incoming radiation, allowing it to record the whole spatial information of a single wavelength at any one time. In the literature, spectral scanning systems are also known as staring arrays. Staring array cameras come in a variety of shapes and sizes.[4]

The most basic one consists of a wheel comprising optical filters, with mechanical switches to switch between them. Despite this, the number of spectral channels available on filter wheels is restricted, and the time necessary to capture a multispectral cube is lengthy due to the mechanical switching of spectral channels. Other technologies, on the other hand, allow for the collection of images with higher spectral resolution.[4]

The Liquid Crystal Tunable Filter (LCTF) and the Acousto-Optic Tunable Filter (AOTF) are spectrum transmission devices that may be electronically modified. The spectral bandwidth of the filter limits the spectral resolution of these systems. Although narrow spectral filters are available on the market, their spectral resolution is inferior to that obtained through light diffraction.[4]

Another disadvantage of AOTFs and LCTFs is that the spectral range they can cover is limited. Despite this constraint, spectral scanning technologies allow standard monochromatic cameras to be used as sensors. As a result, depending on the sensor chosen, the spatial resolution can be quite high. It allows multispectral systems to achieve the maximum spatial resolution feasible.

The number of spectral channels to be collected and the switching time between different wavelengths determine the acquisition time in staring array devices. The switching time in AOTFs and LCTFs is often measured in tens of milliseconds. Another benefit of LCTFs and AOTFs is that they allow you to reduce the time it takes to acquire a multispectral cube by just picking the most important spectral channels for a certain application.[4] Because the spatial information for different wavelengths may fluctuate, spectral scanning techniques are not ideal for situations where the captured object is moving. These cameras, on the other hand, may readily be mounted to medical tools and provide great spatial resolutions.[4]

2.3.3 Snapshot systems

The last form of Multispectral camera is a snapshot camera. The fundamental limitation imposed by the previously discussed Multispectral technologies is real-time acquisition, which Snapshot technology aims to address. For the purpose of executing a scan, it is not possible to capture multispectral data in real time when utilizing the afore-mentioned multispectral technologies (either spatial or spectral). These technologies are limited to static conditions or scenarios in which the moving item moves at a slower rate than the scan speed. For these reasons, a snapshot camera must be used whenever multispectral data of non-static scenes is required. Snapshot cameras can also be directly connected to clinical instruments like endoscopes and laparoscopes. Despite this, the snapshot cameras' spectral and spatial resolutions are lower than those of other multispectral technologies. [4]

2.4 MSI for medical applications

Similarly, the whole spatial information for a particular wavelength may be seen. The information perceived at different wavelengths represents distinct characteristics of materials. Many researchers have used MSI technology for a variety of purposes, including non-invasive food quality inspection, enhancing recycling operations, and studying artworks for exact pigment identification in order to enhance their restoration.[4]

MSI is a technique used by geologists to locate various minerals. Furthermore, this method has been utilized in agriculture to quantitatively assess soil and determine plant stress levels.[4]

These absorption peaks serve as a fingerprint of the molecules' light reaction, providing information that may be exploited in diagnostics. When there is a spatial difference in the reflective index inside tissues, light scattering occurs. Under specific illness circumstances, the dispersion of particular biological components exhibits fluctuations, making it valuable for diagnosis. Finally, when stimulated by an external light source, some tissues glow.[4]

After exposing tissue to UV light, for example, the emission of proteins and nucleic acids can be noticed. Traditionally, these tissue qualities have been evaluated in the therapeutic window, a spectral region between 600 and 1300 nm in which tissues have poor absorption and light is more likely to permeate tissues.[4]

These aspects of light's interaction with biological tissue encourage the development of instruments for diagnostic assistance based on technologies that utilize the information of light propagation through tissues. For diagnostic purposes, Raman Spectroscopy (RS) and Fourier Transform Infrared (FTIR) Spectroscopy employ the information of vibrating molecules created by photons. The ability to detect different types of molecules using RS and FTIR spectroscopy has led to its use in biomedical applications.[4]

3 The anatomy and physiology of the nasal canal

Superior, middle, and inferior turbinates are three pairs of structures that are found along the lateral walls of the nasal canal (figure 3). Conchae are the skeletal components of the turbinates; the conchae of the intermediate, superior, and supreme turbinates are extensions of the ethmoid bones, but the conchae of the inferior turbinate (figure 4)- the biggest turbinate - is a distinct bone. A meatus exists underneath the connection of each turbinate to the lateral nasal wall, into which numerous distinct outflow channels from the orbits and paranasal sinuses drain. Turbinates perform a crucial physiological role in the body by warming and humidifying inspired air and controlling nasal airflow, but they also play a significant role in nasal airway blockage, especially in allergies and viral upper respiratory infections. [5]

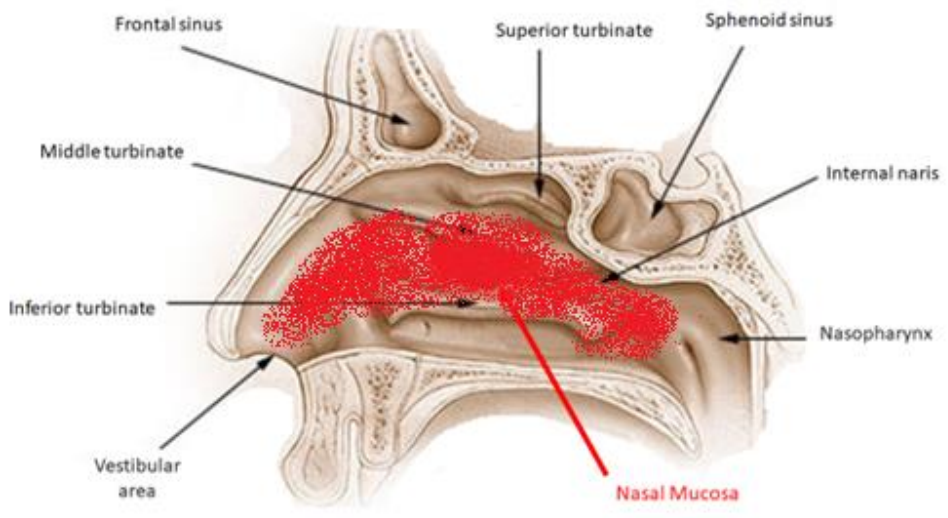


Figure 3. Nasal Anatomy

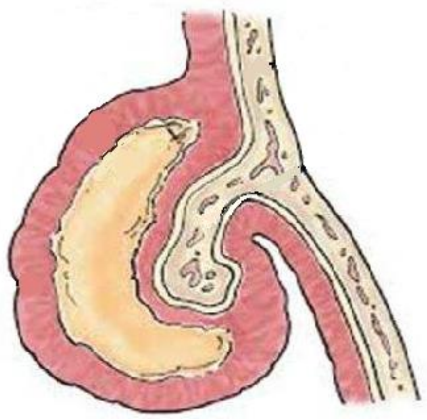


Figure 4. Conchae of the inferior turbinate.

3.1 The function of the nasal turbinates

The nasal turbinates warm and humidify air as it passes through the nasal cavity and into the nasopharynx on its journey to the larynx, trachea, and lungs; their shape as curving ledges extending from the lateral nasal walls gives them more surface area to do so. The turbinates, particularly the inferior turbinates, also control airflow to maintain proper moisture levels in the nasal cavity by swelling and contracting, blocking or allowing air to pass. A layer of erectile tissue contains venous sinusoids that serve to control the volume of the turbinates based on autonomic stimulation underneath the pseudostratified columnar respiratory epithelium that covers the turbinates. In fact, one inferior turbinate swells while the other contracts to divert the majority of airflow through one nostril at a time; the turbinates then alternate which is swollen and which is contracted in a process known as the "nasal cycle," which can last anywhere from half an hour to six hours, with longer cycles occurring during sleep. [6], [7]

The inferior turbinates, unlike the other turbinates, are separate structures that articulate with the ethmoid, maxillary, lacrimal, and palatine bones; the inferior turbinates, unlike the other turbinates, are not processes of the ethmoid bones but are separate structures that articulate with the ethmoid, maxillary, lacrimal, and palatine bones. The outflow tract of the nasolacrimal duct is located in the inferior meatus, which is located beneath the shelf of the inferior turbinate.[5]

The inferior turbinate is a paired, elongated, scroll-like structure that articulates with the nasal surface of the maxilla and the perpendicular plate of the palatine bone. It is made up of a central osseous skeleton and a mucosal layer on either side of the bone. Through temperature control and humidification of inspired air, as well as filtration of foreign particles via the mucociliary clearance system, the inferior turbinate plays a significant role in lung defense and nose physiology. Allergic rhinitis, nonallergic (vasomotor) rhinitis, and rhinitis medicamentosa are only a few of the conditions that can produce major hypertrophy changes in the inferior turbinate, resulting in increased nasal resistance and blockage. [7]

Hasner's valve, also known as the plica lacrimalis, is a mucosal fold that prevents air from refluxing through the lacrimal system. The hiatus semilunaris is located in the middle meatus, just inferior to the ethmoid bulla and posterior to the ethmoid bone's uncinate process. The frontal sinus, maxillary sinus, and anterior ethmoid sinuses all have outflow channels inside the semilunar hiatus. The sphenoethmoidal recess in the superior meatus is where the posterior ethmoid sinuses and the sphenoid sinus drain.[5][8]

Moreover, because they are the first intranasal structures to come into contact with outside air, the inferior turbinates serve a role in immunological surveillance in addition to their respiratory duties. The pathogenicity of inhaled particles is determined by the inferior turbinates, which can activate innate and adaptive immune responses. The middle and superior turbinates do not appear to be involved in the immunological response, but they may be involved in olfaction.[5], [9], [10], [11]

3.2 Lymphatics and Blood Supply of the nasal cavity

Branches of the internal and external carotid arteries deliver blood to the nasal cavity. The sphenopalatine artery, a branch of the internal maxillary artery, which is a terminal branch of the external carotid artery, is the major source. A cadaveric study from 2016 found that the superior turbinate is primarily perfused by the posterior septal artery, a branch of the sphenopalatine artery, while the inferior and middle turbinates appear to be perfused by branches of the posterior lateral nasal artery, which is also a branch of the sphenopalatine artery and enters these turbinates posteriorly.[12], [5]

The anterior lateral nasal artery, a branch of the facial artery from the external carotid, supplies the middle turbinate, while an anastomosis between the anterior lateral nasal artery and the anterior ethmoidal artery, which comes from the internal carotid's ophthalmic artery, supplies the inferior turbinate. The blood supply to the inferior turbinate is particularly important from a surgical standpoint because the posterior aspect of the turbinate frequently causes persistent epistaxis after turbinoplasty; however, the anterior blood supply is sufficient to support the pedicled transfer of the inferior turbinate for nasal cavity reconstructive procedures like septal perforation repair.[5]

3.3 Variants in Physiology of each patient

The anatomy of the ethmoid bone varies greatly, both in terms of the number of nasal turbinates and in terms of pneumatization. Most people are born with three pairs of turbinates, two of which arise from the ethmoid bone - the middle and superior - and one of which is a separate bone. The supreme turbinate, a tiny bony structure above the superior turbinate, is not usually clinically noteworthy, hence it isn't well characterized in the literature. It can appear bilaterally, unilaterally, or not at all, and some ethnic groups are more likely to have it. [13] The shape of the other turbinates is also diverse, with pneumatization of one or both middle turbinates occurring in 30 to 70% of cases, while pneumatization of the inferior or superior turbinates is uncommon. Turbinate bifidity has also been described, but it is unusual. [5], [14], [15]

3.4 Clinical Importance of the nasal cavity

Most patients are unaware that the nasal turbinates play a crucial role in breathing, immune surveillance, and olfaction when they are functioning normally. However, turbinate dysfunction, particularly in the inferior turbinates, can have a considerable impact on quality of life. When the turbinates become hypertrophied, they cause anatomical nasal obstruction; however, if they are removed or their mucosa is badly wounded, the sense of nasal obstruction can exist even when there is no physical obstruction. Finally, olfactory epithelial malfunction, which causes a loss of smell and a reduction in taste, can have a major influence on patient well-being. [5], [16], [17], [18]

To help manage turbinate dysfunction, a variety of medical and surgical therapies are available, which are often administered by primary care providers, allergists, and otolaryngologists; using an interprofessional healthcare team strategy will ensure that patients receive appropriate evaluation and individualized treatment.[5]

3.5 Zones of Epithelium and Basement Membrane of the inferior turbinates

The inferior turbinates, like other respiratory areas, are covered by a pseudostratified ciliated columnar epithelium that contains around 10% goblet cells in addition to deeply placed basal cells and superficially ciliated and nonciliated cells. On the lateral side, they were more frequent than on the medial and inferior sides. Exceptions to this norm include squamous epithelium near the anterior end of the inferior turbinates and tiny irregular islands of transitional epithelium dispersed throughout the organ. The epithelium with a significant barrier function is 40 to 80 μm thick on all sides, with a mean value of 54 μm. The epithelium is separated from the lamina propria by a thin acellular basement membrane zone, which is substantially different in thickness along the medial, lateral, and inferior portions of the mucosa (18.5, 14.5, and 15 μm, respectively [$P = 0.032$]). The difference between the medial and lateral aspects was significant, while the differences between the medial and inferior and the lateral and inferior aspects were minor, according to post hoc analysis.[7] The nasal mucosa has long been recognized to alter in response to environmental stimuli8 from a pseudostratified cylindrical and ciliated with goblet cells epithelium to one that is either stratified cuboidal or stratified squamous or in transition.[7][19]

3.6 Measures and Characteristics of the Human Nasal Cycle

At all levels of biological activity, from microscopic components to the entire organism, cyclic processes are essential. The nasal cycle is a large-scale cycle that occurs in mammals, in which nasal airflow is stronger in one nostril than the other, and the greater airflow nostril swings from left to right over time. The nasal cycle was seen in all mammalian species that were studied.

Because to brief asymmetric nasal tube occlusion by erectile tissue, nasal airflow is larger in one nostril than the other. The nasal cycle is the periodicity with which the level of blockage alternates between nostrils. The nasal cycle is linked to autonomic arousal and is suggestive of brain asymmetry. Changes in nasal cycle periodicity have also been connected to a variety of illnesses. [6], [20], [21], [22]

The nasal cycle is caused by asymmetry in blood flow, which causes erectile tissue to engorge in the anterior section of the nasal septum and inferior turbinate of one nostril over the other. [23] This asymmetrically swollen tissue physically prevents air from passing through one nostril more than the other. The physiological processes that drive this physical mechanism have not been found. Unilateral sympathetic dominance is linked with vasoconstriction and decongestion in one nostril, whereas simultaneous parasympathetic dominance is associated with vasodilation and congestion in the other. These processes are obviously tied to the autonomic nervous system.[23], [24], [25], [6]

Given that a cycle is detected after thoracotomy, vagotomy, and vidian neurectomy, as well as in individuals with autonomic nerve disorder, autonomic nervous system asymmetry may not be the main cause of the nasal cycle. [26] Furthermore, with asymmetric autonomic system modulation alone, there was no association between the nasal cycle and face temperature or middle ear pressure. [27], [28] The nasal cycle's functional relevance is debatable. Some studies explain nasal airflow changes as a mechanism for air cooling and the elimination of entrapped pollutants or mucociliary clearing. Others believe that the nasal cycle aids in the prevention of respiratory infections and allergies, as well as indicating these physiological conditions. These ideas treat the nose as if it were a respiratory organ, ignoring olfaction and the nasal cycle–CNS connection.[29][6]

Previous human investigations have discovered that the nasal cycle periodicity, or the change in higher airflow or lower resistance from one nostril to the other, spans from 25 minutes to eight hours during wakefulness, with a peak interval of 1.5–four hours. [30] These results were calculated using technology-advanced nasal cycle measuring methods. Exhaling via the nose onto a mirror or using sophisticated measures of acute nasal flow such as rhinoresistometry, rhinomanometry, etc. can provide temporally discrete data.[6], [31], [32]

4 Related Work

There are only a few methods that try to find an accurate and objective way of evaluating the pharmaceutical agents as well as an effective way for the clinical diagnosis in other tissues.

At predefined time intervals, there have been performed endoscopic examinations of the anterior third of the inferior turbinate before and after the administration of Oxymethazoline. In the anterior 1/3 of the inferior turbinate (figure 3) were able to detect time-related variations in the oxygenation properties of hemoglobin. Relevant curves were recorded, processed, and quantitatively illustrated at each pixel of the area of interest.

Prokopakis et al described a method for objectively assessing and monitoring tissue blood supply utilizing a specifically built endoscopic imaging colorimeter that allowed quantitative color modeling of back-scattered light during endoscopic inspection in his study. This approach was used to assess tissue blood volume changes in the nasal mucosa caused by xylometazoline hydrochloride nasal spray. Quantitative imaging was discovered to be a sensitive, repeatable, and reliable method for monitoring and mapping tissue blood supply that is also simple to apply on a regular basis. Their findings revealed that saturation declines with time, making it the color parameter most susceptible to the vasoconstriction treatment (figure 5). [1]

The color parameters in the space that were studied were the hue, the saturation and the intensity. The hue expresses the predominant wavelength of a color and takes values from 0-360 which in the color sphere refer to degrees. The saturation that expresses the purity of the shade (in relation to other impurities) and takes values from 0-100%. Thus, 2 hue-like color stimuli (same predominant wavelength) may have varying degrees of white impurity and therefore different saturation. Finally, the intensity or otherwise brightness expresses the brightness of the color in relation to white and the values it receives are from 0-100%. The vertical axis of the exhibited color histograms in figures 5, 6 and 7 were given in pixel counts, while the horizontal axis was expressed in H (degrees), S (percentage), and L (percentage) units. The estimated H, S, and L medians against time curves for the 10 people were depicted in those three pictures and curves demonstrated the induced color changes during the vasoconstriction time evolution.[1]

Objective indices for optical tissue characterization and analysis appear to be promising in terms of understanding the pathophysiology of tissue alterations and objectively assessing their response to various therapy schemes. For their experiment they used an image colorimeter attached to an endoscope, that made possible a noncontact, repeatable color information capture, measurement, and mapping for any spatial point inside the endoscope's field of vision [1]. In their research they also found that the hue color parameter is virtually unaffected by the latter since hemoglobin is the only tissue chromophore that controls the dominant wavelength of back-scattered light [1].

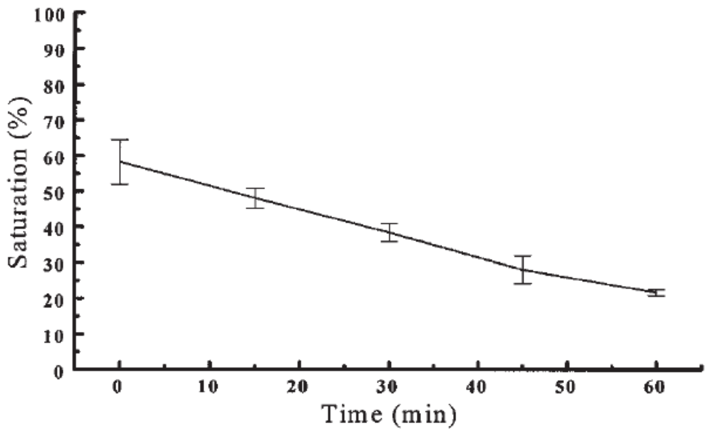


Figure 5. Saturation/time graph [1].

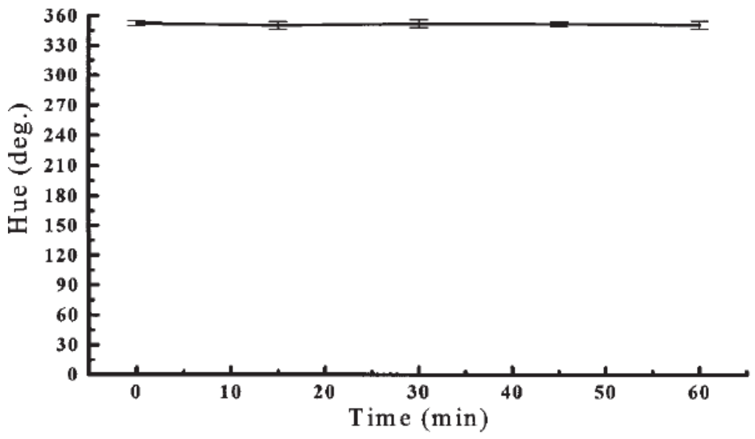


Figure 6. Hue/time graph[1].

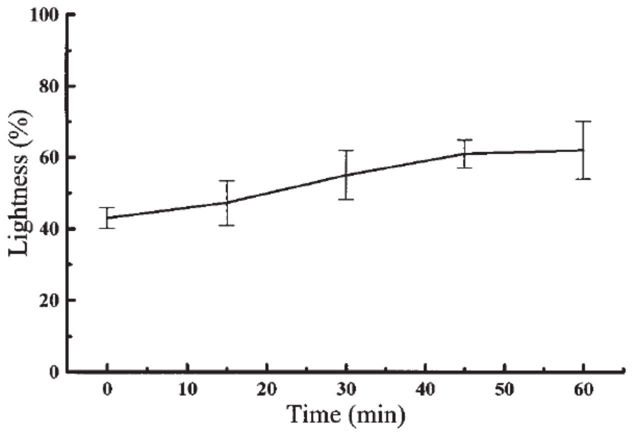


Figure 7. Lightness/time graph[1].

Helidonis et al provided an innovative solution to the challenge of non-destructive tissue lesions identification and staging. Their approach is based on a quantitative assessment of the spatial and temporal changes in light scattering characteristics caused by epithelial dysplasias and cervical and laryngeal cancers in vivo. They stated that limits of conventional approaches, necessitate the development of more effective diagnostic and screening procedures. The method they described can improve the sensitivity and specificity of in vivo diagnostics and providing lesion mapping. Spectral filtering and the elimination of surface reflection improves sensitivity, allowing for the early diagnosis of pre-malignant lesions. Furthermore, they claimed that the promising results obtained for the larynx suggest that the method's applicability might be broadened to other tissues [33].

They relied on the notion that hemoglobin is mostly responsible for the vascular plexus's color, and that by limiting red wavelengths, the back-scattered component may be reduced.

Because their back-scattering cross sections remain considerable in the shorter-wavelength region of the visible spectrum, this band-pass filtering will not impede the recording of back-scattered light from acetic acid-responsive epithelial regions. As a result of the proper spectrum filtering, the contrast between normal and diseased tissue will be maximized [33].

Still there is no objective and accurate way to diagnose and evaluate the status of the mucosa tissue. In comparison to other mucous membranes, the nasal mucosa is easily accessible and serves as a convenient entry point for tiny and big molecules. Intranasal administration has several advantages, including a quick onset of therapeutic effects, no first-pass effect, no gastrointestinal degradation or lung toxicity, noninvasiveness, essentially painless application, and ease of use by patients – especially children – or physicians in emergency situations. [3]

Many medicinal substances can be delivered locally and systemically by intranasal administration. The development of nasal products is guided by therapeutic and pharmacological concerns [34]. The natural penetration barrier and the effective cleaning process limit the overall quantity of medicine that may be absorbed, even if the epithelial tissue within the nasal cavity provides an excellent absorption region. As a result, a therapeutic agent with appropriate potency is required to achieve clinically efficacious medication concentrations at the target location.[3]

4.1 Objective Evaluation and Diagnostic Studies

Although a nasal obstruction visual analog scale can be used to provide a subjective assessment of nasal blockage, it is typically better to acquire a quantitative objective assessment. One of the earliest objective assessments of the nasal airway was hygrometry. Zwaardemaker reported this procedure in 1894, which entailed the patient breathing into a mirror. The fog created by each nostril was measured and compared. [35]

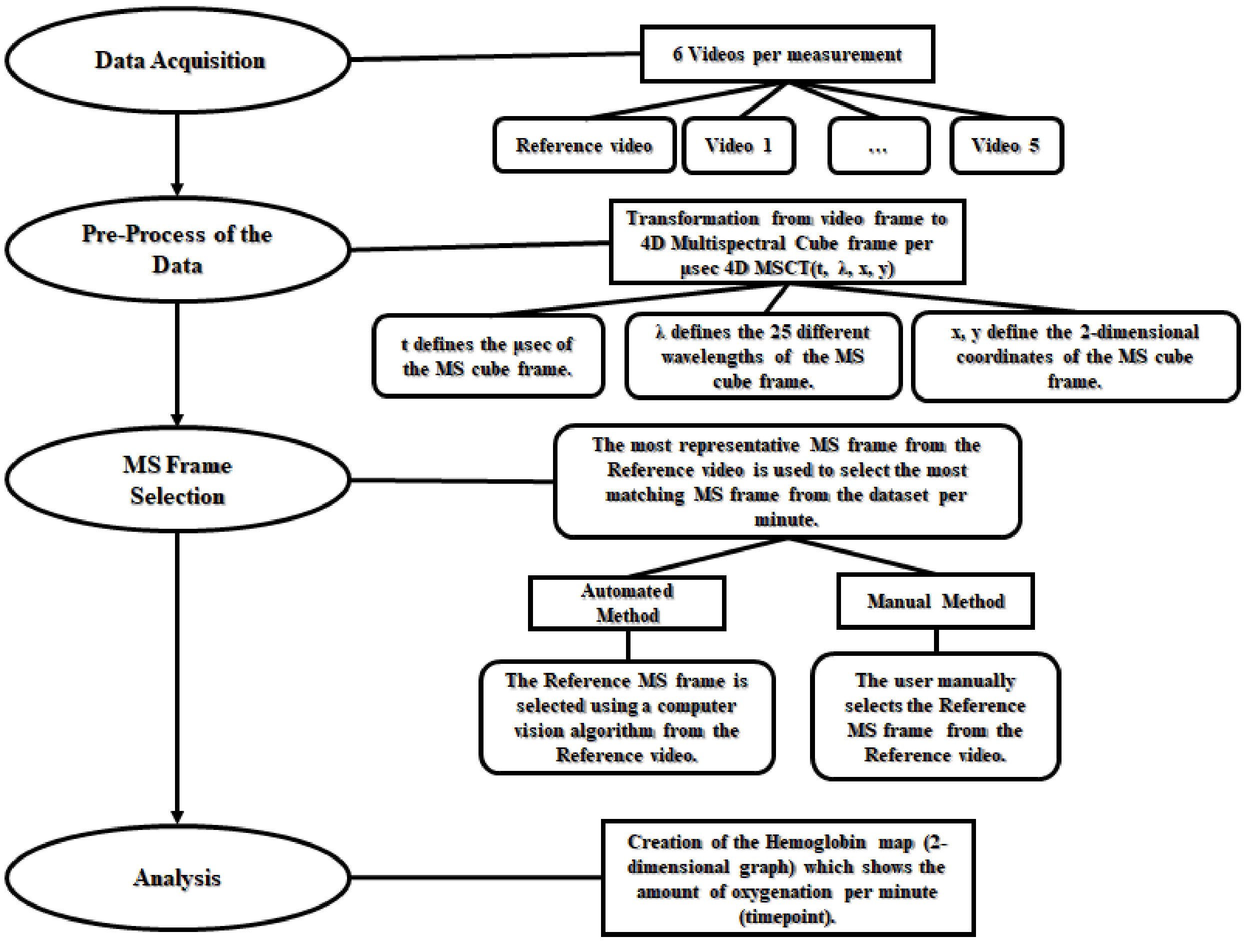
The following are some of the current ways for objectively assessing the nasal airway:

1) Peak nasal inspiratory flow (PNIF): The PNIF test is a simple, noninvasive procedure for determining nasal patency. It is a physiologic parameter that indicates the maximum nasal airflow in liters per minute during maximal forceful nasal inspiration. Because transnasal pressure differences are not recorded, PNIF is regarded to be sensitive to significant variability dependent on the subject's effort and compliance, as well as the investigator's accurate instructions. The utilization of utmost effort increases the likelihood of turbulent airflow as well. Another issue with PNIFs is that they don't represent genuine physiologic circumstances because normal breathing begins at a much lower tidal volume. Finally, because the procedure has been proven to be repeatable and consistent with other objective tests, it is indicated to be trustworthy for assessing nasal patency. In addition, respiratory comorbidities might alter PNIF results by decreasing the inspiratory effort. [35]
2) Acoustic Rhinometry (AR): Hilberg and colleagues were the first to employ AR to quantify the nasal airway's cross-sectional size in 1989. AR is a simple, noninvasive, and relatively inexpensive method of measuring nasal airway cross-sectional area as a function of longitudinal distance along the nasal route while an acoustic impulse travels along it. It's the most widely used method for determining the shape of the nasal cavity. From contiguous cross-sectional measurements, nasal passage volumes may be determined. The approach is suited for anatomic evaluation and structure of the nasal airway, medication effects, and surgical changes in the nasal cavity, including changes in the mucovascular components of the nasal valve area, as well as alterations based on disease such as nasal polyposis or septal deviation [35].
3) Rhinomanometry (RM): The measurement of transnasal pressure and airflow is part of RM, which is a functional evaluation of airflow. The mean pressure, volume, work (pressureflow), and resistance (pressure/flow) associated with each breath may be calculated using these data. Each side of the nose's resistance can be compared to each other and to overall nasal resistance, allowing the physician to determine how each nasal canal contributes to the patient's complaint. The result is an S-shaped curve, with the x-axis indicating pressure difference and the y-axis representing flow. Inspiratory airflow is the most typical means of reporting data. A pressure transducer is used to measure posterior nasal pressure, a pneumotachometer is used to measure flow, a mask is used to monitor anterior nasal pressure and flow, and a computer is used to transform these results into digital information [35].

4) Odiosoft Rhino (OR): Odiosoft rhino (OR) is a novel objective approach that translates the frequency of sound produced by nasal airflow into measures of cross-sectional area. The assumption behind the technique is that when turbulence rises, nasal airflow produces a higher frequency sound. A microphone, nasal probe, sound card, and computer are used in this noninvasive technology created by Seren. The nasal probe is linked to a microphone 1 cm from the nostril, and the participant is instructed to seal the other nostril in order to avoid any distortion of the test nostril. Unlike acoustic rhinometry, which utilizes reflected noises to compute nasal cross-sectional area, the odiosoft rhino method directly measures the sound generated during breathing. When compared to AR, this approach provides a sensitive and precise evaluation of nasal airway patency with a greater connection to patient symptom levels, according to a recent research published in 2006 [36]. Although these findings are encouraging, the pursuit for the ultimate objective testing technique continues.[35]

5 Materials and Methods

The flow of the methodology of this thesis is illustrated in figure 8. As it can be seen initially it starts by acquisitioning the data, each measurement of each subject is consisted of 6 videos, the first one being the reference (default) video. The reason we specify the first video as the reference video is because the pharmaceutical agent has not yet been applied on the subject. Next the preprocessing of each measurement takes place, each frame of the video contains information of 25 spectral wavelengths and needs to be cropped, thus a transformation of the data is needed for better accessibility and easier processing. So, in the pre-processing takes place the transformation from video frame to 4-dimentioanal multispectral cube frame 4D MSCT(t, λ, x, y), t being the μsecond of the MS cube frame, λ defines the spectral wavelength and x, y are the 2-dimentional coordinates of the MS cube frame of each spectral wavelength. After the preprocessing the processing takes place, which is essentially the multispectral (MS) frame selection, in which the most representative frame from the reference video is used to select the most matching MS frame from the rest of the videos where the pharmaceutical agent has been applied. In this thesis two methods have been presented and executed, the first one being the Automated method and the second one being the Manual method. In the manual method the user selects a reference MS frame of its preference from the Reference video, while in the Automated method the reference MS frame is selected automatically from the reference video using computer vision. Finally, the last step in this thesis methodology is the Analysis of the results that were found after the MS frame selection. The results can be seen in the hemoglobin map, which is a 2-dimentional graph that shows the amount of oxygenation per timepoint and in this thesis case minute.



Data Acquisition
6 Videos per measurement
Reference video
Video 1
...
Video 5
Pre-Process of the Data
Transformation from video frame to 4D Multispectral Cube frame per μsec 4D MSCT(t, λ, x, y)
t defines the μsec of the MS cube frame.
λ defines the 25 different wavelengths of the MS cube frame.
x, y define the 2-dimensional coordinates of the MS cube frame.
MS Frame Selection
The most representative MS frame from the Reference video is used to select the most matching MS frame from the dataset per minute.
Automated Method
Manual Method
The Reference MS frame is selected using a computer vision algorithm from the Reference video.
The user manually selects the Reference MS frame from the Reference video.
Analysis
Creation of the Hemoglobin map (2-dimensional graph) which shows the amount of oxygenation per minute (timepoint).

Figure 8. The structure of the methodology

5.1 Subjects

Twelve healthy people (subjects) without any history of rhinosinusitis or current inflammation were included in the study. Exclusion criteria also included the use of corticosteroids, decongestives and/or antihistamines within the past 30 days. For control purposes, time “zero” is considered as the measurement of nasal mucosa before the application of any pharmaceutical agent. Measurements were performed with the use of a multispectral imaging camera coupled to an endoscope to better characterize blood spectral features. Each measurement is a collection of recorded videos and was completed in 35-40 minutes. The recordings of each measurement were continuously performed for the first 6 minutes, followed by acquired short term video sequences (~2 minutes each) every 6 minutes.

5.2 Instrumentation

The camera that was used is the XIMEA MQ022HG-IM-SM5X5-NIR (figure 9) that has 1088-pixel height and 2048-pixel width. However, the Active sensor area has an offset x = 0 pixel, y=3 pixel, which means that the actual width is 2045 pixels and height is 1080 pixels, since the NIR filter pattern is 5x5 as we can see on the Table 1. This camera has a small size of 4cm*4cm*3cm. As we can see in figure 10 the camera is connected, with the optical fiber, a light source and connected with a USB3.0 interface cable to the laptop. Then an acquisition software records the frames from the camera and visualizes them so that the doctor can have a good perception throughout the measurement. A screenshot of the data acquisition software is presented in figure 11.

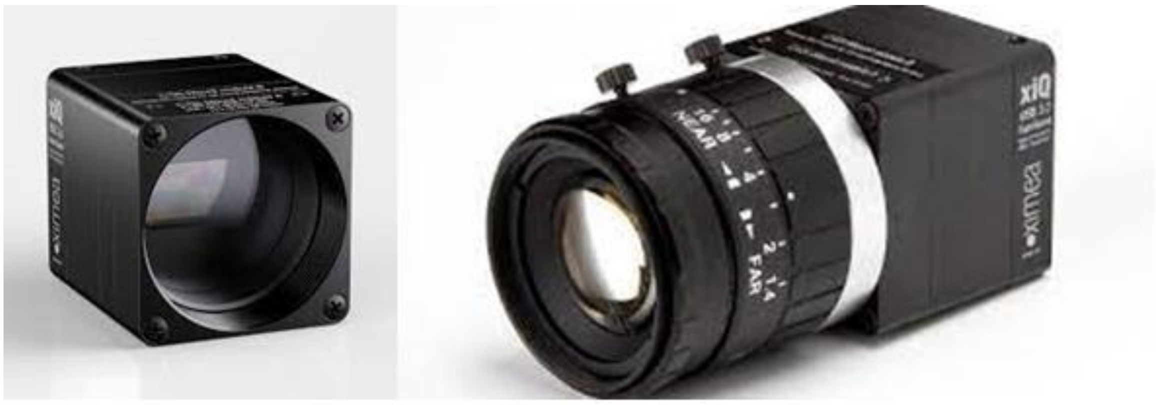


Figure 9. XIMEA camera MQ022HG-IM-SM5X5-NIR. This is the multispectral snapshot camera we used to acquire the data

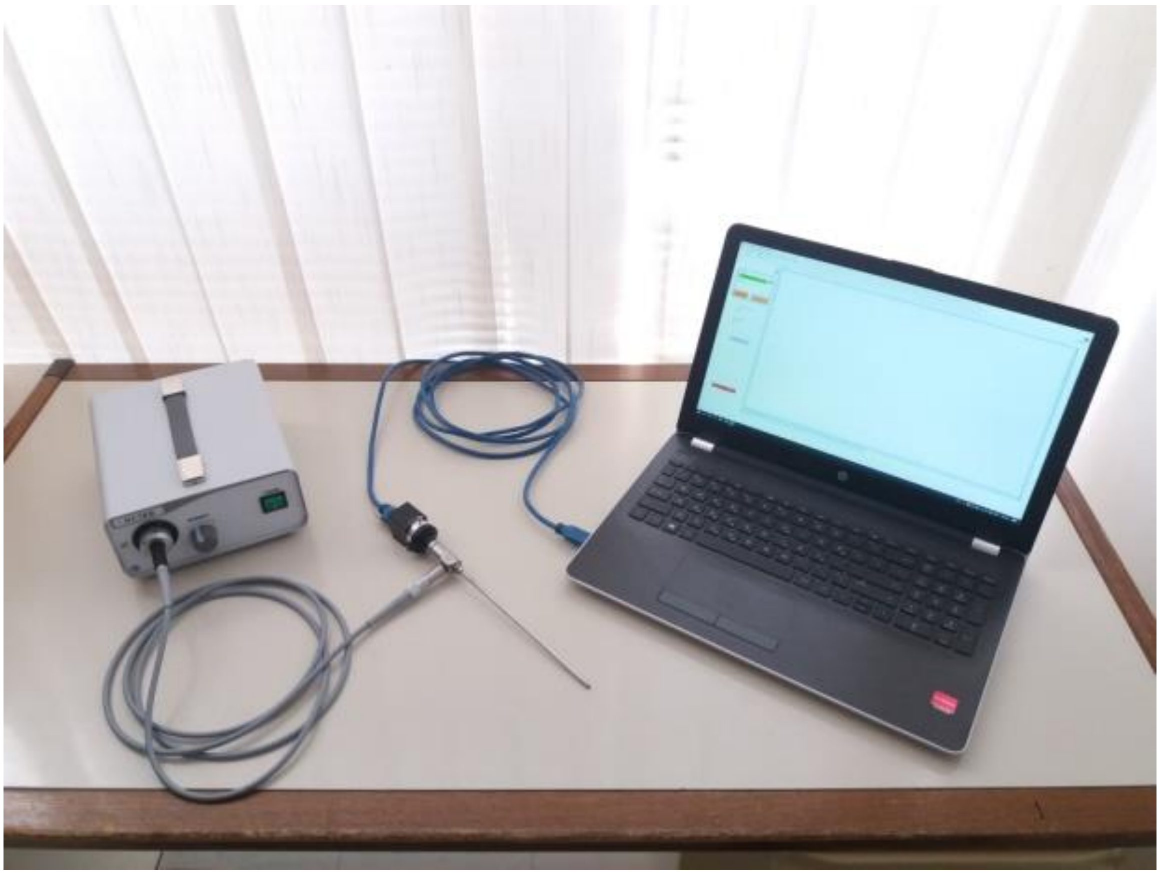


Figure 10. The experimental setup. In this image it is represented the XIMEA multispectral IR sensor, an optical fiber, a light source and connected with a USB3.0 interface cable to a laptop.[33]

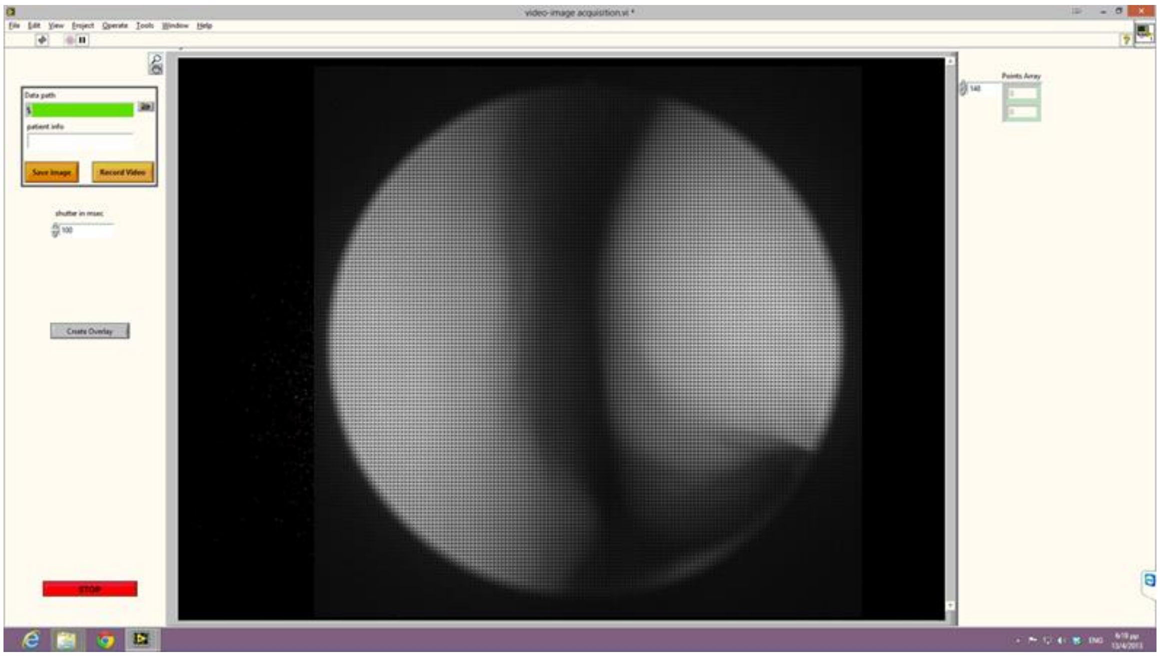


Figure 11 Acquisition Software.[33]

Table 1: Pattern wavelength values of the XIMEA camera (25 spectral bands between 650-900nm)

900	*909*	*892*	*882*	*683*
809	*821*	*797*	*784*	*693*
759	*772*	*746*	*732*	*708*
943	*949*	*935*	*927*	*975*
861	*873*	*852*	*840*	*955*

The camera was used to record videos instead of single photos for a number of reasons. Firstly, finding the exact same angle of view and the same distance from the nasal mucosa is extremely difficult since there is a constant movement from both the doctor and the subject. They cannot stay still for 6 continuous minutes and even if they could stay still, breathing creates significant movements in the micrometer resolution scale the camera visualizes the tissue. Moreover, after the application of the pharmaceutical agent the mucosa changes its physical dimensions, and in particular is getting smaller, so the whole field of view is different. Lastly, the skills the doctor has in the acquisition of the data play a significant impact.

To minimize the acquisition problems and acquire a correct dataset, a significant amount of time might be needed for the doctor to learn the techniques. All those reasons led to the use of video, since through the acquisition of multiple images, instead of one per minute, the extraction of optimal frames could be achieved, resulting in the minimization of the movement.

The necessity to use this multispectral camera was to record the reflection spectrum from the tissue and calculate the hemoglobin reflection map. This multispectral camera was sensitive in the spectral range from 600 to 900 nm. In figure 12 we can observe that the absorption of $Hb0_2$ and the Hb is significantly different especially in the bands 642 and 870 nm. The blue color represents the venous blood (de-oxygenated) and the red the arterial blood (oxygenated). From the inverse contrast of the curves' reflectivity presented in the two aforementioned wavelength bands, we can calculate the ratio, which will define a larger dynamic range between the oxygen saturation in blood. In particular, this ratio is the division of the specific band images, which results in a map of the blood oxygenation.

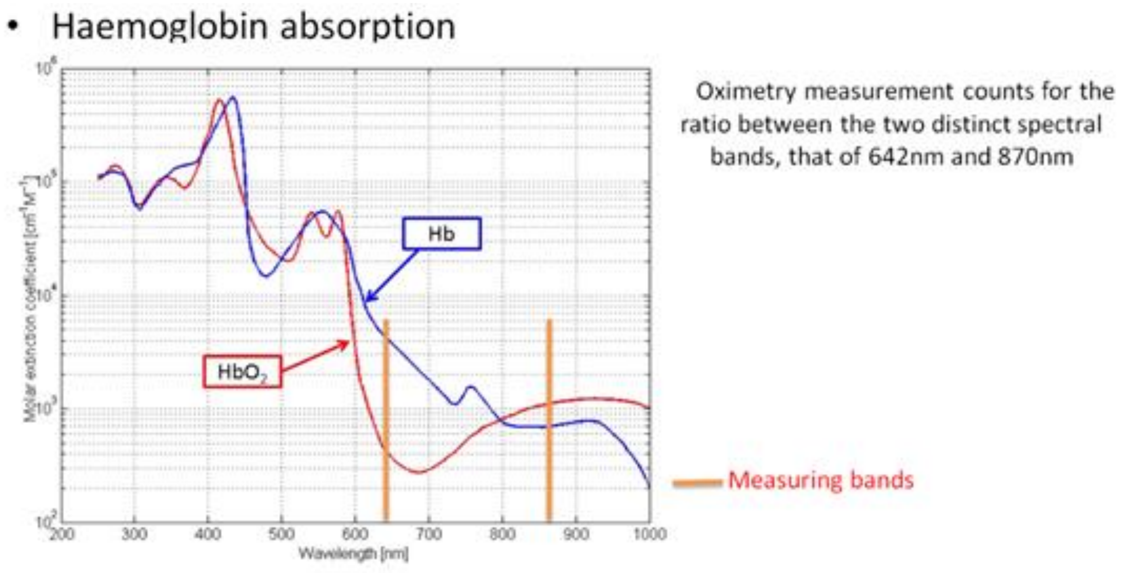


Figure 12. Hemoglobin oximetry [37]. *Relative Intensity Measurements: Blue line represents the venous blood (de-oxygenated) and the Red line represents the arterial blood (oxygenated).*

5.3 The software written for the data processing and analysis

The code for this thesis was written in python version 3.9 and it is presented in the appendix. In the Table 2 the most important python library versions are presented. The *opencv* library was used to process and analyze the images and the videos, matpot.lib to visualize the results in 2 dimensional or 3 dimensional graphs. Moreover, *NumPy* was used to make the code faster, since the processing of *NumPy* arrays is faster than the python lists and *numpy* library has a lot of functions included that we used in our code. For example: *numpy*.where(condition) that returns the elements that satisfy the condition inside the parenthesis from an array, *numpy*.average() that computes the weighted average along the specified axis, if the weights are not inserted it is just the average of the array that was put as input and returns the outcome, numpy.float32() which was used as a mask for the data (it transforms all the elements into type float 32), *numpy*.delete() that returns a new array with sub-arrays along an axis deleted (we used this function to crop our images), *numpy*.sort() that returns a sorted copy of the array that we entered, *numpy*.amax() that returns the maximum of an array or maximum along an axis and a lot more. In addition, the *Scikit-image* library that contains a collection of algorithms was used for image processing. It was specifically used it to calculate the P.S.N.R. metric(skimage.metrics.peak_signal_noise_ratio()).

Table 2. Library versions:

Scikit-image	*1.6.2*
Matplot.lib	*3.4.2*
Numpy	*1.20.2*
Opencv	*4.5.2*

5.3.1 Algorithms

Data collection and analysis were carried out using software created expressly for this study. The methodology is based on changes in blood oxygenation absorption, which correlates to the inferior turbinate's spatial blood concentration. The code of the software we developed as well as the user manual are presented on the appendix.

After the data had been collected, a software was used to preprocess and analyze the data. Every measurement is consisted of six videos, the first one is the Reference video, since the pharmaceutical agent has not yet been implemented to the subject. The second video which is the largest in time (lasts about six minutes) is taken after the pharmaceutical agent has been implemented. The rest of the videos deviate from each other for six minutes and have a duration of approximately 2 minutes.

5.3.2 Pre-processing:

Each video is saved with a specific name. The name format is: name of the subject"_" day"-"month"-"year"-"hour"-"minutes" " am or pm. For example: Patient1_14-3-2018 11-21-28 am, Patient2_14-3-2018 12-37-37 pm. Every frame in each video is preprocessed so that is has the appropriate form in order to be processed.

Initially, each frame is saved in a 2-dimensional array and append each one of them in 1 array. So, we are going to have an array that in each element has 2-dimentional arrays, in other words a 3-Dimensional array. Moreover, the representative time-point of each frame is going to be calculated and stored in a 1-dimensional array. This way we are going to know the exact time of each frame. The algorithm takes into consideration the frames per second that were used to capture the data.

After that it is important to crop the frames as it can be seen in the figure 13. The area that contains the information is inside the circle, the rest of the image is not important, because it is black thus contains no information. The Sensor is covering all the field of view, but the optical fiber is covering just a circle in the center, if we had a bigger fiber, it would be better since the field of view would be bigger. Cropping the image is going to make the analysis and the processing a lot easier and faster. The cropping of every frame is performed by removing pixels in multiples of 5, because the Snapshot Mosaic is 5 x 5 and we also take into account the fact that the Active sensor area has off set x = 0 and off set y=3. From an image with a width of 2045 pixels and a height of 1080 pixels we end up having an image with size: 875 pixels x 875 pixels. The original frame of each video (2045 pixels x1080 pixels) has 2,208,600 pixels and the cropped one has 765,625 pixels, we remove 1,442,975 pixels from each frame. Since there are less pixels, the processing of each frame is faster and the memory that the program occupies is going to be significantly less.

Following, the separation of each wavelength is necessary, the cropped frame contains a 5x5 pattern that contains 25 different wavelengths (Table 1). Each frame is transformed to a Multispectral cube MSC(λ, x, y). Each frame has 25 images clustered in an array pattern (figure 14). We need to read the individual images from the video frame, use them as different points in the spectral cube. From the 2MPixes (2.228.224 pixels) of each grayscale frame (8bit), we will generate a new 3D MS cube that will have (89129 pixels in spatial domain and 25 in the spectral domain). This will generate a cube of the following dimensions (λ, x, y). Then each cube (for each frame) is going to be merged to create a 4D MS cube which is going to be MSCT(t, λ, x, y,). In which t represents the time of the frame, λ the wavelength in nm of the MS cube and the x, y are the rows and columns of the 2-dimensional images. In other words this 4-dimensional array

MSCT contains all the 3-dimensional MSC(λ, x, y) frames of a measurement. The new cropped images of every MS cube have a size of 175 pixels x 175 pixels ((875/5) x (875/5)). As it can be seen in figure 14 the spectral images present an intensity variation across the wavelengths, strongly related to the absorption of hemoglobin.

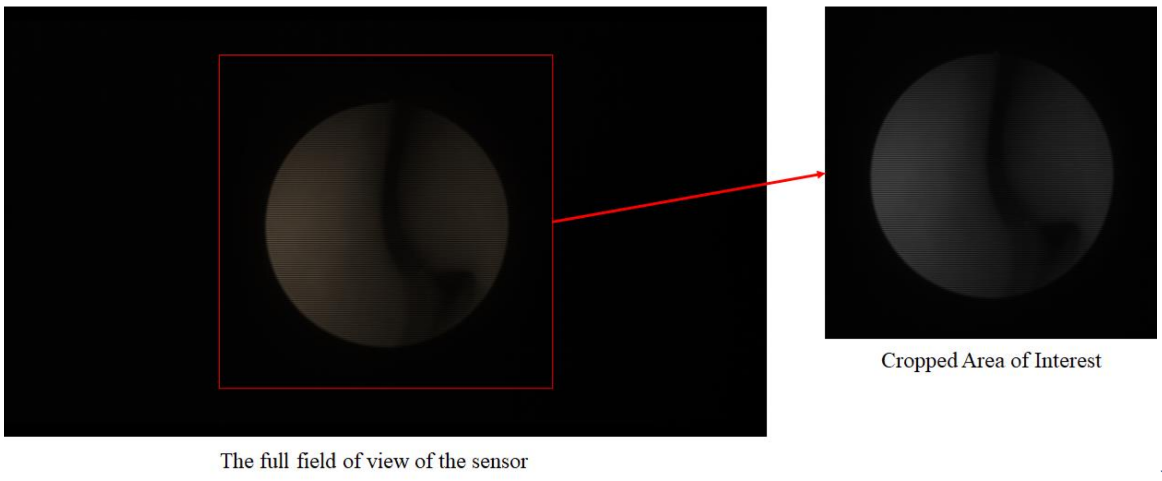


Figure 13. Cropping to the area of interest

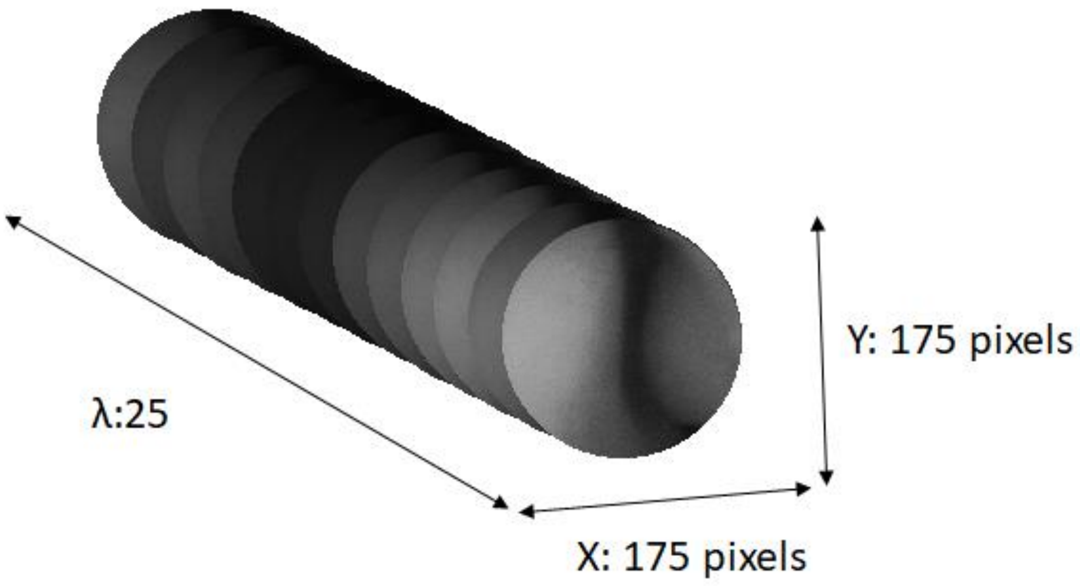


Figure 14. The 25 different spectral images acquired across all the sensitivity wavelength range. 3-Dimentional Multispectral (MS) cube.

5.3.3 Multispectral Cube Frame selection:

Initially, we need to define the best frame to use as a reference from the first video. In other words, we need to find the most representative Multispectral (MS) frame from the Reference video (where no pharmaceutical agent is introduced). The most representative MS frame from the reference video is used to select the most matching MS frames from our dataset per minute. There are 2 methods we created for the selection of the default image, the one is manual and the other one is automated.

1)Manual Method:

The user can select from the first video (where no pharmaceutical agent is introduced) the cropped frame (FULL PATTERN, original frame of the first video) that will be used as a reference for the best viewpoint for the analysis. In other words, the user is going to select an frame from the Reference video, that he or she thinks as the most representative, before we create the MSCT(t, λ, x, y). This way is fast, but the image that the user will select might not be the most representative one, so the result might not be so accurate, and it could also be objective, because each user might have a different opinion in which one is the most representative. In our approach we define as most representative the frame that has the same angle of view with most frames of the current measurement.

2)Automated Method:

In this method an automated algorithm will calculate the most common angle of view across all MS frames of all videos. Every MS frame from the Reference video, is going to be compared with every frame that is contained in the rest of the videos with the PSNR metric. The comparison is happening on the image wavelength 746 nm (figure 15) of every 4-dimentional MS cube frame. After each comparison we save the sum of the results of the PSNR score for every MS frame. Since the higher the PSNR score of the result of the comparison between the 2 images, the closer the images are, the frame of which the sum is higher is going to be the default image. This way is exponentially slower from the manual one, since the selection is more complicated, but the Reference image it selects it's for our way of selecting (PSNR metric) more precise and accurate than the manual method. The only downside of this mode is that if the noise in the dataset is more than the information, the default image might not be accurate.

After the MS frame selection of the Reference MS frame a comparison of it with every MS frame from the rest of the videos from that measurement is going to take place with the Peak-Signal-to-Noise Ratio (PSNR) metric in order to find the images with the higher score per minute| (timepoint).

PSNR is formally defined as:

$$\text{PSNR}=log_{10}\left(\frac{MAX_I^2}{MSE}\right) \text{ (ii)}$$

Squared Error (MSE) is formally defined as:

$$\text{MSE}=\frac{1}{n}\sum(y-\bar{y}\ )^2 \text{ (iii)}$$

Where $y-\bar{y}$ are the squares of the difference between the image that is being compared and the image that is being compared with. The MAX_I represents the Max value of the image.

After the MS frame selection of the default image, each MS frame from the rest of the dataset is going to be compared with the default image using the PSNR metric. The image with the higher score in every minute will be extracted in a new folder containing a filename with the subject name and the related time-point. In case of a tie between 2 or more images, all of them are going to be extracted. The comparison is happening after we have pre-processed fully the data and we compare on the image wavelength 746 nm (figure 15). The wavelength 746 nm is one of the bright ones, having a darker pattern would not depict the contrast and the differences around the mucosa. The selection of this "filter wave" is objective, someone could have picked another one bright wavelength, as long as it is not a dark one. For instance, any of the following could be a good choice 759 nm, 772 nm or 784 nm, but not the 892 nm or 900 nm. For the comparison it is used one wavelength image, which means that it is faster than using the cropped image with the "FULL PATTERN", since it is smaller. However, this algorithm all together is not really fast because it has to run for every MS frame of our dataset.

The user should not select the Reference video he or she used in order to find the Reference image for the MS frame selection, because if he or she does select it, it will cause the program to crash. If the algorithm compares two identical images with the PSNR metric, the value it returns is infinite and cannot be saved on a variable.

Each video will then be cropped, and the name of the video is going to be sliced so that the required data are going to be used for the process. The processed and selected images are going to be saved on a folder named "selected". The name of its image is going to have the following format: hour-minute-second-μsec_wavelength.png (12-38-41-5_683.png).

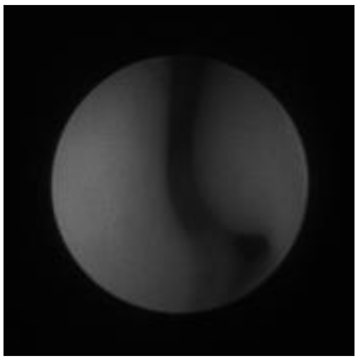


Figure 15. 746 nm spectral band

5.4 Analysis

After the MS frame selection, with each selected MS frame the calculation of the intensity of hemoglobin oxygenation is taking place. In order to calculate the intensity of hemoglobin reflection per minute (timepoint), the two distinct spectral image bands (in our case 680 and 873 nm) are divided for every frame that was chosen by frame selection for each minute (timepoint). It is obvious that after the division the intensity of each pixel is going to drop so it is multiplied by 100 so that the divided image can be bright and visible. The result is shown in the figure 16. This division is also going to calculate the oxygenated hemoglobin distribution map.

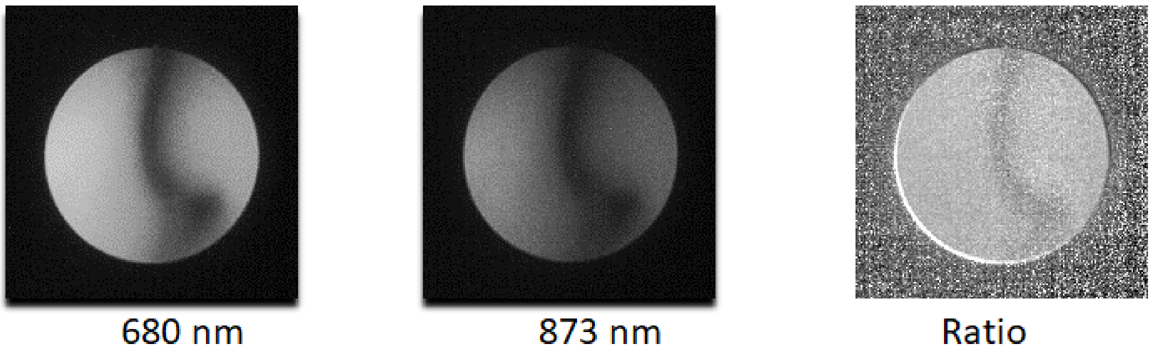


Figure 16. The 680 nm, 873 nm enhanced spectral images and the calculated ratio image. The ratio of 680/873 nm is related to the Hemoglobin oxygenation. The ratio image is the outcome of the division of the 680 nm by the 873 nm enhanced spectral image. This division is going to be the oxygenated hemoglobin distribution map.

In the MS frame selection, it is mentioned that the frame or frames with the highest score per minute are being saved. In case of more than one frames in one minute the average of each average is taken and the time that point has is also the average of the time these frames had. The averaging takes place in order to have 1 point in every minute.

Moreover, for the analysis it does not matter whether in the MS frame selection it was used neither the Manual nor the Auto Method. Both methods result in the same type of data in the same name folders. That’s why an important note would be to name the project based on the method of the MS frame selection that is going to be used. For example, for the Auto method a name that could be used would be “Subject1_ayto”.

After the division of the two wavelength images (figure 16) a small, but distinct area of the inferior turbinate is selected (figure 17). The area that is selected has a shape of a 5x5 array, a total pixel size of 25. The average of this area represents the oxygenation of hemoglobin at that particular minute.

Time is transitioned from the actual time that the data were obtained to a scale that starts from 0 to 40 minutes since the process lasts 35-40 minutes. Moreover, the intensity in every map is around 0.7 and 1.4 so the y axis of the intensity had to start in 0.65 and end in 1.45. Time and intensity are being processed in order to have a better observation of the data.

The first timepoint in the 2-Dimensional graph is the reference timepoint of the reference image that was a frame obtained before the application of the pharmaceutical agent. The graph that is going to be the result is going to look like figure 18, it will have in the vertical axes (y) the intensity and on the horizontal (x) will have the time measured in minutes. This graph is called Hemoglobin Oxygenation Dynamic Graph, since it depicts the Hemoglobin Oxygenation changes through time(timepoint). In this study the measurements of the HOD are calculated per minute, since as mentioned before with frame selection the MS frame with the highest score each minute would had been selected.

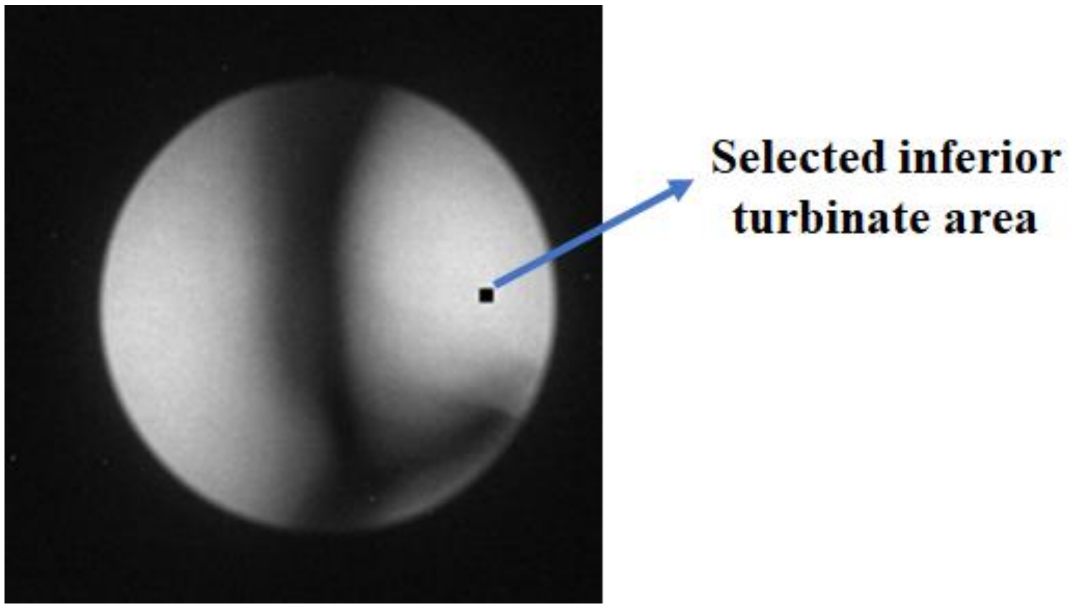


Figure 17. The selected by the user 25-pixel area of inferior turbinate area.

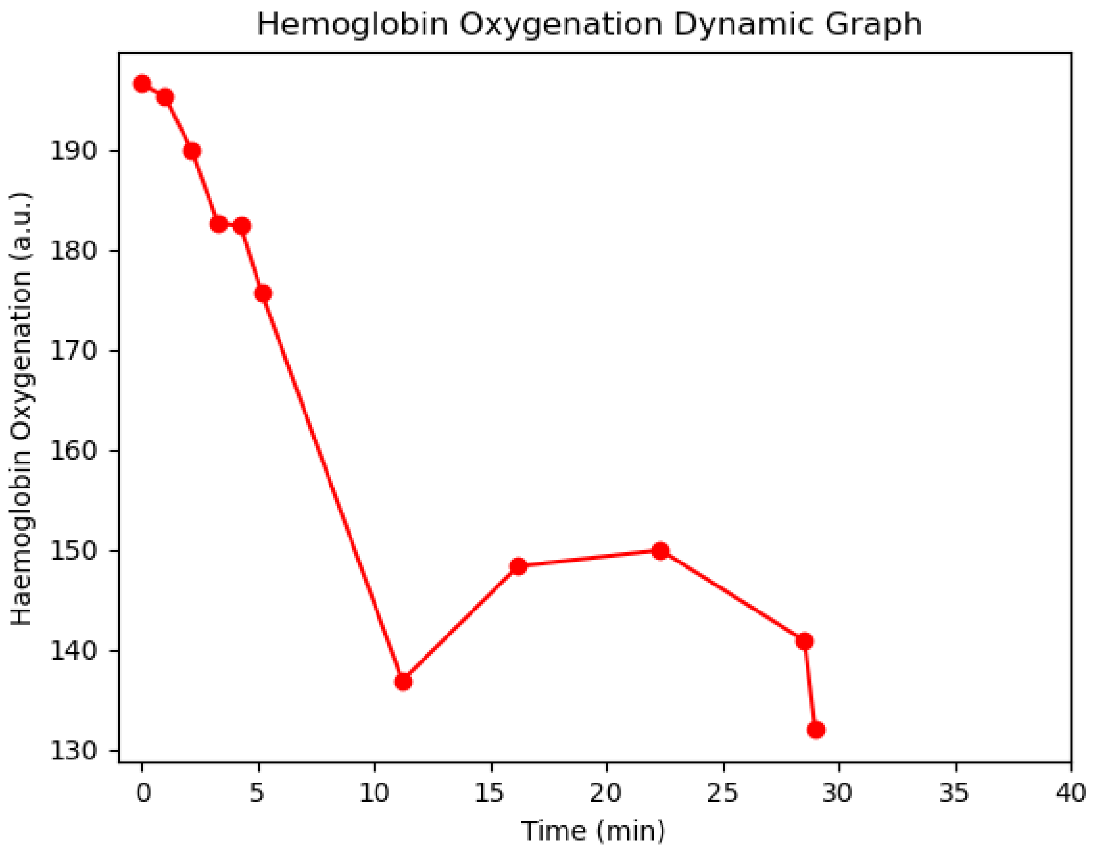


Figure 18. An example of a Hemoglobin Oxygenation Dynamic Graph (HOD Graph).

Lastly, the algorithm that was created produces a 3-Dimentional graph. In the 3-D graph the average of the conchae of the inferior nasal turbinate area is depicted through time for each wavelength, In each of the three graphs the intensity is normalized by dividing them with the same value that were found for the default image. In other words, the value of average of the inferior nasal turbinate that would have on the first minute it is going to be divided with the value of average of the inferior nasal turbinate that would have on the default image. For the zero time in the 3-dimensional graph we represent the default image. Since every point is normalized with the default value the value of the zero time is 1(it is divided with itself). The 3D graphs (figure 19) were created to obtain in which spectrum was the largest dynamic change in the measurement through time. In one axis there is the time with a range from 0-40 minutes, in the second one is the intensity and on the last one we have all the spectrum bands, in other words our wavelength. Moreover, the comparison of each wavelength with its neighbor wavelengths (the wavelengths of other timepoints) is also important to understand the changes during time. This 3D graphs are called Spectral reflectivity dynamics (SRD) graphs because they represent the changes that happen through time (per minute) for all the spectral reflectivity's (wavelengths).

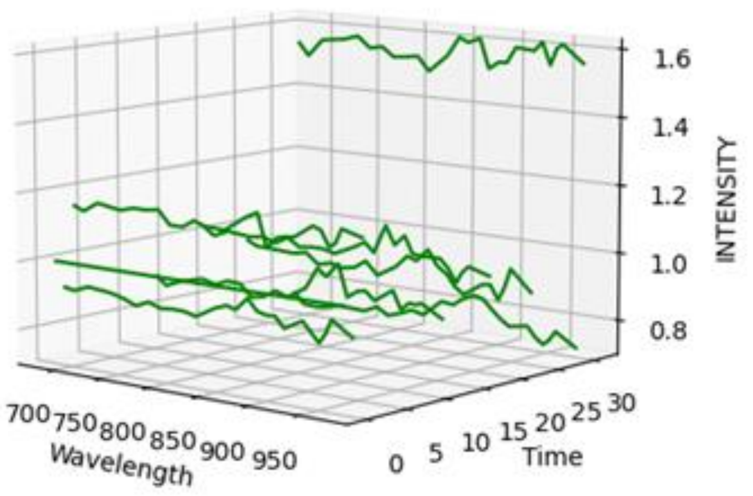


Figure 19. An example of a Spectral Reflectivity Dynamics Graph (SRD Graph).

6 Results

6.1 Sensor Validation

Unfortunately, there were multiple patterns provided by the company (Ximea), and we were unable to convince them to send us the correct pattern. For this purpose, the testing of all the patterns (3 patterns) we had in our hands was necessary, to identify the correct one. To do this validation, we used two different monochromatic light sources (LED) that we know their illumination wavelength. The pattern that was selected Table 1 presented the following results as seen in figure 20. The correct wavelength pattern is critical to our research, so in order to have a correct spectrum we had to validate it.

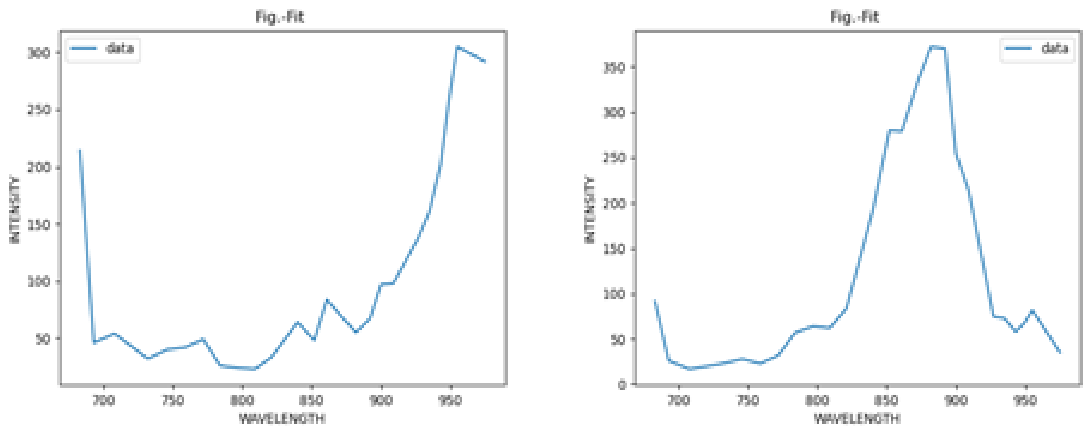
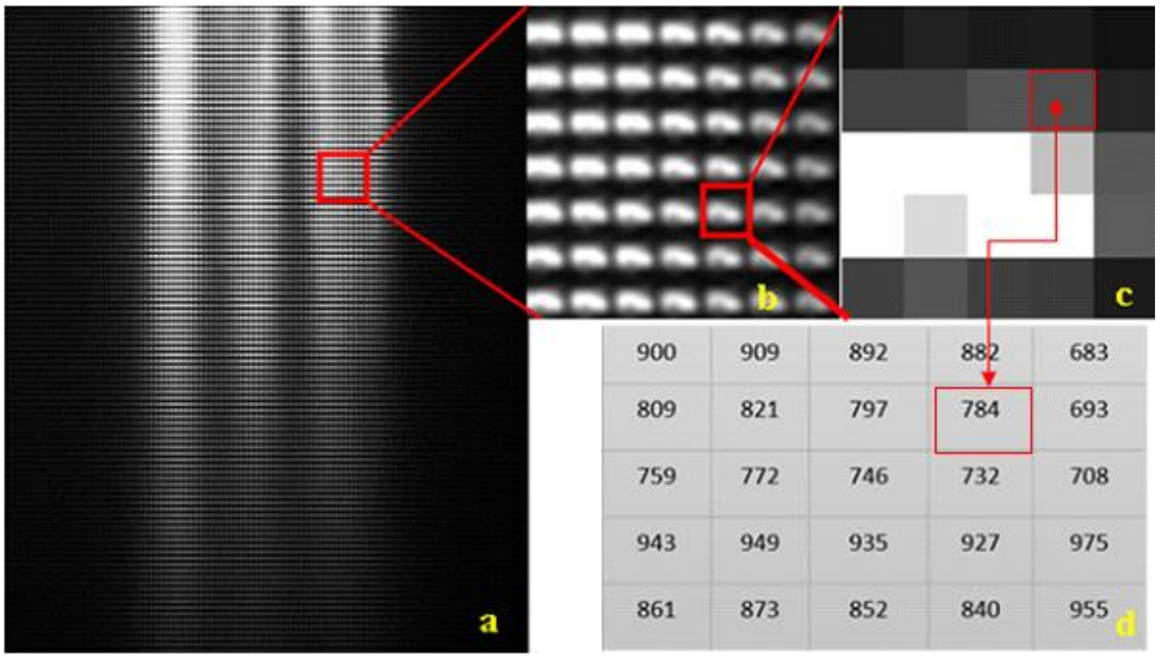


Figure 20. In the first image there is the 950 nm wavelength and in the second one is the 850 nm flashlights.

As discussed for the validation there were used 2 flashlights, the first one was transmitting light of the 950 nm wavelength and the second of the wavelength 850 nm. images were taken in order to choose (figure 20) from which an area was cropped where the flashlight is transmitting (figure 21), but the intensity was not saturated, and it was not black either and of course it was in the active area of the circle. That image was cropped even further to an image of size 25 pixels as our pattern. The intensity of each pixel was then presented in a graph based on the value of the wavelengths that are represented on the table 1. Since the flashlights were transmitting a specific wavelength, the intensity should have each max values around that area. After the outcome of the validation, it was reassuring that the pattern of the table 1 was the correct one.



900	909	892	882	683
809	821	797	784	693
759	772	746	732	708
943	949	935	927	975
861	873	852	840	955

Figure 21. a: the image that was used to validate the bands, b: smaller image, c: the pattern in pixels the table represents the wavelengths of the pattern

6.2 Selection of the most representative wavelengths

In order to identify the best wavelength bands to perform this analysis, the visualization of the spectral changes that were happening during the pharmaceutical agent effect should be examined. It was found that the best wavelength bands that were expressing dynamic changes through the measurement time were the 680 and 873 nm. These values were very close to the to the ones used for blood oxygenation measurements in medicine.

More specifically, in figure 22 the 3-dimentional Spectral reflectivity dynamics (SRD graph) graphs for one dataset can be observed. In the figure 22, 4 SRD graphs are presented each one from a different subject. The 3-dimentional SRD graphs were created in order to identify the wavelengths that had the largest variations. The division of the wavelength with the larger spectral fluctuations with the wavelength that has the lowest spectral fluctuations is going to result in the maximum contrast of the phenomenon for this study. In other words, for each subject is found where there was the spectral variation over time. The conchae of the inferior nasal turbinate is the tissue that is analyzed.

In the 3D spectral data, we are using the time 0 (reference) measurement to normalize all the following measurements. This is done in order to exclude from our analysis the physiological parameters of the tissue. The 3D SRD (Spectral reflectivity dynamics) graph allows us to identify visually the wavelengths where reflection variations are maximum and minimum. By calculating the ration of those two wavelengths we are removing the contribution of the distance between the tissue and the optical fiber, which significantly affects the intensity. This results in a clear measurement of the dynamic effects from the pharmaceutical agent on the tissue of nasal mucosa.

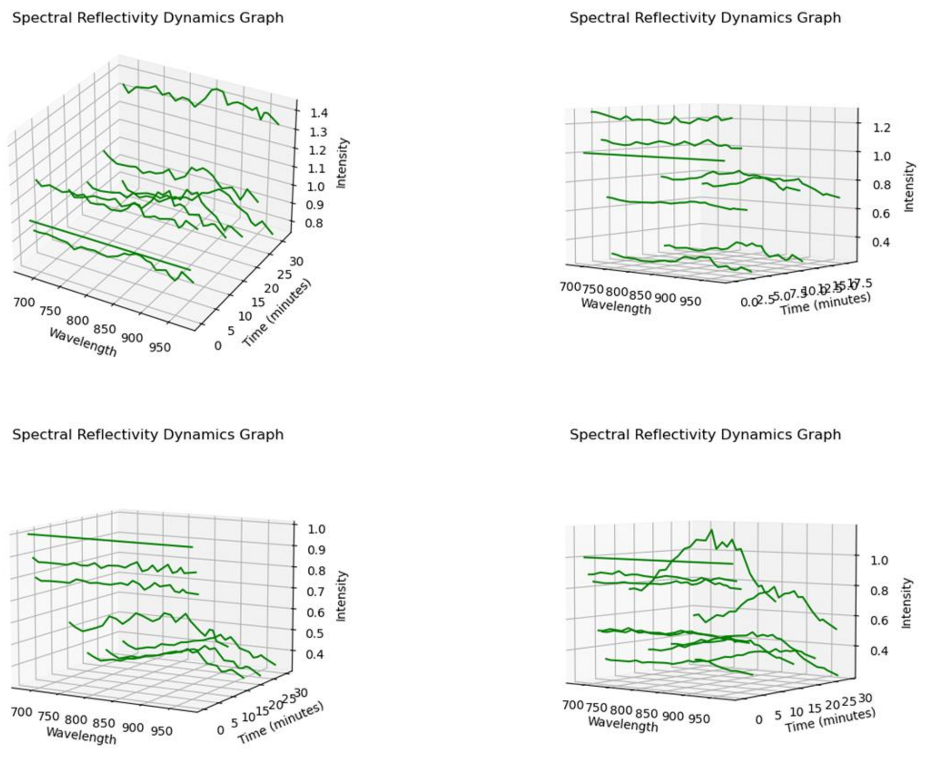


Figure 22. The results of Spectral reflectivity dynamics (SRD graph) 3-Dimensional graphs of 4 subjects. Each SRD graph represents a different subject.

6.3 Nasal mucosa responses

The dynamics of the blood oximetry of the nasal mucosa were successfully recorded across the experimental period of 30mins. Although the dynamic resolution and dynamic range is sufficient, nasal mucosa responses under the effect of the pharmaceutical agent followed a gradual decline in most of the subjects. The inconsistencies are most probably due to the personalized response of the subjects to the pharmaceutical agent.

In figures 23 and 24 the default images, that were selected for the auto and manual algorithm, are presented and the small area of the conchae of the inferior nasal turbinate is highlighted. From the default images it can be observed that there is a large diversity in the shape and brightness of the conchae of the inferior nasal turbinate as well as its surroundings. Moreover, since there are two inferior nasal turbinate tissues in the human body, in the right and left nostril, the inferior nasal turbinate could either be on the right or the left side of the image.

For the diagram to be correct the selection of the areas is really important. As it is presented in the figure 17 the selection of conchae of the inferior nasal turbinate area is on the right side of the inferior turbinate area in case the measurement is taking place in the other nostril the conchae will be on the opposite side. As expected by the literature [1], the conchae is going to shrink over time so the area that is to be selected, should not be pointing to a black area after all the shrinking.
Between the two algorithms that were created for the selection of the default image (manual and auto) the diagrams similar (figure 23 for Auto and figure 24 for Manual method). In some datasets both the doctor and the program selected the same default image, which could result in some graphs to be the same in case the selected area of the conchae of the inferior turbinate is also the same, if the selected area is different the HOD graph is going to be different as well.
In some graphs at the end of the 30-minute period, the intensity has almost the same value as the default image, or the intensity drops drastically. This means that the intensity has come back to its normal form before the application of the pharmaceutical agent or that the intensity has dropped after the 30-minute period. there are and some cases where the intensity has risen to a higher value. During this 30-minute period there are a lot of ups and downs in the values, but there is mostly a drop in intensity values.
The selection of only a single area point of the conhae of the inferior turbinate might not be correct and the result might not the anticipated one. Since the selection of the inferior turbinate area is really important and determining in the result of the Hemoglobin oxygenation, it was decided to select more than one area of the conchae of the inferior turbinate and average the results in order to calibrate the outcome. Of course, the results were better using the average of exactly 5 selected areas for each subject. The results were as anticipated that the mucosa would react and can be seen in figures 29-32. Almost all of the graphs have their values declining.
Responses in the following figures are calculated in two methods. First in Figures 25-28, one area of the mucosa is considered, which consists of 5x5 pixels, in the 25 and 26 the manual algorithm was implemented and in the 27 and 28 the auto algorithm was implemented. In the second method, as seen in Figures 29-32, five different areas on the mucosa are considered each consisting of 5x5 pixels, in the figures 29 and 31 was implemented the manual and the remaining 2 the auto algorithm. This was done as said before to reduce the potential statistical error of the single area.

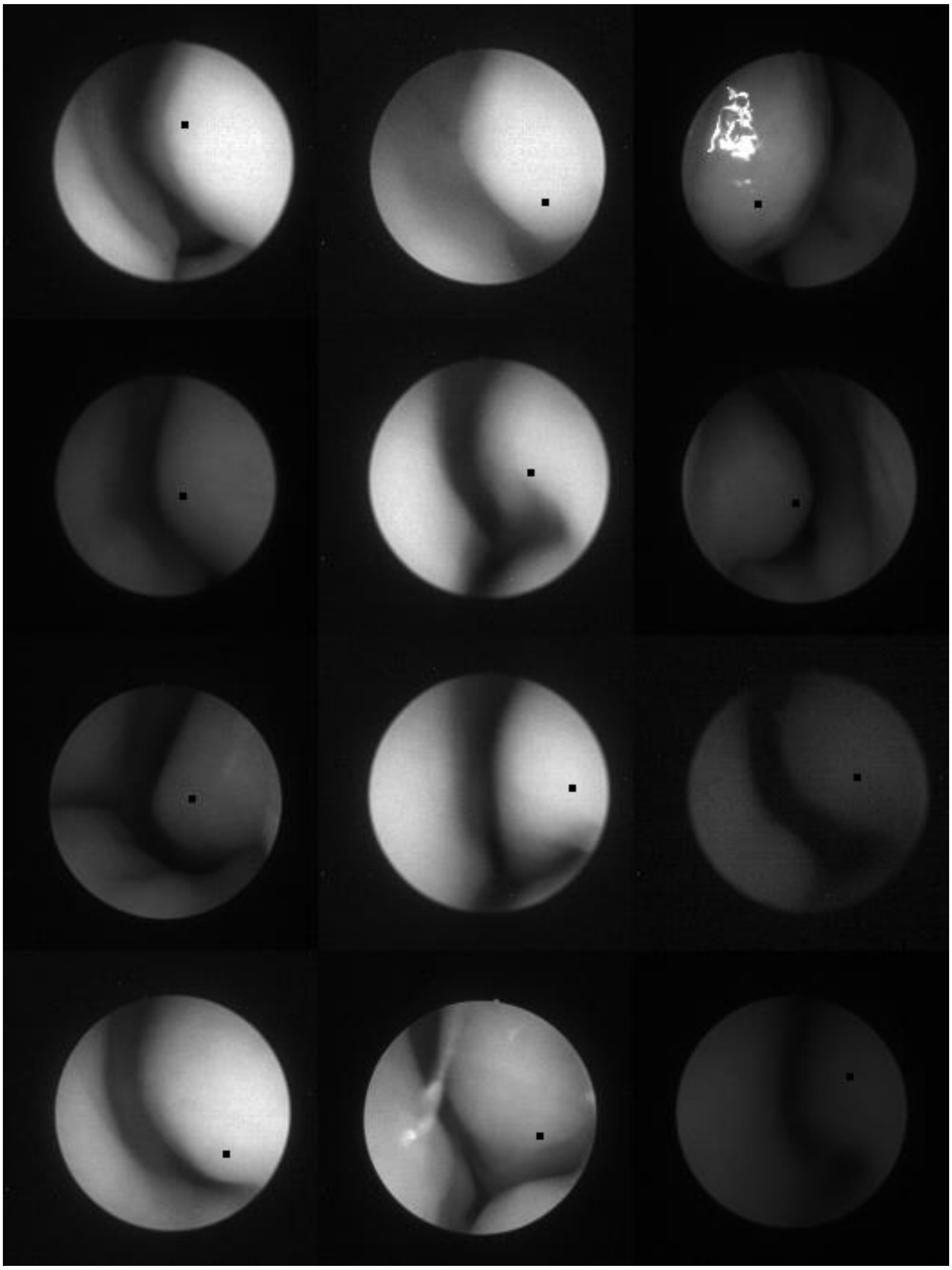


Figure 23. The Default images that were selected from the Auto algorithm for each subject. The Frame selection algorithm selects the MS cube frame from the reference video, here it is represented the 746 nm wavelength. The black dots represent the area of the inf

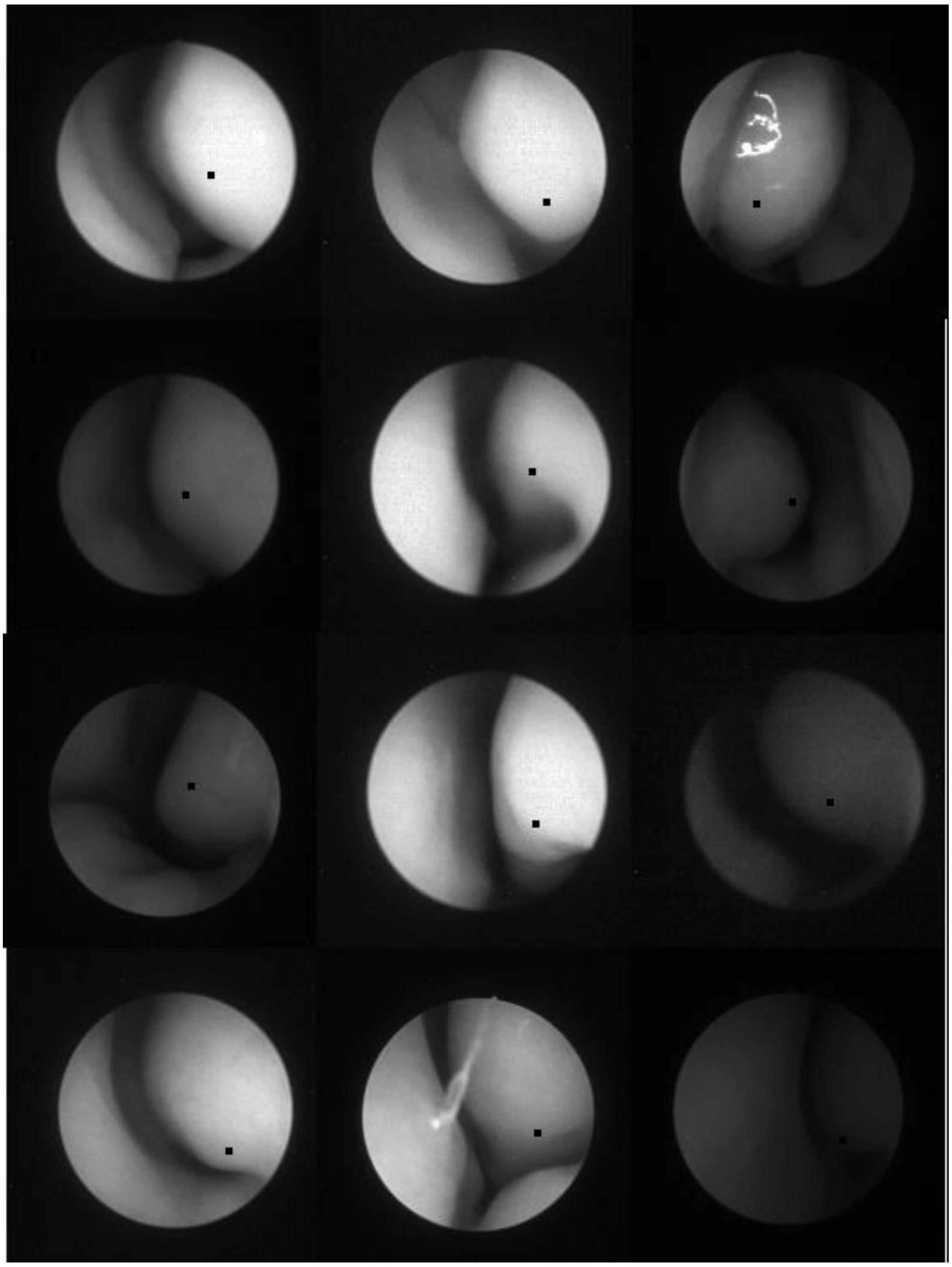


Figure 24. The Default images that were selected from the Manual algorithm for each subject. The Frame selection algorithm selects the MS cube frame from the reference video, here it is represented the 746 nm wavelength. The black dots represent the area of the inferior turbinate that the user selected.

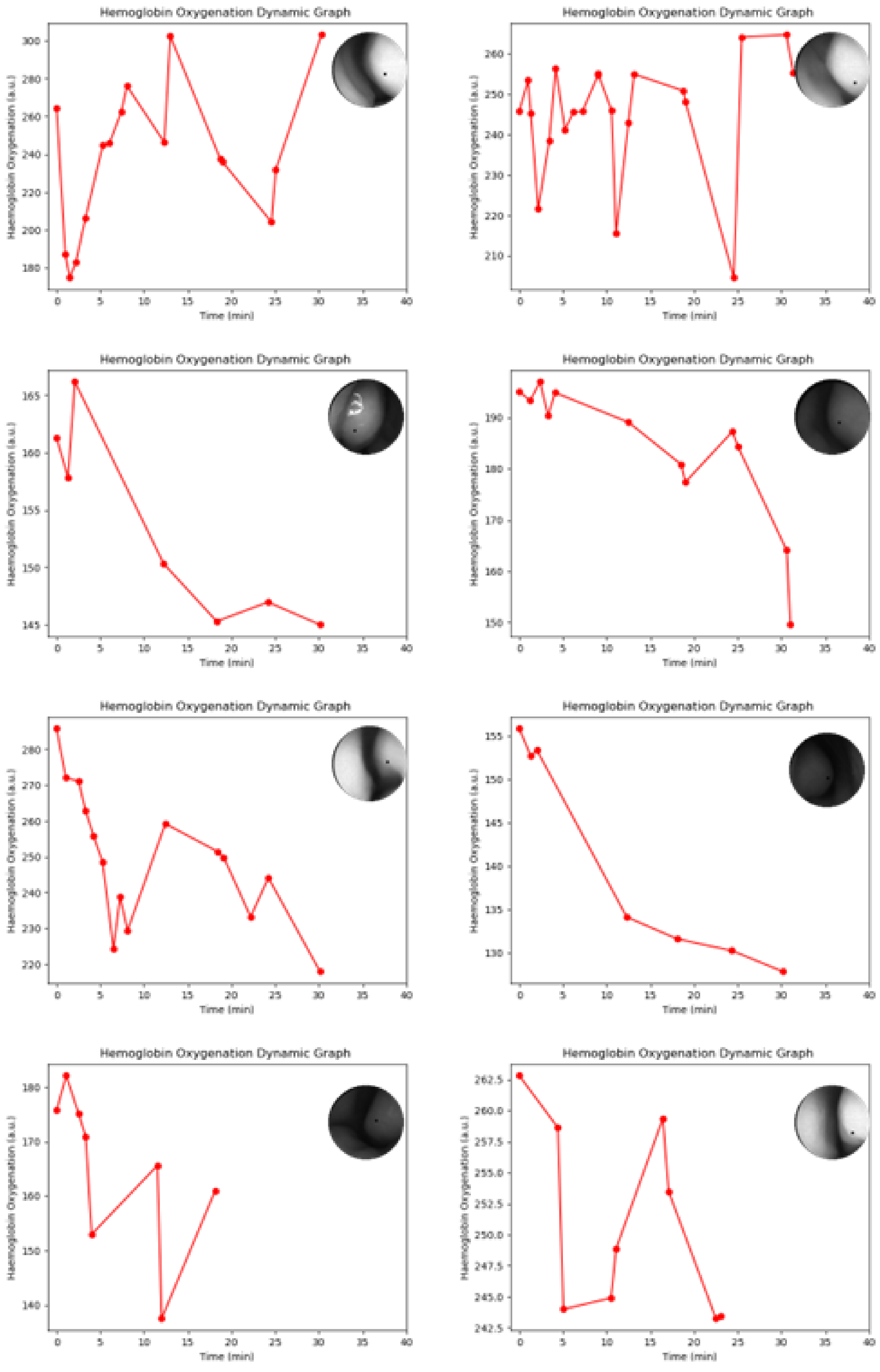


Figure 25. The results of the Manual method of 8 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD).

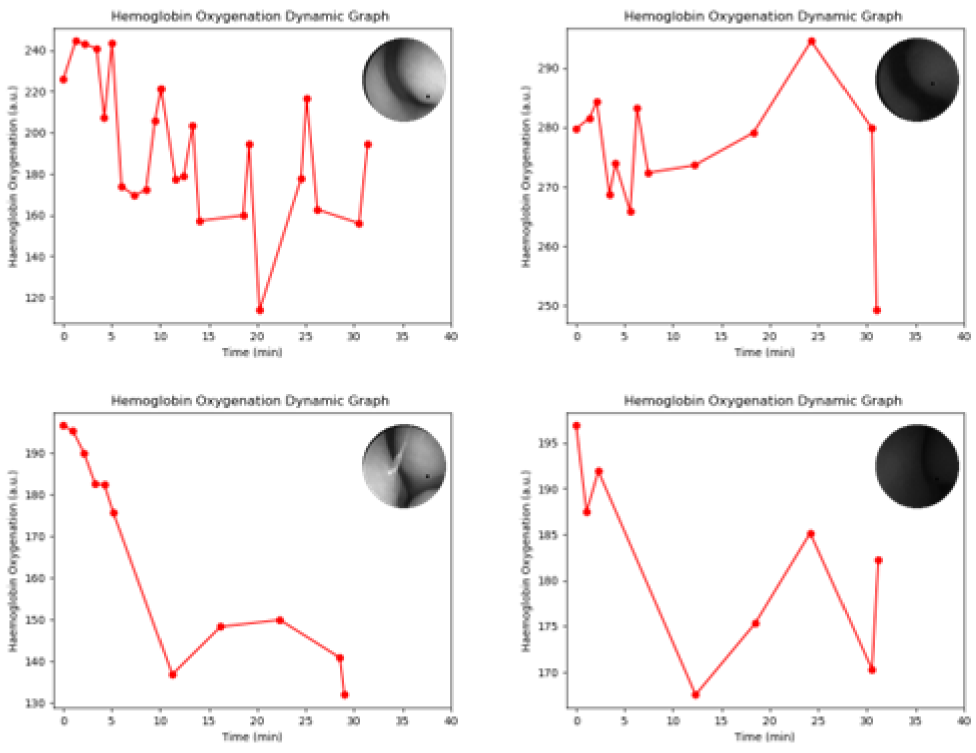


Figure 26. The results of the Manual method of 4 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD).

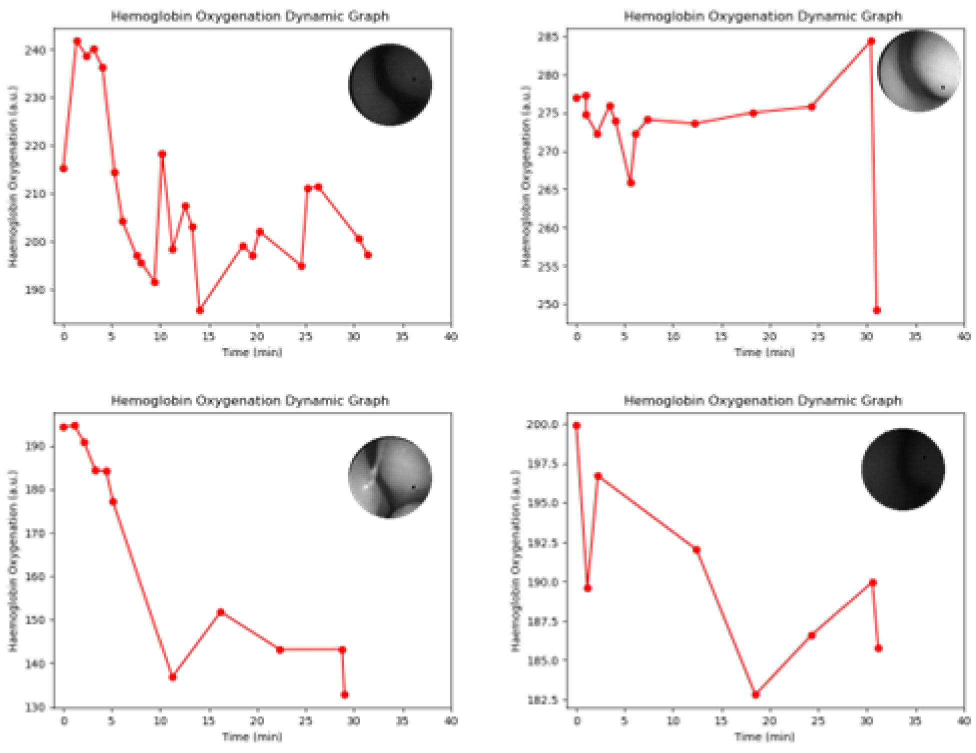


Figure 27. The results of the Auto method of 4 subjects represented in each Hemoglobin Oxygenation Dynamic Graph (HOD).

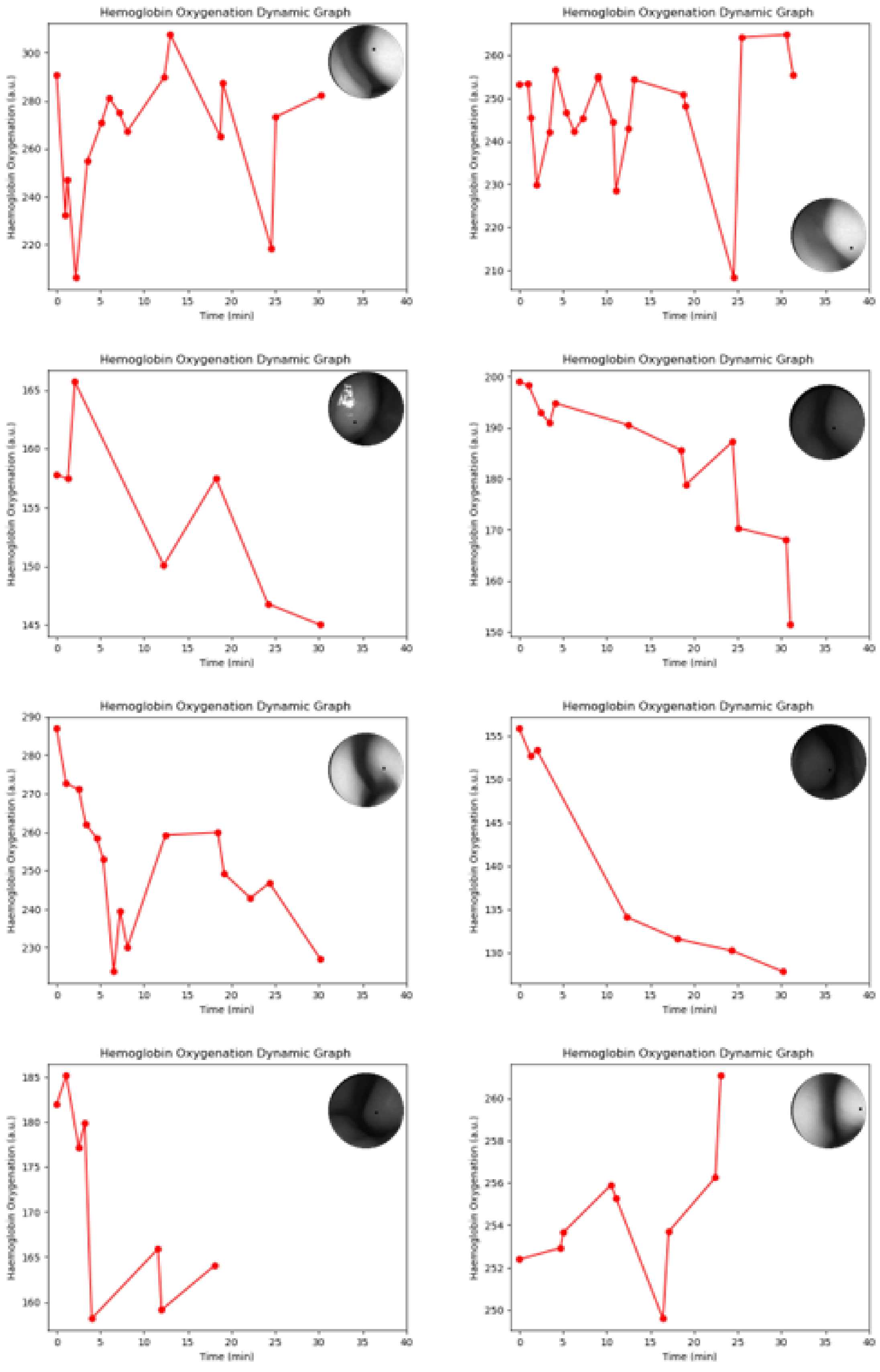


Figure 28. The results of the Auto method of 8 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD).

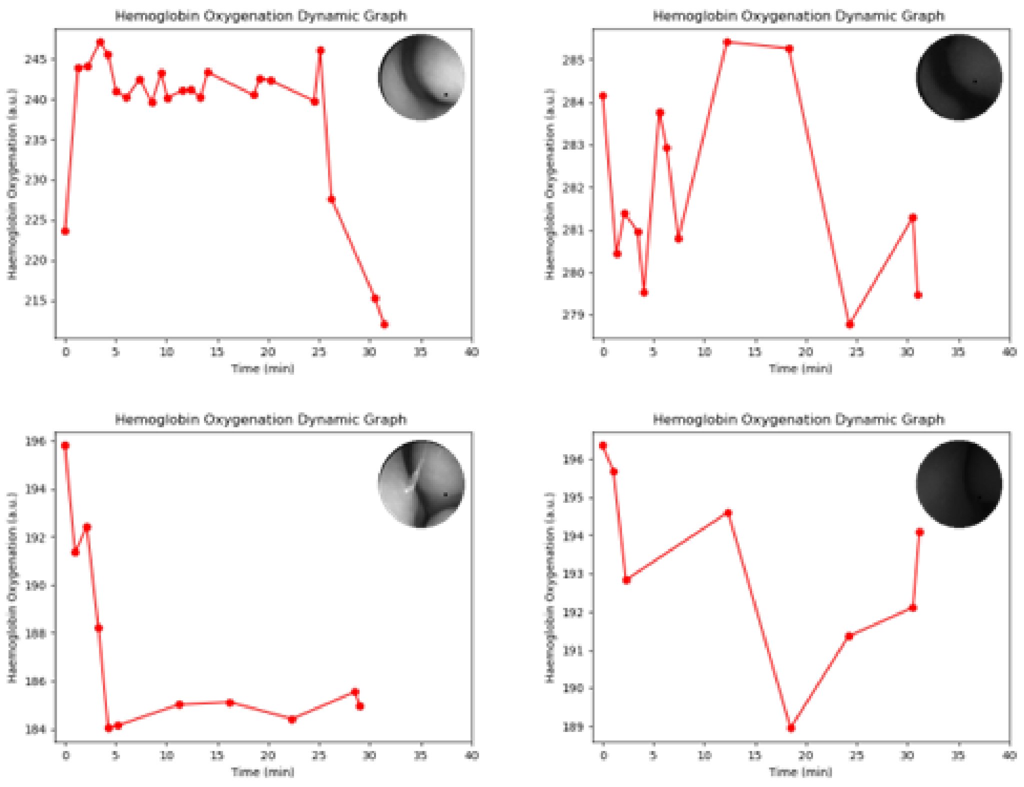


Figure 29. The results of the Manual method of 4 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD) using multiple inferior turbinate areas as input.

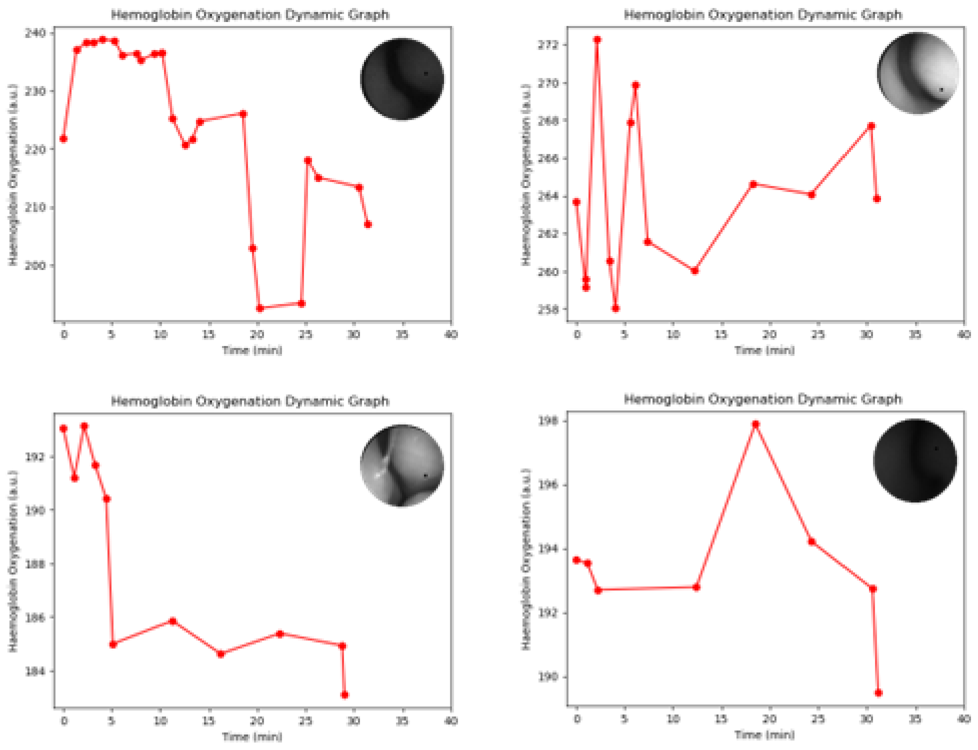


Figure 30. The results of the Auto method of 4 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD) using multiple inferior turbinate areas as input.

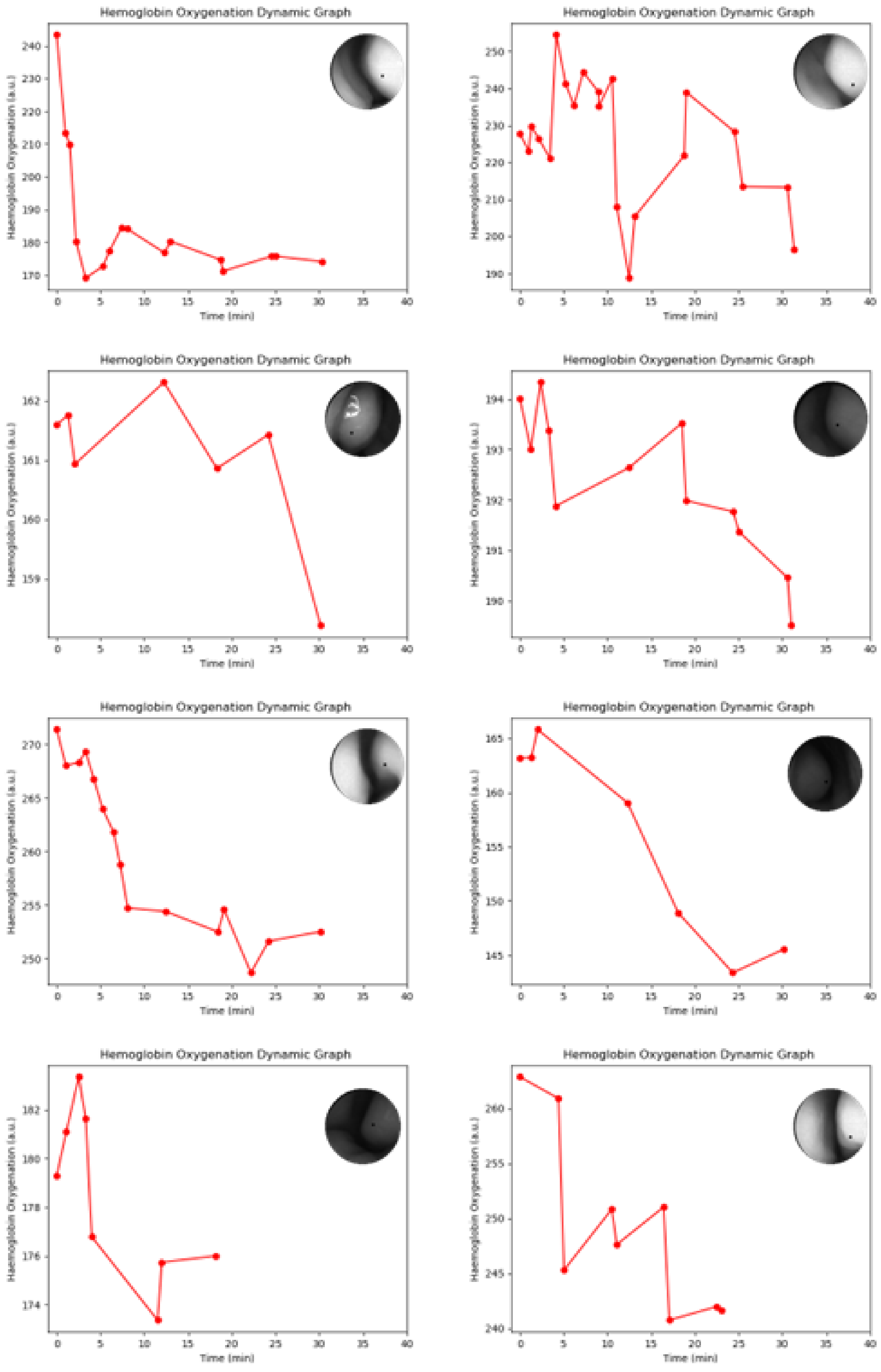


Figure 31. The results of the Manual method of 8 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD) using multiple inferior turbinate areas as input.

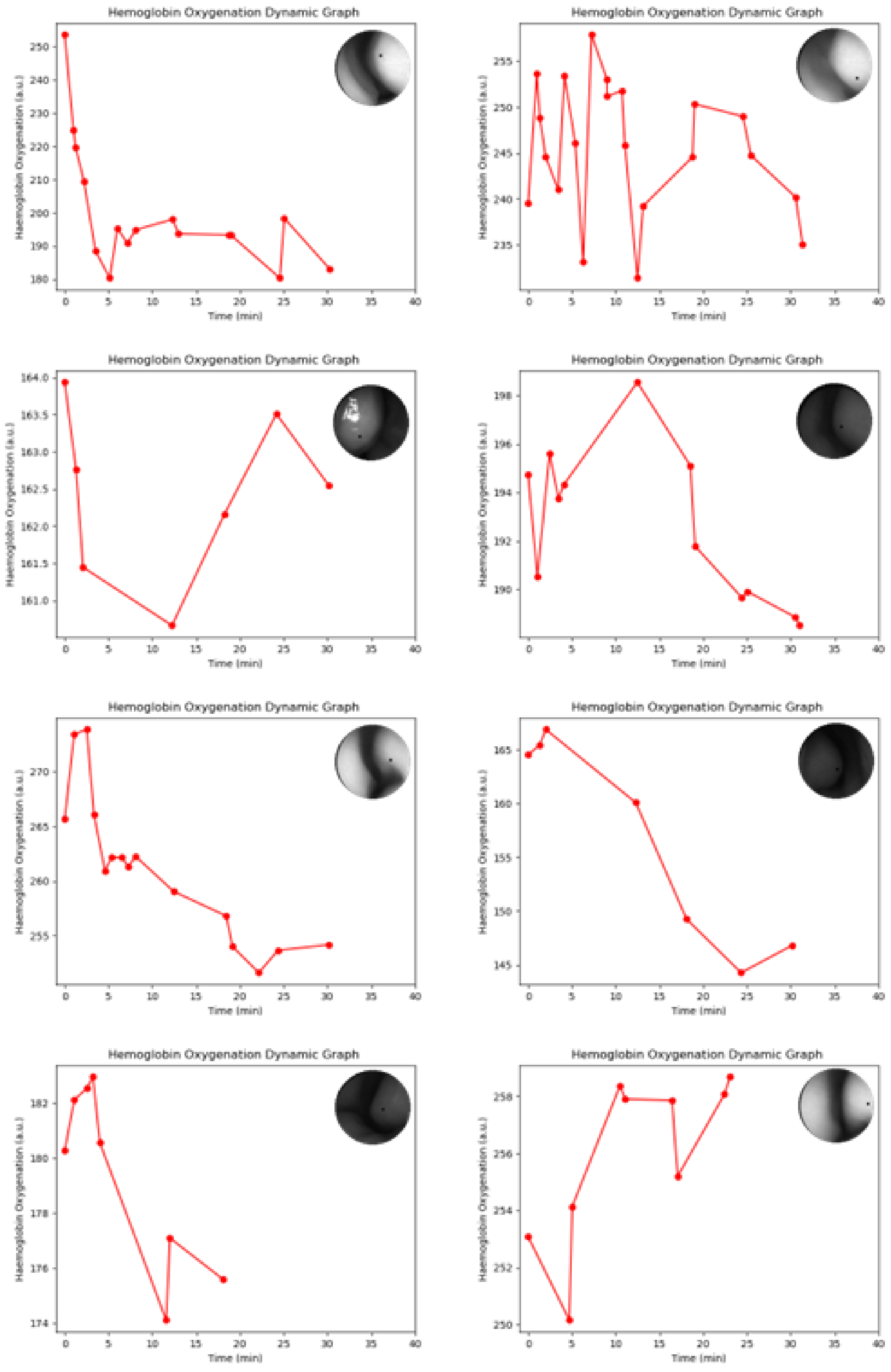


Figure 32. The results of the Auto method of 8 subjects for the Hemoglobin Oxygenation Dynamic Graph (HOD) using multiple inferior turbinate areas as input.

In the above figures (figures 25-32) for each method (manual and auto) the subjects are presented in the exact same order. This can also be assumed through the default images inside the graphs on the top right corner. After examining the results of the 12 subjects of our dataset in each algorithm, it can be clearly observed that the results are following a specific pattern, showing a pattern that the blood oxygenation is dropping through time. In 8 of the 12 subjects the dropping pattern is certain and very similar. Although they are preliminary data an innovative model can be generated by averaging the HOD of these 8 subjects with the common pattern. The result of the integration of these graphs can be seen in figure 33. In the other 4 subjects that this dropping pattern could not be seen, could be due to multiple factors, such as personalized issues like allergies or even measurement noise. For a more representative model more subjects (over 20) need to be considered to ensure robustness. For the creation of the model in figure 33 the values of each subject were normalized by dividing each value with the average of all the values of each subject. After that the standard error (iv) was calculated of all the normalized values of each subject per minute. Finally, the data was fitted to an exponential curve decay (v).

$$standard\ error = \frac{\sigma}{\sqrt{n}} \text{ (iv)}$$

Where σ is the standard deviation and n is the number of samples.

$$exponential\ curve\ decay = a * e^{-k*x} + b \text{ (v)}$$

Where: parameter a=0.14938655326827058, parameter k=0.16429989637393821, parameter b=0.9521969592780284 and x which is the input takes values from 0 to 30 in our case.

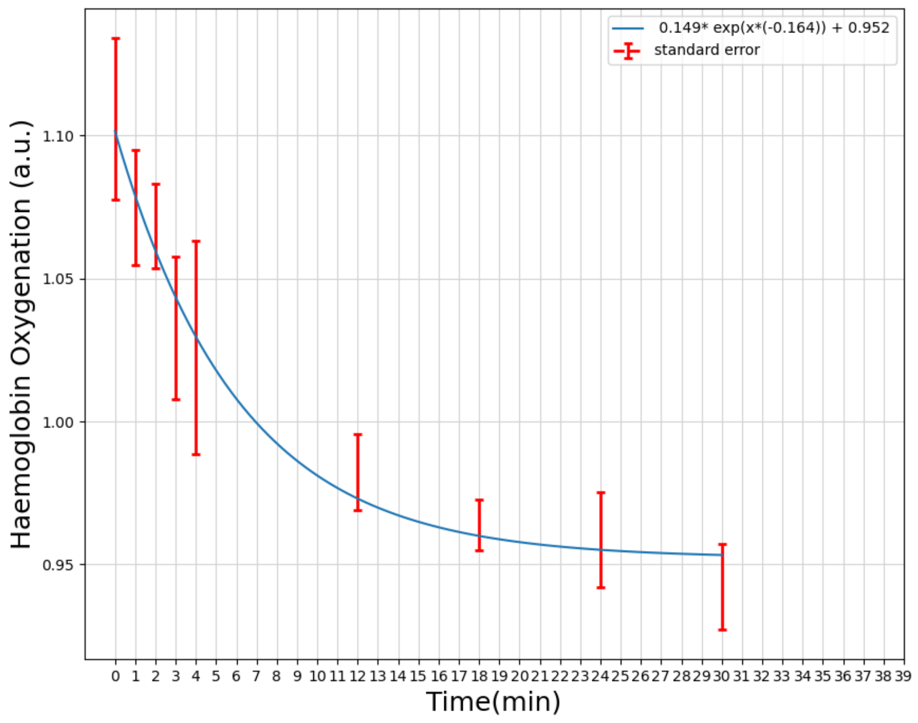


Figure 33. A HOD graph that represents the integration of 8 subjects that follow the dropping pattern.

It seems that this HOD graph (figure 33) is corresponding with the pharmaceutical phenomenon of the application of oxymethazoline at the nasal mucosa. The result of this application is the vasoconstriction of the mucosa, and it is clinically expressed with the better breathing of each individual in a matter of minutes.

7. Discussion and Future Work

An important note that was experienced during this study is that for this technique to work properly, the doctor needs to have the necessary experience to collect data with minimal movements. If the doctor isn’t experienced enough, he could be changing the angle of view and depth all the time leading to data that are difficult to process. If the frames don’t have a flow, meaning that the objects inside them change dramatically their position, the position of the inferior nasal turbinate area might not be in place. The coordinates that are used are the same for every frame selected from the frame selection, meaning that if the objects aren’t aligned the inferior turbinate area might be pointing to a black area or an area that wasn’t supposed to. This is why the algorithm is developed to extract the closest related frames from the videos, resulting in the most accurate measurement of the nasal mucosa.

It is not desirable to make the doctors select each area in each frame selected, it is already tiring to make the doctor select the areas as it is. For a future work a machine learning algorithm could be used to identify using pattern recognition the nasal inferior turbinate area. The areas obtained for this research as well as the results could be used as a training set for that algorithm.

Moreover, something that could impact significantly this research is to create a device that could acquire the data instead of a doctor. Removing the one of the two variants that create movement could lead to the minimization of the movement. This device could either be a helmet that the subject could wear and adjust the camera’s optical fiber in his or her nose or something like a camera stand where the camera would remain still throughout the data acquisition.

The graphical user interface (GUI) that was created for this work is designed so that a doctor or a user can make the data processing and analysis easily. The user can create projects to save the data and re-run them whenever he or she wants and of course he or she would not have to write or manipulate code in order to run data. The program will automatically do everything, as long as the user has read the user manual and knows exactly what needs to be done.

The MS frame selection is taking around 5 minutes for every dataset of one subject. However, the selection would take much more time since it would use the MS frame selection algorithm for every frame of the Reference video. This means that it would take 5*n +5 minutes approximately, with n the number of MS frames of the first video. In each device the 5 minutes would of course variate. The more the frames of the first video the longer it will take to finish, but it will be for the best since there are more chances of finding a frame that would find the most common field of view across all frames of all videos. As a result, the processing for the Auto algorithm is going to take more time than the Manual method, but it is more accurate since it finds the most accurate image for the MS frame selection. That image is selected by comparing all the frames from the Reference video with all of the video from the rest of our dataset. Another reason that could affect the running time is the frames per second, obviously higher fps means more frames in each video even if the minutes of the process are the same, which would result in more time. The program will automatically acknowledge the fps from the file properties.

In addition, for future work, it would be important to also try a different group of subjects, with some clinical pathologies, and to test also different pharmaceutical agents on the same subjects.

The information given by this study is fundamental in order to optimize the current clinical diagnosis of clinicopathologic conditions, since its an innovative methodology for capturing the pharmacokinetics of any substance that can theoretically have an effect on the nasal mucosa.

8 Conclusion

Multispectral imaging was tested as a diagnostic method to test the effect of pharmaceutical agents on nasal mucosa.

The quality of the measurements is strongly dependent on the doctor’s experience due to the multiple moving degrees of freedom (breathing, subject movements, doctor movements).

Measurements involved 12 individual subjects with no known pathological condition.

An analytical platform was developed to assist on the analysis of the spectral data. The user manual is presented in the appendix, where there are all the steps are explained in detail so that the user can easily operate the platform.

The best wavelengths were identified showing the highest dynamic variability during measurement. The oxygenation of hemoglobin is calculated by dividing the two wavelength bands, 642 nm and 870 nm.

A gradual dropping dynamic pattern was identified, most probably due to the limited blood supply on the nasal conchae of the inferior turbinate and the model that was created by integrating the 8 out of 12 subjects can be seen in figure 33.

More individual subjects are necessary to achieve better statistics and to use machine learning approaches.

Arguably, the 12 subjects are not a big dataset, but it can be clearly understood that the results are following a specific pattern, showing a pattern that the blood oxygenation is dropping through time (figure 33). Certainly, when the number of subjects will be larger, results will be improved.

In this study the tool that was created for the nasal mucosa blood oxygenation could be used for the diagnosis of clinicopathologic conditions or it could optimize the current clinical diagnosis of clinicopathologic conditions and could be useful in the evaluation of the different pharmaceutical agents’ efficiency.

This HOD graph (Figure 33) appears to match to the pharmacological response of applying oxymethazoline to the nasal mucosa. The effect of this application is mucosal vasoconstriction, which is clinically evidenced by each individual's best breathing within a few minutes.

9 References

[1] E. P. Prokopakis, C. J. Balas, P. N. Christodoulou, N. C. Gourtsoyiannis, A. D. Tosca, and E. S. Helidonis, “Assessment of nasal mucosa blood supply by quantitative endoscopic imaging of back-scattered light: International Editor,” *Otolaryngology—Head and Neck Surgery*, vol. 121, no. 3, pp. 307–312, 1999.

[2] H. M. Blom, T. Godthelp, W. J. Fokkens, A. KleinJan, P. G. Mulder, and E. Rijntjes, “The effect of nasal steroid aqueous spray on nasal complaint scores and cellular infiltrates in the nasal mucosa of patients with nonallergic, noninfectious perennial rhinitis,” *Journal of allergy and clinical immunology*, vol. 100, no. 6, pp. 739–747, 1997.

[3] C. Bitter, K. Suter-Zimmermann, and C. Surber, “Nasal drug delivery in humans,” *Topical Applications and the Mucosa*, vol. 40, pp. 20–35, 2011.

[4] S. Ortega Sarmiento, “Automatic classification of histological hyperspectral images: algorithms and instrumentation,” 2021.

[5] B. Georgakopoulos and P. H. Le, “Anatomy, Head and Neck, Nose Interior Nasal Concha,” 2019.

[6] R. Kahana-Zweig, M. Geva-Sagiv, A. Weissbrod, L. Secundo, N. Soroker, and N. Sobel, “Measuring and characterizing the human nasal cycle,” *PLoS One*, vol. 11, no. 10, p. e0162918, 2016.

[7] G. Berger, M. Balum-Azim, and D. Ophir, “The normal inferior turbinate: histomorphometric analysis and clinical implications,” *The Laryngoscope*, vol. 113, no. 7, pp. 1192–1198, 2003.

[8] D. M. Dalgorf and R. J. Harvey, “Sinonasal anatomy and function,” *American journal of rhinology & allergy*, vol. 27, no. 3_suppl, pp. S3–S6, 2013.

[9] F. de Rezende Pinna, B. Ctenas, R. Weber, P. H. Saldiva, and R. L. Voegels, “Olfactory neuroepithelium in the superior and middle turbinates: which is the optimal biopsy site?,” *International archives of otorhinolaryngology*, vol. 17, no. 02, pp. 131–138, 2013.

[10] D. H. Smith, C. D. Brook, S. Virani, and M. P. Platt, “The inferior turbinate: an autonomic organ,” *American journal of otolaryngology*, vol. 39, no. 6, pp. 771–775, 2018.

[11] K. Hadley, R. R. Orlandi, and K. J. Fong, “Basic anatomy and physiology of olfaction and taste,” *Otolaryngologic Clinics of North America*, vol. 37, no. 6, pp. 1115–1126, 2004.

[12] M. Orhan, R. Midilli, S. Gode, C. Y. Saylam, and B. Karci, “Blood supply of the inferior turbinate and its clinical applications,” *Clinical Anatomy*, vol. 23, no. 7, pp. 770–776, 2010.

[13] T. Gotlib, M. Kumiska, J. Sokolowski, T. Dziedzic, and K. Niemczyk, “The supreme turbinate and the drainage of the posterior ethmoids: a computed tomographic study,” *Folia morphologica*, vol. 77, no. 1, pp. 110–115, 2018.

[14] K. D. Smith, P. C. Edwards, T. S. Saini, and N. S. Norton, “The prevalence of concha bullosa and nasal septal deviation and their relationship to maxillary sinusitis by volumetric tomography,” *International journal of dentistry*, vol. 2010, 2010.

[15] M. C. Rusu, M. Săndulescu, C. J. Sava, and D. Dincă, “Bifid and secondary superior nasal turbinates,” *Folia morphologica*, vol. 78, no. 1, pp. 199–203, 2019.

[16] C. Neuland, T. Bitter, H. Marschner, H. Gudziol, and O. Guntinas-Lichius, “Health-related and specific olfaction-related quality of life in patients with chronic functional anosmia or severe hyposmia,” *The Laryngoscope*, vol. 121, no. 4, pp. 867–872, 2011.

[17] J. S. Rhee, D. T. Book, M. Burzynski, and T. L. Smith, “Quality of life assessment in nasal airway obstruction,” *The Laryngoscope*, vol. 113, no. 7, pp. 1118–1122, 2003.

[18] C.-C. Huang, P.-W. Wu, C.-H. Fu, C.-C. Huang, P.-H. Chang, C.-L. Wu, and T.-J. Lee, “What drives depression in empty nose syndrome? A Sinonasal Outcome Test-25 subdomain analysis,” *Rhinology*, vol. 57, no. 6, pp. 469–476, 2019.

[19] L. R. A. Schall, “II. The Histology and Chronic Inflammation of the Nasal Mucous Membrane,” *Annals of Otology, Rhinology & Laryngology*, vol. 42, no. 1, pp. 15–38, 1933.

[20] B. Rusak and I. Zucker, “Biological rhythms and animal behavior,” *Annual review of psychology*, vol. 26, no. 1, pp. 137–171, 1975.

[21] R. Kayser, “Die exacte messung der luftdurchgangigkeit der nase,” *Arch Laryngol*, vol. 94, pp. 149–156, 1895.

[22] F. Bojsen-Moller and J. Fahrenkrug, “Nasal swell-bodies and cyclic changes in the air passage of the rat and rabbit nose.,” *Journal of anatomy*, vol. 110, no. Pt 1, p. 25, 1971.

[23] D. F. Proctor and I. H. P. Andersen, *The nose: upper airway physiology and the atmospheric environment*. Elsevier Biomedical Press Amsterdam, 1982.

[24] J. Keuning, “On the nasal cycle.,” *Int Rhinol*, vol. 6, pp. 99–136, 1968.

[25] D. Shannahoff-Khalsa, “Lateralized rhythms of the central and autonomic nervous systems,” *International Journal of Psychophysiology*, vol. 11, no. 3, pp. 225–251, 1991.

[26] J. Ishii, T. Ishii, and M. Ito, “The nasal cycle in patients with autonomic nervous disturbance,” *Acta Oto-Laryngologica*, vol. 113, no. sup506, pp. 51–56, 1993.

[27] M. Preece and R. Eccles, “The effect of pressure and warmth applied to the axilla on unilateral nasal airway resistance and facial skin temperature,” *Acta oto-laryngologica*, vol. 113, no. 6, pp. 777–781, 1993.

[28] L. Knight, “Temporal changes in unilateral middle ear pressure under basal conditions,” *Clinical Otolaryngology & Allied Sciences*, vol. 16, no. 6, pp. 543–546, 1991.

[29] D. E. White, J. Bartley, and R. J. Nates, “Model demonstrates functional purpose of the nasal cycle,” *Biomedical engineering online*, vol. 14, no. 1, pp. 1–11, 2015.

[30] M. Hasegawa and E. Kern, “The human nasal cycle.,” in *Mayo Clinic Proceedings*, 1977, vol. 52, no. 1, pp. 28–34.

[31] L. Soubeyrand, “Action of Vasomotor Drugs on the Nasal Cycle and Ciliary Function.(Rhinometric and Biological Study),” *Revue de laryngologie-otologie-rhinologie*, vol. 85, pp. 49–113, 1964.

[32] E. Szucs, L. Kaufman, and P. Clement, “Nasal resistance—a reliable assessment of nasal patency?,” *Clinical Otolaryngology & Allied Sciences*, vol. 20, no. 5, pp. 390–395, 1995.

[33] C. J. Balas, G. C. Themelis, E. P. Prokopakis, I. Orfanudaki, E. Koumantakis, and E. S. Helidonis, “In vivo detection and staging of epithelial dysplasias and malignancies based on the quantitative assessment of acetic acid–tissue interaction kinetics,” *Journal of Photochemistry and Photobiology B: Biology*, vol. 53, no. 1–3, pp. 153–157, 1999.

[34] H. R. Costantino, L. Illum, G. Brandt, P. H. Johnson, and S. C. Quay, “Intranasal delivery: physicochemical and therapeutic aspects,” *International journal of pharmaceutics*, vol. 337, no. 1–2, pp. 1–24, 2007.

[35] R. K. Chandra, M. O. Patadia, and J. Raviv, “Diagnosis of nasal airway obstruction,” *Otolaryngologic Clinics of North America*, vol. 42, no. 2, pp. 207–225, 2009.

[36] R. Tahamiler, D. T. Edizer, S. Canakcioglu, M. G. Guvenc, E. Inci, and A. Dirican, “Nasal sound analysis: a new method for evaluating nasal obstruction in allergic rhinitis,” *The Laryngoscope*, vol. 116, no. 11, pp. 2050–2054, 2006.

[37] P. E. Papadakis V. Doulaptsi M Tsagkatakis G. Karatzanis A. Velegrakis S., “Non contact, time related, objective assessment of drug effectiveness in nasal mucosa, by means of multispectral imaging.” .

10. Appendix

10.1 User's Manual

Instructions on how to use the software are presented in steps in the following text:

Step 1.

When the user executes the program a Graphical User Interface will appear as you can see in figure S1. There is a white bar in the right under the text label "enter the name of the project" and 2 buttons ("AUTO" and "select the first video") that are enabled for the user.

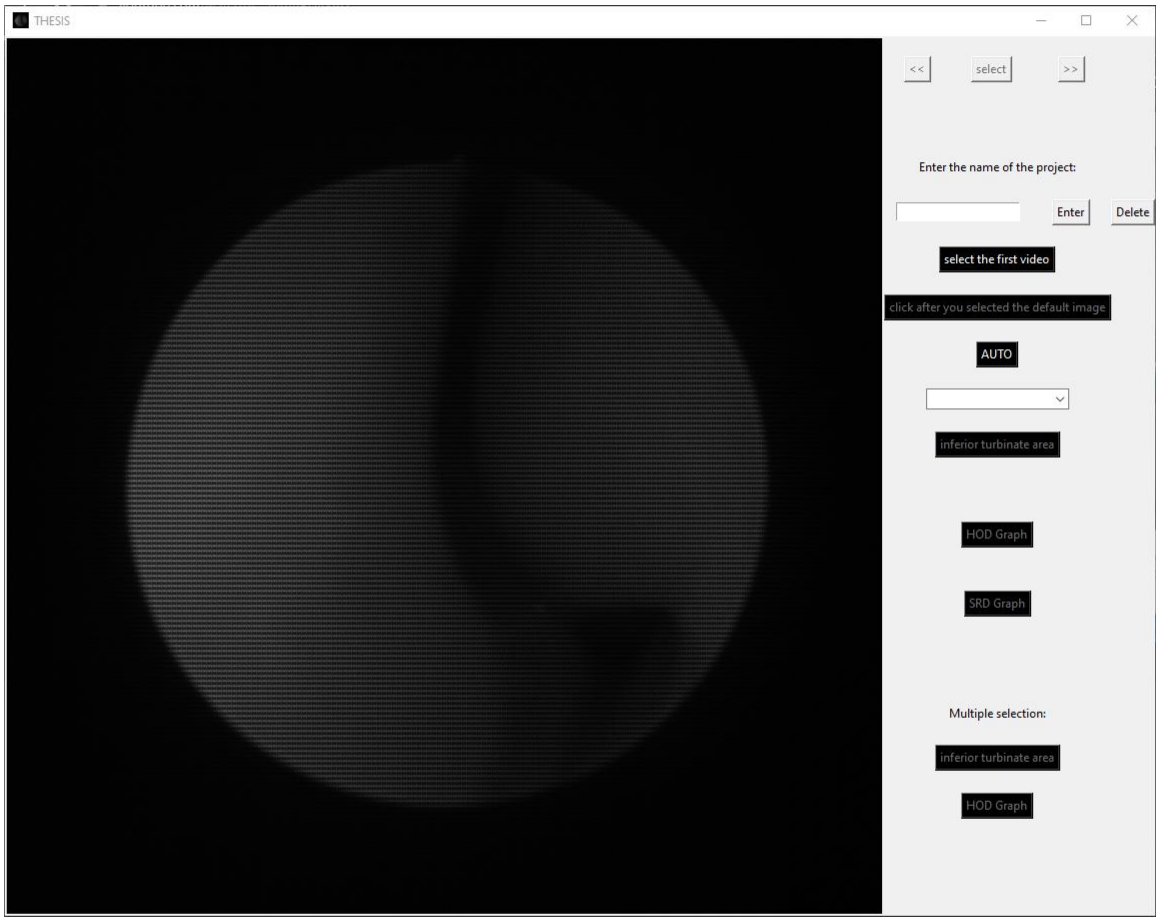


figure S 1.GUI initial

Step 2.

Initially the user will have to write in that white bar the name of the project or the name he wants to save that specific data if he wants to analyze and press "enter" (figure S2). The program will then create a new folder for this project. From that folder the analysis and the Visualization are going to take place for that project. If the user does not write anything, the data are going to be saved, but not in a specific folder and if he or she had saved data before again without a name the previous data will be overwritten. If the user wants to delete a dataset, he has to write the name of the project in that same bar and press the button “delete”. After that the user will have 2 options to choose, either the “AUTO” button or the “enter the name of the project” button.

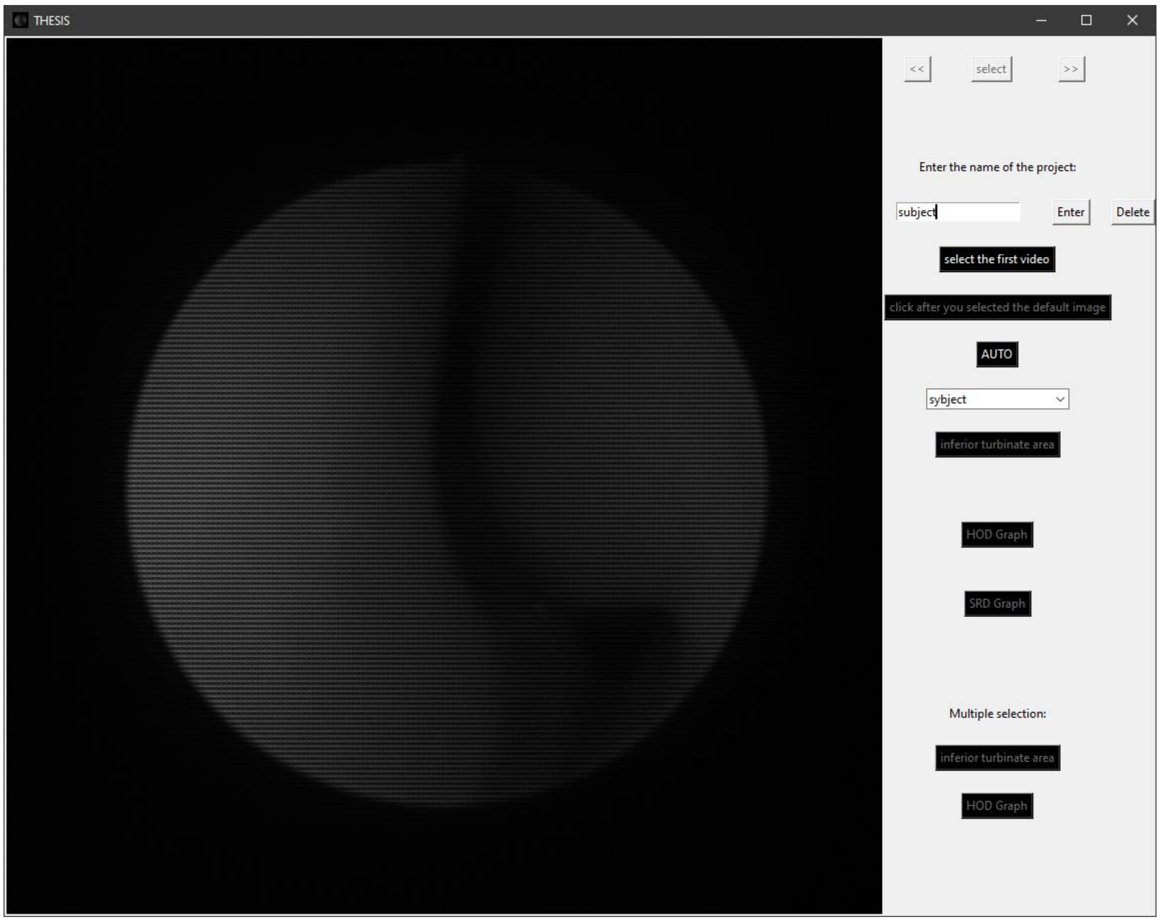


figure S 2. Naming a project.

<u>Step 3.</u>

If the user clicks the “select the first video” button, he or she will have to select the first video as input (figure S3) (the first one is the default or reference video, since the pharmaceutical agent has not yet been implemented to the subject). Then the program will read the video and process it. The area that contains the information is inside the circle, the rest of the image Is not important, because it is black (all the time). The Sensor is covering all the field of view, but the optical fiber is covering just a circle in the center, if the fiber was bigger, it would be better. Cropping the image is going to make the analysis and the processing a lot easier and faster.

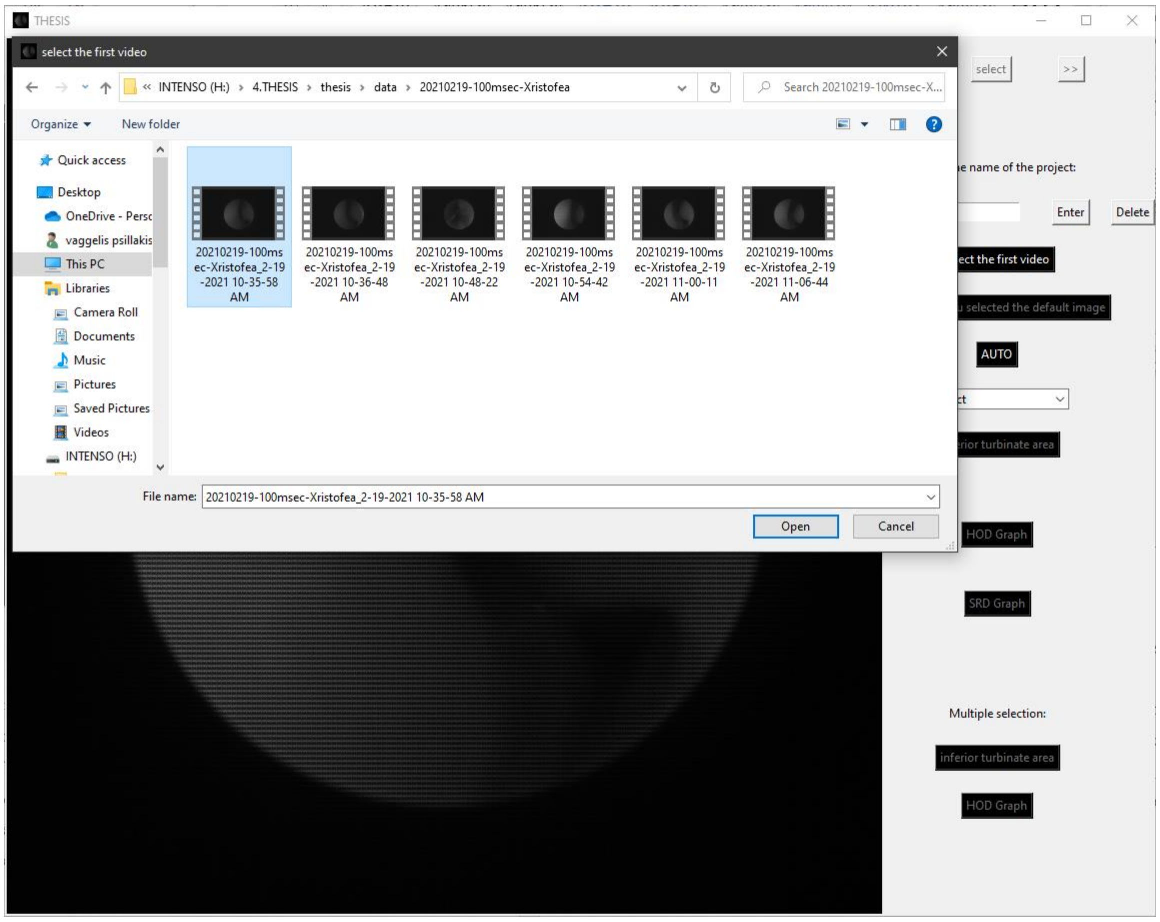

THESIS
select the first video
« INTENSO (H:) › 4.THESIS › thesis › data › 20210219-100msec-Xristofea
Search 20210219-100msec-X...
Organize
New folder
Quick access
Desktop
OneDrive - Perso
vaggelis psillakis
This PC
Libraries
Camera Roll
Documents
Music
Pictures
Saved Pictures
Videos
INTENSO (H:)
20210219-100msec-Xristofea_2-19-2021 10-35-58 AM
20210219-100msec-Xristofea_2-19-2021 10-36-48 AM
20210219-100msec-Xristofea_2-19-2021 10-48-22 AM
20210219-100msec-Xristofea_2-19-2021 10-54-42 AM
20210219-100msec-Xristofea_2-19-2021 11-00-11 AM
20210219-100msec-Xristofea_2-19-2021 11-06-44 AM
File name: 20210219-100msec-Xristofea_2-19-2021 10-35-58 AM
Open
Cancel
select
>>
e name of the project:
Enter
Delete
ect the first video
u selected the default image
AUTO
ct
erior turbinate area
HOD Graph
SRD Graph
Multiple selection:
inferior turbinate area
HOD Graph

figure S 3. Selection of the default video.

Step 4.

After the processing the image on the left will change to the first frame of our Reference video and 3 buttons (“<<”, “select”,”>>”) on the top right will be enabled as well as a bar and a new text label will be shown. The GUI will look like in the figure S4. The text label will show as the position of the frame starting from 1 till the last. Using the bar or the buttons the user can navigate through the frames in order to find the one that we think is the most fitting. This way the user will manually select the image from which the rest of the frames from the other videos are going to be selected. In other words, the user will choose the image he thinks is the most representative, so that the algorithm will select all the other frames based on that image. When the user finds that image, he or she can press the "select" button and that image is going to be saved.

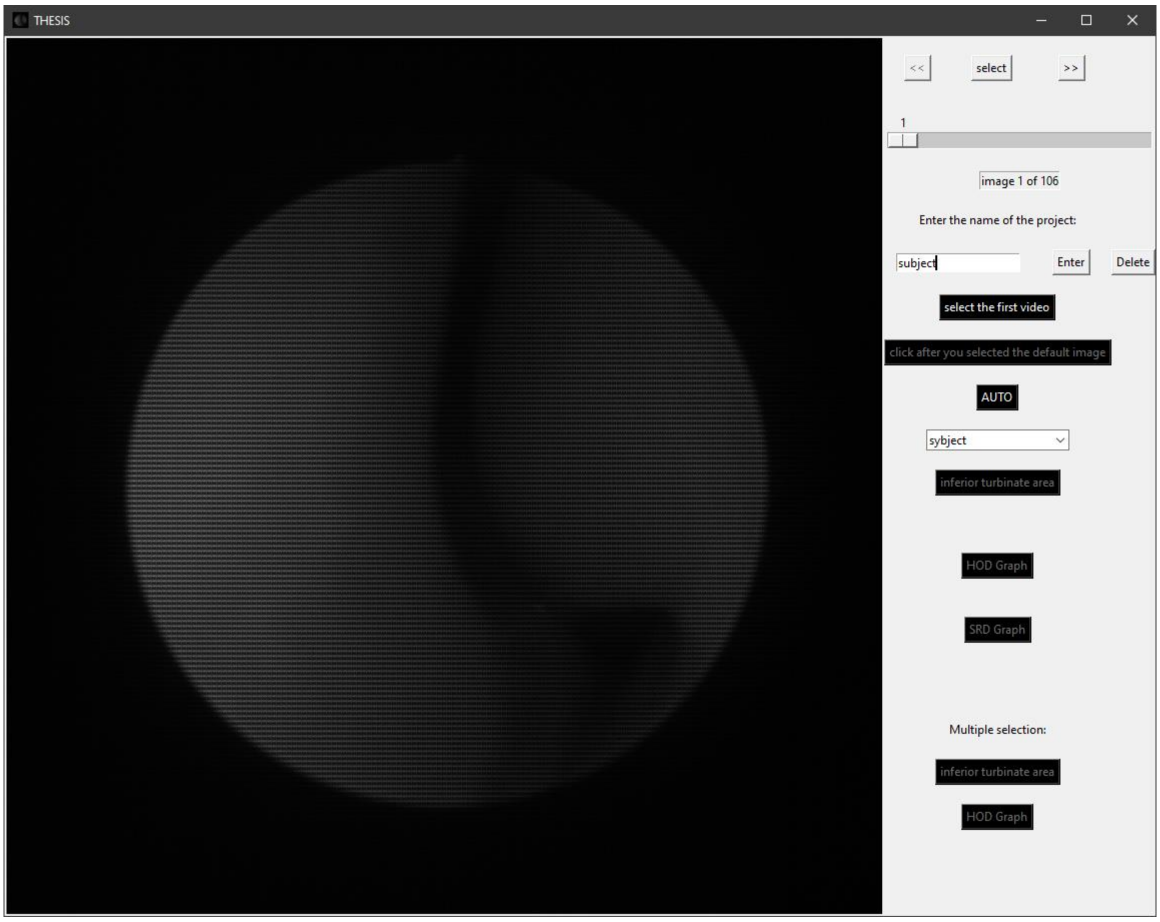


figure S 4. GUI selection of the default image.

Step 5.

Then a new button is going to be enabled, that is called "click after you selected the default image"(figure 5). When the user clicks it, he or she will have to give as an input the rest of the videos (figure S6) contained in the folder that from which he took the first video for the frame selection based on the image that the user selected earlier. Warning he should not select the initial video he used as default, because if he or she does it will cause the program to crash. If the algorithm compares two identical images with the PSNR metric the value it returns is infinite and cannot be saved on a variable. Each video will then be cropped and the name of the video is going to be sliced so that the required data are going to be used for the process. The time is essence for the analysis and visualization know exactly the time of each frame, because we know the frame per second(fps) of the videos. each frame is an image that is consisted as mentioned of a 5X5 NIR pattern: width: 2045 pixels, height 1080 pixels. we split the image in 25 images as many as the wavelengths in order to create a 3d cube. Each frame is transformed to a spectral cube (λ, x, y). Each frame has 25 images clustered in an array pattern.

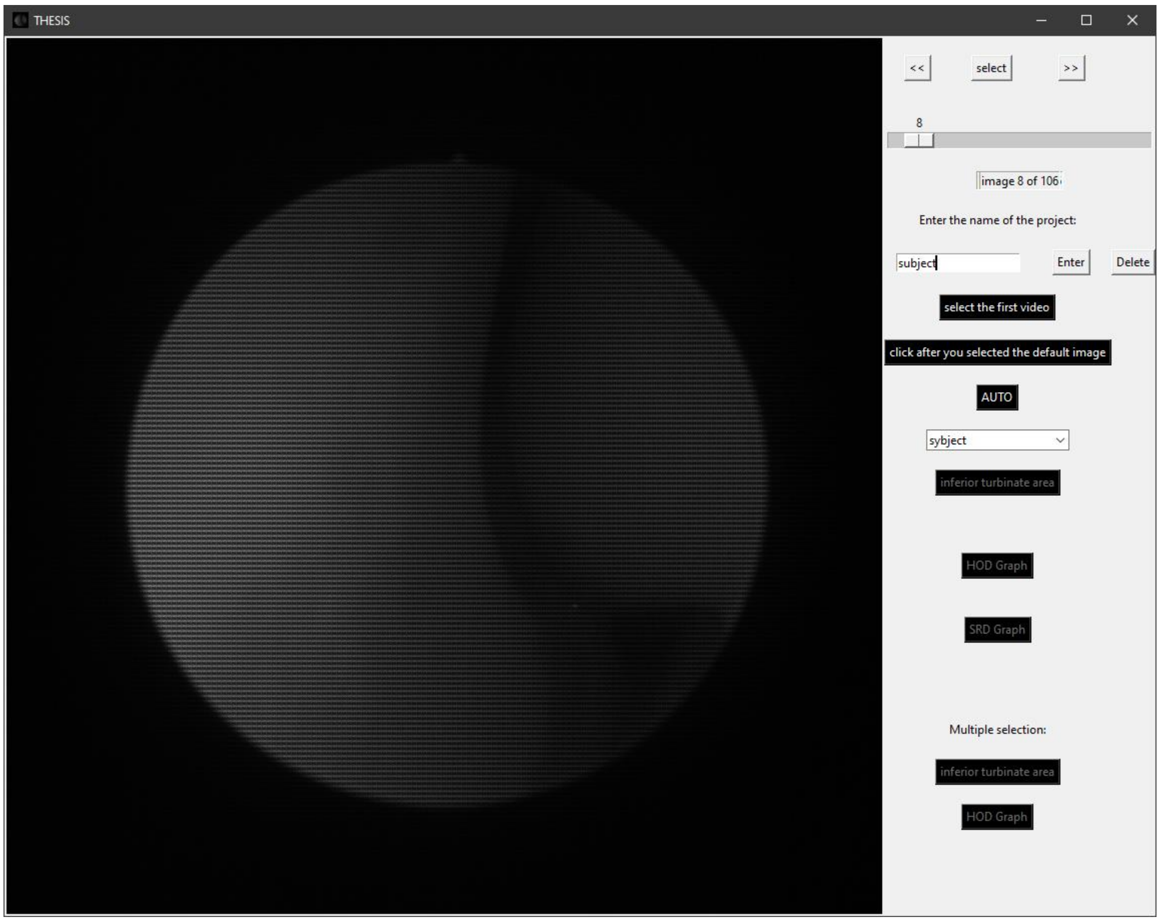
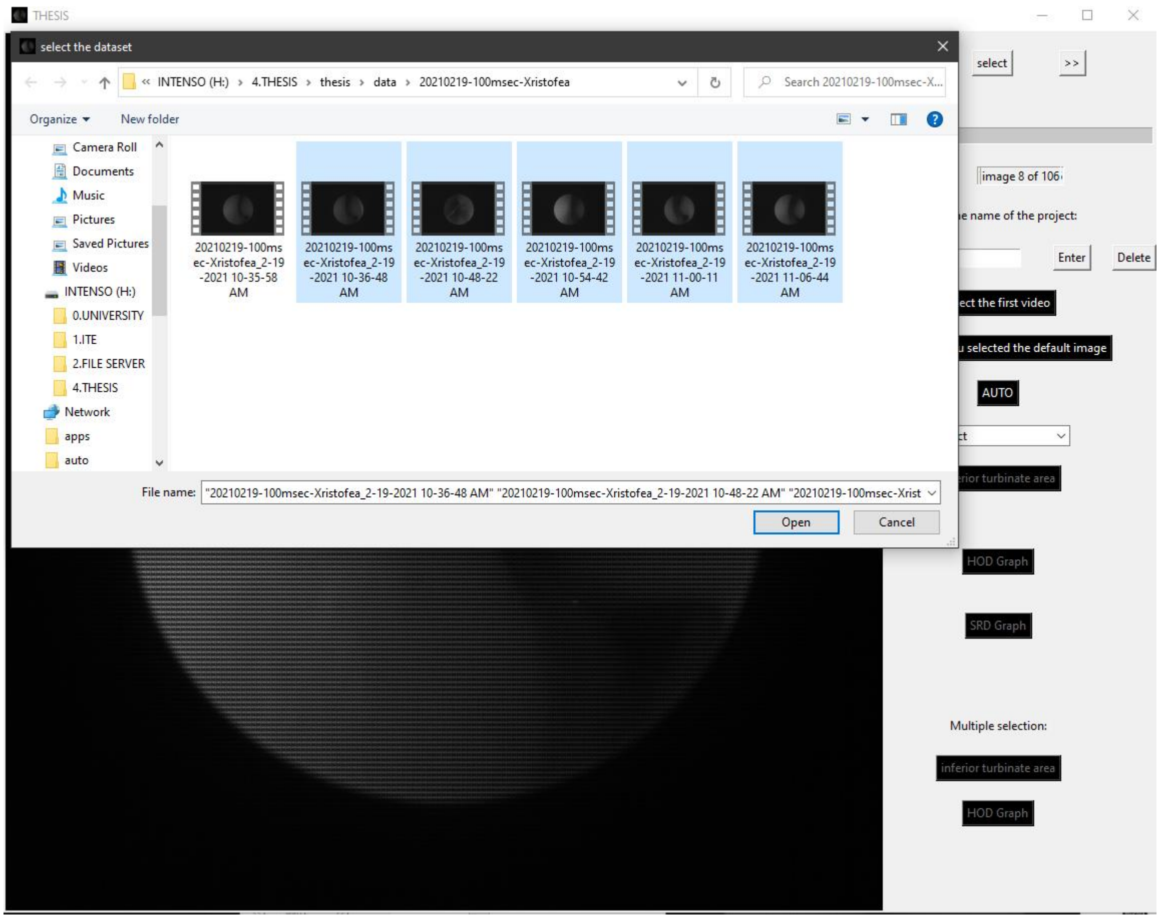


figure S 5. GUI after pressing the "select" button.

We need to read the individual images from the video frame, use them as different points in the spectral cube. From the 2MPixes (2.228.224 pixels) of each grayscale frame (8-bit), we will generate a new 3D Multispectral cube that will have (89129 pixels in spatial domain and 25 in the spectral domain). This will generate a cube of the following dimensions (λ, x, y). Then each cube (for each frame) is going to be merged to create a 4d cube which is going to be (t, λ, x, y) each cube per time. But before that we had placed the wavelengths in ascending time order. After that we are going to compare all the images from the rest of the dataset with the PSNR metric. The comparison is going to be between the wavelength 746 of every cropped image. After each frame has its PSNR score we take the best score for each minute that has passed. considering that the total time is around 11 minutes we should have 11 timepoints, one for each minute. It could be more than one video that has the same higher score, so we save those images too. The images can be seen in the folder "selected" inside the project that we are working on. The names of every image are following the following format:

hour"-"minute"-"seconds"-"µseconds"_"wavelength. For example:12-3-15-0000_683


THESIS
select the dataset
INTENSO (H:) › 4.THESIS › thesis › data › 20210219-100msec-Xristofea
Search 20210219-100msec-X...
Organize
New folder
Camera Roll
Documents
Music
Pictures
Saved Pictures
Videos
INTENSO (H:)
0.UNIVERSITY
1.ITE
2.FILE SERVER
4.THESIS
Network
apps
auto
20210219-100msec-Xristofea_2-19-2021 10-35-58 AM
20210219-100msec-Xristofea_2-19-2021 10-36-48 AM
20210219-100msec-Xristofea_2-19-2021 10-48-22 AM
20210219-100msec-Xristofea_2-19-2021 10-54-42 AM
20210219-100msec-Xristofea_2-19-2021 11-00-11 AM
20210219-100msec-Xristofea_2-19-2021 11-06-44 AM
Open
Cancel
select
>>
image 8 of 106
e name of the project:
Enter
Delete
ect the first video
u selected the default image
AUTO
rior turbinate area
HOD Graph
SRD Graph
Multiple selection:
inferior turbinate area
HOD Graph

File name: "20210219-100msec-Xristofea_2-19-2021 10-36-48 AM" "20210219-100msec-Xristofea_2-19-2021 10-48-22 AM" "20210219-100msec-Xrist

figure S 6. Selection of the rest of the dataset.

Step 6.

In case the user selects the “AUTO” button then the same processing is going to happen, but the user will not have to manually select the image that is going to be used for the frame selection, it is going to be select automatically. The user still has to select the default video (figure S3) and after that the rest of the video (figure S6), but then he or she will not have to do anything but wait. This processing is going to take more time, but it is more accurate since it finds the most accurate image for the frame selection. That image is selected by comparing all the frames from the default video will all of the video from the rest of our dataset. After the comparison the frame that has the higher score is going to be the image, which will be saved.

Step 7.

The processing part is over and now some actions are required before we move on to the visualization. Firstly, the user will have to select the name of the project that he or she wants to visualize on the bar under the button "AUTO" (figure S7).

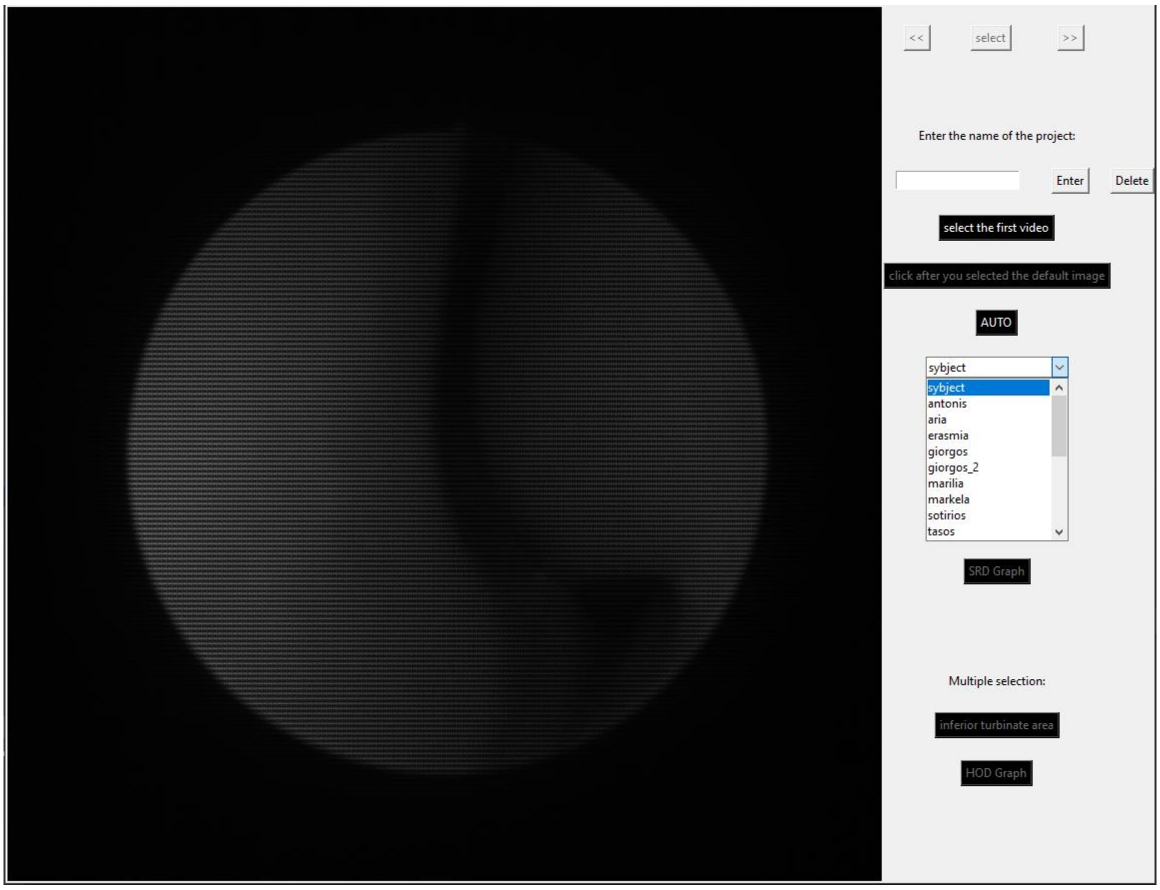


figure S 7. Selecting a project to visualize.

Step 8.

Then 3 new buttons are going to be enabled "inferior turbinate area", "HOD graph" and "SRD graph", as well as a label that says, "Multiple selection" and under it 2 buttons "inferior turbinate area", "HOD graph" (figure S8). The "inferior turbinate area" represents the area that the user will have to select of the inferior turbinate by clicking on it with the mouse. The "HOD graph" and "SRD graph", are the buttons that the user will click to visualize the result. By clicking one of the area button a new image will pop on the user's screen (figure S9), which is always the saved wavelength Reference image, and the user depending on the name of the button will have to double click on the specific area of the image so that its coordinates will be saved on a folder named "areas", each in a .txt file with the name "mucosa" as the name. the other 2 buttons under the label are for the selection of more than one inferior turbinate area to average the result.

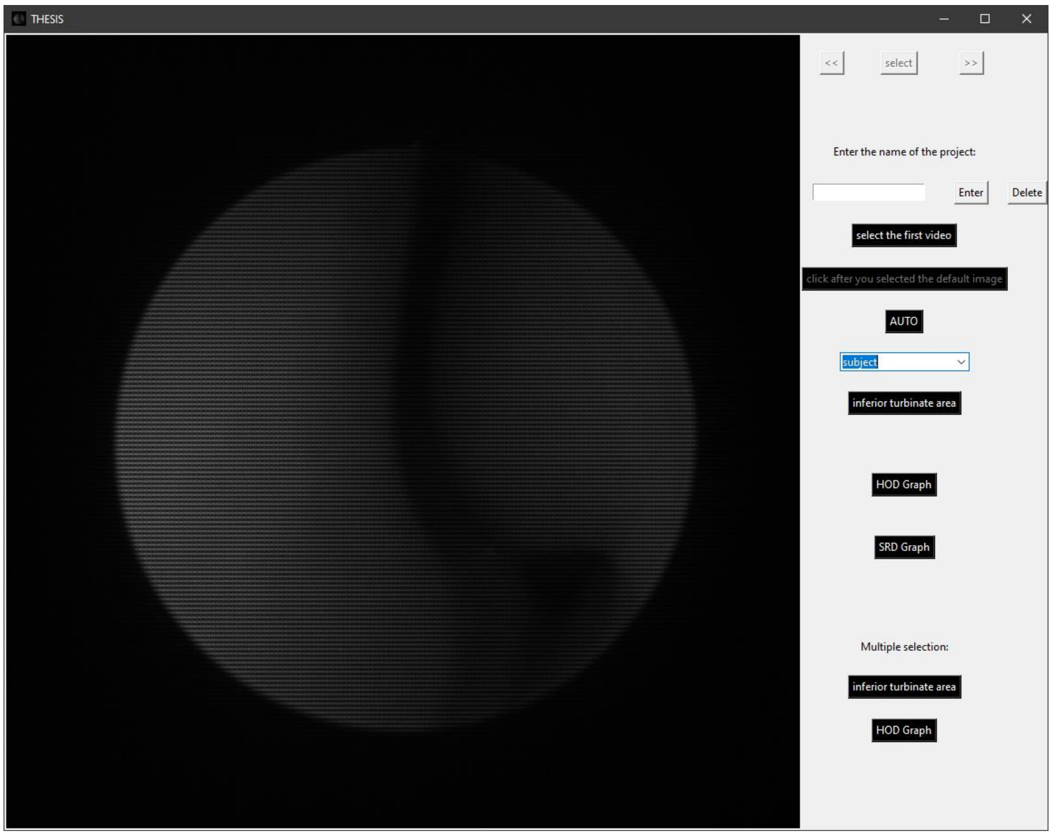


figure S 8. Visualization buttons.

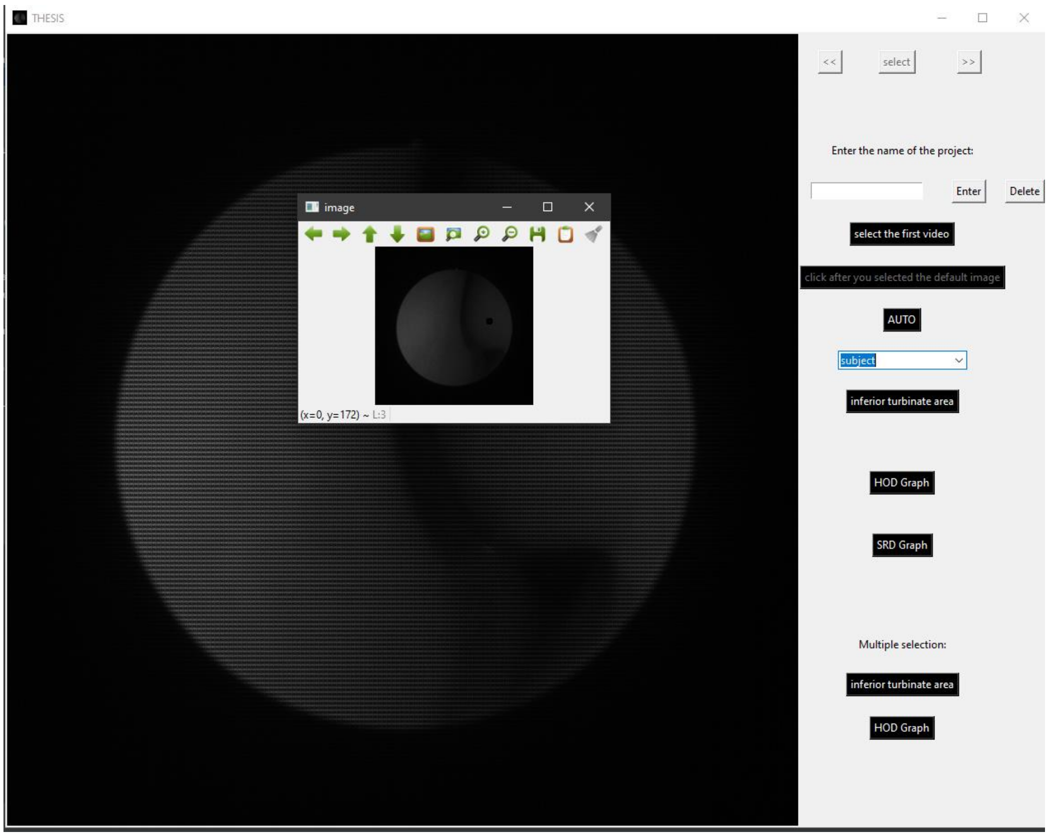


figure S 9. Area selection.

Step 9.

After the user selects all the areas he can move on to the "HOD graph" button. When the user clicks the "HOD graph" button the calculation of the intensity of hemoglobin absorption per minute(timepoint) will begin and the outcome will be the Hemoglobin Oxygenation Dynamic Graph. We do that by dividing the two distinct spectral bands (in our case 680 and 873 nm) of the average of the selected area of the inferior turbinate area the user selected before.

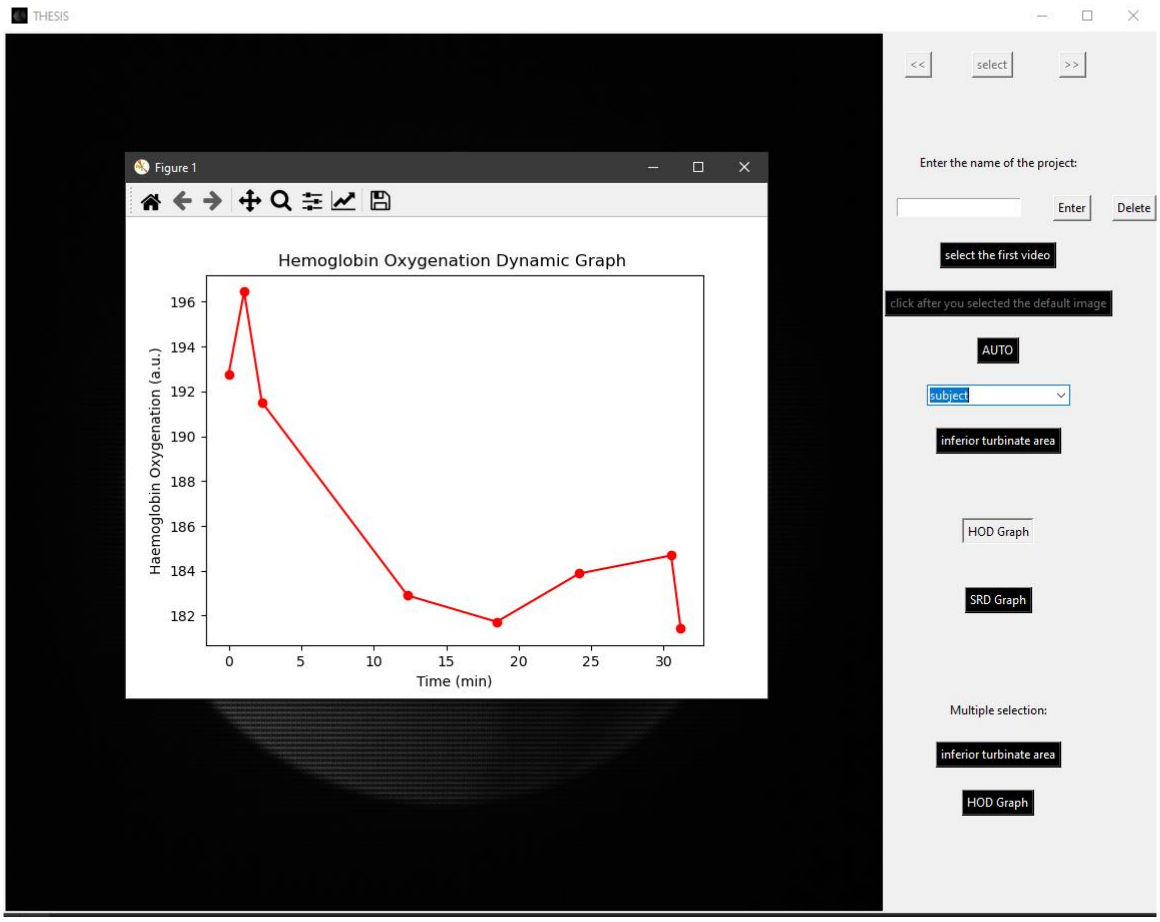


figure S 10. Hemoglobin Oxygenation Dynamic Graph. The HOD graph will automatically appear after pressing the division button.

We do that for each timepoint. The first timepoint is the reference timepoint before the application of the pharmaceutical agent. Since we want to have 1 point on our graph in the case of more than one image with the same higher P.S.N.R. score at the same minute we average the intensities that we found for each frame. After the processing a 2d Hemoglobin Oxygenation Dynamic graph will appear (figure S8) in the screen that will visualize the intensity of hemoglobin absorption per minute (timepoint).

Step 10.

The button "SRD graph" creates a 3-dimentional Spectral reflectivity dynamics (SRD) graph. By pressing the "SRD graph" button the average of the inferior turbinate area is depicted (figure S11). The values of the SRD graph are normalized by dividing each intensity point with the same intensity value that was found for the default image. In other words, the value of average of the inferior turbinate area that would have on the first minute it is going to be divided with the value of average of the inferior turbinate area that would have on the default image. For the zero time in the 3-dimensional graph we represent the default image. Since we normalize every point with the Reference value the value of the zero time is 1. From the 3D SRD graphs the user can obtain in which spectrum was the largest dynamic change in the measurement through time. In one axis there is the time with a range from 0-40 minutes, in the second one is the intensity and on the last one we have all the spectrum bands, in other words our wavelength.

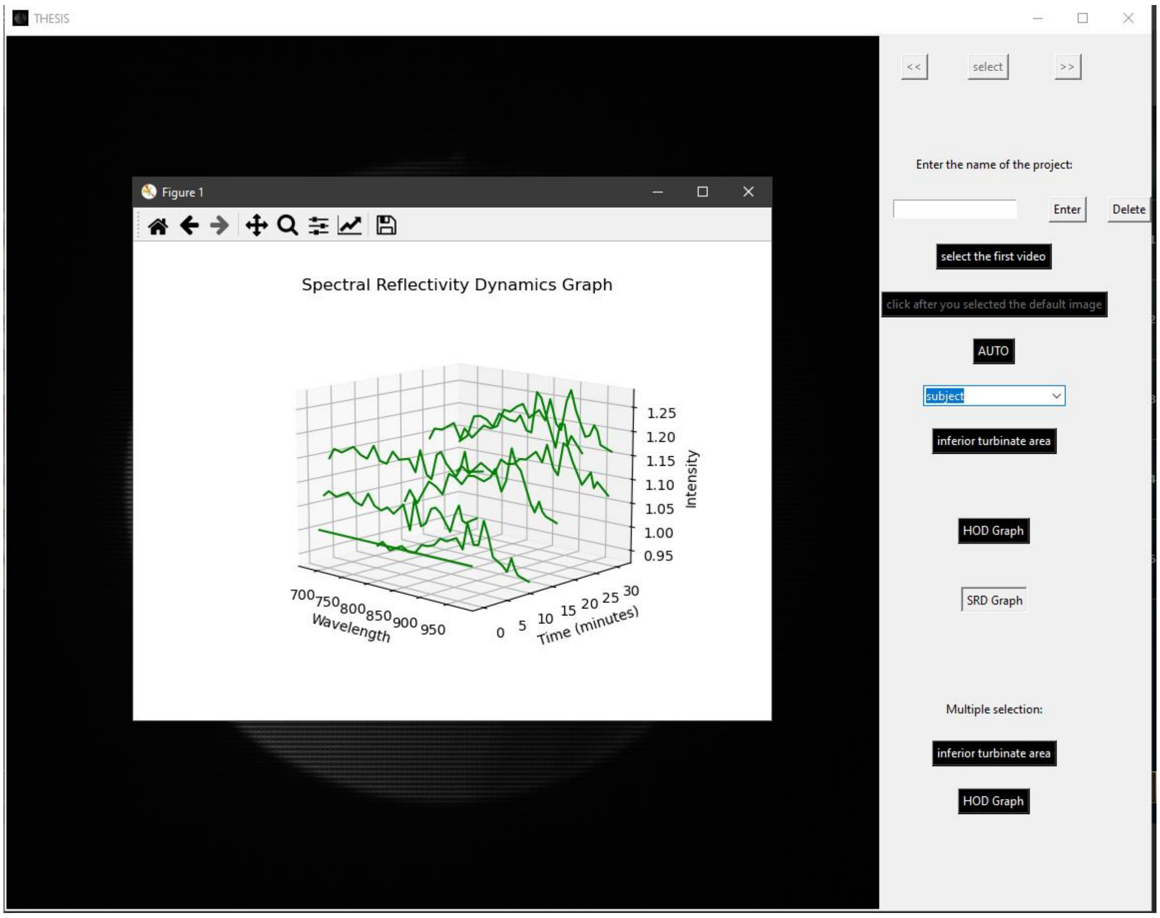


figure S 11. SRD Graph.

Step 10.

Under the label "Multiple selection" the 2 buttons "inferior turbinate area", "HOD graph" (figure S8) do exactly like the other buttons "inferior turbinate area", "HOD graph" on top of that label but when the user presses the inferior turbinate area button the areas, he or she selects are saved so the HOD graph presents the average of those areas. In comparison to the other one where the file is rewritten in order to have only one area every time. This way the user can have a statistically better result. Since the possibility of a wrong selection of area is normalized.

10.2 Python Code

GUI.py

```
from tkinter import *
import numpy as np
from PIL import ImageTk, Image
import cv2
import os
from tkinter import ttk
import project_functions
import functions_time
import functions_default_image
import functions_crop
import analysis_functions
import functions_areas
import general_functions
import selection_functions
import time

global project
global projects_dropdown
global path     # original path
global path_def  # specific path for each project

import multiple #



def frame_selection():
    """ this function is used as its name suggests to select the frames that are the closest to the saved image
        it is used an image metric P.S.N.R.
         to compare the images.The comparison is going to be between the wavelength 746  of every cropped image.
        we are going to select the images that have the best P.S.N.R. score in every time point (every minute).
        the score of every image is going to be stored in a text file and the selected images in a sub_folder of the
        initial (project) folder with the name "selected".
    """
    t0 = time.time()
    filter_wave = 4
    general_functions.remove_files(path + "selected//")
    # saved image
    sorted_table_selection = functions_default_image.default_get(path)
    selected = sorted_table_selection[0][filter_wave]
    # rest of the dataset
    filename = general_functions.import_file("select the dataset")
    sum_of_all_time = []
    sum_of_all_frames = []

    if filename[0]:
        for f in range(len(filename)):
            if filename[f].split(".")[-1] == "avi":
                sorted_table, sum_of_time = selection_functions.pre_process(filename[f])
                time_point = functions_time.time_points(sum_of_time)
                # metrics
                metric = selection_functions.metrics_func(sorted_table, ".txt", f, selected, filter_wave, path)
                metric_time, metric_frames = functions_time.per_minute(time_point, metric, sorted_table, sum_of_time)
                for i in range(len(metric_frames)):
```

```
                sum_of_all_time.append(metric_time[i])
                sum_of_all_frames.append(metric_frames[i])
        # save all
        selection_functions.save_all(sum_of_all_frames, sum_of_all_time, path)
    t1 = time.time()
    t = t1 - t0
    print("time of processing is:  " + str(int(t/60)) + " MIN")
    print("frame selection ended")


def auto():
    t0 = time.time()
    filter_wave = 4

    general_functions.remove_files(path + "selected//")
    # rest of the dataset
    sum_of_all_time = []
    sum_of_all_frames = []
    filename_initial = general_functions.import_file("select the default")
    filename_rest = general_functions.import_file("select the dataset")
    list_of_sums = []
    final_cropped_frames = []
    if filename_initial[0]:
        if filename_initial[0].split(".")[-1] == "avi":
            final_cropped_frames = np.array(functions_crop.final_crop(functions_crop.square_crop(

functions_crop.initial_crop(general_functions.file_opening_selection(filename_initial)))))
            list_of_sums = np.zeros(len(final_cropped_frames), dtype=int)
            for i in range(len(final_cropped_frames)):
                cube_selection = functions_crop.pattern([final_cropped_frames[i]])
                sorted_table_selection = functions_crop.sorted_cube(cube_selection)
                selected = sorted_table_selection[0][filter_wave]
                small_sum = 0
                if filename_rest[0]:
                    for f in range(len(filename_rest)):
                        if filename_rest[f].split(".")[-1] == "avi":
                            sorted_table, sum_of_time = selection_functions.pre_process(filename_rest[f])
                            # metrics
                            metric, small = selection_functions.metrics_func_quick(sorted_table, selected,
filter_wave)
                            small_sum = small_sum + small
                list_of_sums[i] = small_sum
    # saved image
    result = np.where(list_of_sums == np.amax(list_of_sums))
    thesis = result[0][0]
    selected_img = final_cropped_frames[thesis]
    if not os.path.exists(path + "saved"):
        os.makedirs(path + "saved")
    general_functions.remove_files(path + "saved")
    cv2.imwrite(path + "saved//" + str(thesis) + "_selected.png", selected_img)

    sorted_table_selection = functions_crop.sorted_cube(functions_crop.pattern([selected_img]))
    selected = sorted_table_selection[0][filter_wave]

    if filename_rest[0]:
        for f in range(len(filename_rest)):
            if filename_rest[f].split(".")[-1] == "avi":
                sorted_table, sum_of_time = selection_functions.pre_process(filename_rest[f])
                time_point = functions_time.time_points(sum_of_time)
                # metrics
                metric = selection_functions.metrics_func(sorted_table, ".txt", f, selected, filter_wave,
path)

                metric_time, metric_frames = functions_time.per_minute(time_point, metric,
sorted_table, sum_of_time)
                for i in range(len(metric_frames)):
```

```
                sum_of_all_time.append(metric_time[i])
                sum_of_all_frames.append(metric_frames[i])

        selection_functions.save_all(sum_of_all_frames, sum_of_all_time, path)
    t1 = time.time()
    t = t1 - t0
    print("time of processing is:  " + str(int(t/60)) + " MIN")
    print("frame selection ended")



def enable():
    """this function enables all the buttons for the visualisation! """
    Button(root,
           text='HOD Graph',
           compound=LEFT,
           command=lambda: analysis_functions.method_div(path),
           bg="black",
           fg="white",
           ).grid(row=12, column=3, columnspan=3)

    Button(root,
           text='SRD Graph',
           compound=LEFT,
           bg="black",
           fg="white",
           command=lambda: analysis_functions.method_wave(path)
           ).grid(row=14, column=3, columnspan=3)



def multiple_en():  #
    Label(root, text="Multiple selection:").grid(row=18, column=3, columnspan=3)
    Button(root,
           text='inferior turbinate area',
           compound=LEFT,
           bg="black",
           fg="white",
           command=lambda: multiple.area("mucosa", path)
           ).grid(row=19, column=3, columnspan=3)
    Button(root,
           text='HOD Graph',
           compound=LEFT,
           command=lambda: multiple.method_div_mul(path),
           bg="black",
           fg="white",
           ).grid(row=20, column=3, columnspan=3)


def multiple_dis():  #
    Label(root, text="Multiple selection:").grid(row=18, column=3, columnspan=3)
    Button(root,
           text='inferior turbinate area',
           compound=LEFT,
           bg="black",
           fg="white",
           state=DISABLED
           ).grid(row=19, column=3, columnspan=3)
    Button(root,
           text='HOD Graph',
           compound=LEFT,
           bg="black",
           fg="white",
           state=DISABLED
           ).grid(row=20, column=3, columnspan=3)


def enable_areas():
    """it enables the buttons that are used for selecting the area of  mucosa"""
    Button(root,
           text='inferior turbinate area',
           compound=LEFT,
           bg="black",
           fg="white",
           command=lambda: functions_areas.area("mucosa", path)
           ).grid(row=9, column=3, columnspan=3)


def areas_dis():
    """this function disables the buttons that are used for the selection of the areas on the image"""
    Button(root,
           text='inferior turbinate area',
           compound=LEFT,
           bg="black",
           fg="white",
           state=DISABLED
           ).grid(row=9, column=3, columnspan=3)


def disable_vis():
    """this function disables all the buttons that are used for visualisation"""
    Button(root,
           text='HOD Graph',
           compound=LEFT,
           bg="black",
           fg="white",
           state=DISABLED
           ).grid(row=12, column=3, columnspan=3)
    Button(root,
           text='SRD Graph',
           compound=LEFT,
        bg="black",
        fg="white",
        state=DISABLED
        ).grid(row=14, column=3, columnspan=3)


def select_image():
    """ this code runs after the user has selected the first video ,which is also the default video(without the
        pharmaceutical agent). it enables the user to press the buttons on the top right side of the GUI(3 buttons and
        a bar) and it loads all the frames of that video and shows it to the user so that he or she can select the image
        that he or she thinks is an image that can be the representative in order to be compared with the rest of the
        dataset.
        with the buttons and bar the user can navigate through the images.
        after the user selects the image with the "select" button the frame that was shown is saved!
    """
    # definitions start
    all_frames = []
    all_frames_np = []
    # definitions end

    def back(image_number):
        """ the function of the << button"""

        if (image_number >= 0) and (image_number <= len(all_frames)):
            Label(image=all_frames[image_number]).grid(row=0, column=0, columnspan=3, rowspan=25)
            horizontal.set(image_number + 1)
        else:
            print("error")
        Label(root, text=("image " + str(image_number + 1) + " of " + str(len(all_frames))), bd=1,
              relief=SUNKEN, anchor=E).grid(row=2, column=3, columnspan=4)
        Button(root, text=">>", command=lambda: forward(image_number + 1)).grid(row=0,
column=5)
        if image_number - 1 == 0:
            Button(root, text="<<", state=DISABLED).grid(row=0, column=3, pady=10)
        else:
            Button(root, text="<<", command=lambda: back(image_number - 1)).grid(row=0,
column=3, pady=10)

    def select():
        state_of_i = horizontal.get() - 1
        """ the function of the select button"""

        if not os.path.exists(path + "saved"):
            os.makedirs(path + "saved")
        general_functions.remove_files(path + "saved")
        selected = all_frames_np[state_of_i]
        cv2.imwrite(path + "saved//" + str(state_of_i) + "selected.png", selected)
        Button(root,
               text='click after you selected the default image',
               command=frame_selection,
               compound=LEFT,
               bg="black",
               fg="white",
               ).grid(row=6, column=3, columnspan=3)

    def forward(image_number):
        """ the function of the >> button"""
        if (image_number >= 0) and (image_number <= len(all_frames)):
            Label(image=all_frames[image_number]).grid(row=0, column=0, columnspan=3,
rowspan=25)
            horizontal.set(image_number + 1)
        else:
            print("error")
        Label(root, text=("image " + str(image_number + 1) + " of " + str(len(all_frames))), bd=1,
              relief=SUNKEN, anchor=E).grid(row=2, column=3, columnspan=4)
        Button(root, text="<<", command=lambda: back(image_number - 1)).grid(row=0,
column=3, pady=10)
        if image_number + 1 == len(all_frames):
            Button(root, text=">>", state=DISABLED).grid(row=0, column=5)
        else:
            Button(root, text=">>", command=lambda: forward(image_number + 1)).grid(row=0,
column=5)

    def slide(var):
        """ the function of the bar """
        state_of_i = horizontal.get() - 1
        Label(root, text=("image " + str(state_of_i + 1) + " of " + str(len(all_frames))), bd=1,
              relief=SUNKEN, anchor=E).grid(row=2, column=3, columnspan=4)
        Label(image=all_frames[state_of_i]).grid(row=0, column=0, columnspan=3, rowspan=25)

        if state_of_i == 0:
            Button(root, text="<<", state=DISABLED).grid(row=0, column=3, pady=10)
        else:
            Button(root, text="<<", command=lambda: back(state_of_i - 1)).grid(row=0, column=3,
pady=10)

        if state_of_i + 1 == len(all_frames):
            Button(root, text=">>", state=DISABLED).grid(row=0, column=5)
        else:
            Button(root, text=">>", command=lambda: forward(state_of_i + 1)).grid(row=0,
column=5)

    filename = general_functions.import_file("select the first video")
    if filename[0]:
```

```
    if filename[0].split(".")[-1] == "avi":
        video = general_functions.file_opening_selection(filename)
        frames = np.array(video)
        cropped_frames = np.array(functions_crop.initial_crop(frames))
        square_cropped_frames = np.array(functions_crop.square_crop(cropped_frames))
        final_cropped_frames = np.array(functions_crop.final_crop(square_cropped_frames))
        all_frames_np = final_cropped_frames

        # GUI START
        for i in range(len(final_cropped_frames)):
            all_frames.append(ImageTk.PhotoImage(Image.fromarray(final_cropped_frames[i])))
        my_label = Label(image=all_frames[0])
        my_label.grid(row=0, column=0, columnspan=3, rowspan=25)
        status_image = Label(root, text=("image " + str(1) + " of " + str(len(all_frames))), bd=1,
                             relief=SUNKEN, anchor=E)
        button_forward = Button(root, text=">>", command=lambda: forward(1))
        button_select = Button(root, text="select", command=select)
        button_back = Button(root, text="<<", state=DISABLED)
        horizontal = Scale(root, from_=1, to=len(all_frames), orient=HORIZONTAL,
command=slide)

        button_forward.grid(row=0, column=5)
        button_back.grid(row=0, column=3, pady=10)
        horizontal.grid(row=1, column=3, columnspan=4, sticky=W + E)
        status_image.grid(row=2, column=3, columnspan=4)
        my_label.grid(row=0, column=0, columnspan=3, rowspan=25)
        button_select.grid(row=0, column=4)
        # GUI END



def project_selection(event):
    """The user selects what folder or data wants to visualise.
      The user selects between the folders where the saved data are stored(after the analysis).
```

```
        it calls some functions in order for the user to select and visualise what he wants.
    """
    global path
    for i in range(len(project)):
        if projects_dropdown.get() == project[i]:
            path = path_def + "projects//"+projects_dropdown.get()+"//"
            enable()
            enable_areas()
            multiple_en()  #



def take_entry():
    """takes a str from the user that is going to be used for the name of the folder that the data are
going to
        be saved!
        In other words it changes the path global variable.
    """
    global path
    global project
    global projects_dropdown

    user_input.get()
    project_functions.write(path_def, user_input.get())
    project = project_functions.read(path_def)
    dropdown_list()
    path = path_def + "projects//" + user_input.get()+"//"



def take_entry_to_delete():
    global path
    global project
    global projects_dropdown
    user_input.get()
    project_functions.delete_project(path_def, user_input.get())

    project = project_functions.read(path_def)
    dropdown_list()
    path = path_def + "projects/" + user_input.get()+"/"



def dropdown_list():
    global projects_dropdown

    projects_dropdown = ttk.Combobox(root, values=project)
    projects_dropdown.current(0)
    projects_dropdown.bind("<<ComboboxSelected>>", project_selection)
    projects_dropdown.grid(row=8, column=3, columnspan=3)



def default_gui():
    Label(root, image=my_img1).grid(row=0, column=0, columnspan=3, rowspan=25)
    Button(root, text=">>", state=DISABLED).grid(row=0, column=5)
    Button(root, text="select", state=DISABLED).grid(row=0, column=4)
    Button(root, state=DISABLED, text="<<").grid(row=0, column=3, pady=10)
    Label(root, text="Enter the name of the project:").grid(row=3, column=3, columnspan=3)
    Button(root,
           text='select the first video',
           command=select_image,
           compound=LEFT,
           bg="black",
           fg="white",
           ).grid(row=5, column=3, columnspan=3)
    Button(root,
           text='click after you selected the default image',
           compound=LEFT,
           bg="black",
           fg="white",
           state=DISABLED
           ).grid(row=6, column=3, columnspan=3)
    Button(root,
           text='AUTO',
           command=auto,
           bg="black",
           fg="white",
           compound=LEFT).grid(row=7, column=3, columnspan=3)


if __name__ == "__main__":
    # PATH START
    path = general_functions.get_path()
    path_def = path
    project = project_functions.read(path_def)
    # PATH END

    # default GUI START
    root = Tk()
    root.title("THESIS")
    root.wm_iconbitmap('def_images//image.ico')
    my_img = ImageTk.PhotoImage(Image.open("def_images/image.png"))
    my_img1 = ImageTk.PhotoImage(Image.open("def_images/image_def.png"))
    # default GUI END

    # initial buttons that change once the user interacts with the app START
    dropdown_list()
    user_input = StringVar(root)
    Entry(root, textvariable=user_input).grid(row=4, column=3, columnspan=2)
    Button(root, text='Enter', command=take_entry).grid(row=4, column=5, columnspan=1)
    Button(root,      text='Delete',      command=take_entry_to_delete).grid(row=4,      column=6,
columnspan=1)
    default_gui()
    disable_vis()
    areas_dis()
    multiple_dis()
    # initial buttons that change once the user interacts with the app END
    root.mainloop()
```

analysis_functions.py

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import functions_default_image
import general_functions
import functions_crop
import functions_areas



def divided(sum_of_all_frames):
    """ After the frame selection, with each selected frame we are going to divide the wavelength
683 with the
        wavelength 873.
        This division  is going to be the oxygenated hemoglobin distribution map.
    """
    all_div                                    =                                    np.zeros([len(sum_of_all_frames)],
dtype=type((cv2.divide(np.float32(sum_of_all_frames[0][0]),
                                                              np.float32(sum_of_all_frames[0][14]))) * 100))
    for i in range(len(sum_of_all_frames)):
        all_div[i]                        =                        (cv2.divide(np.float32(sum_of_all_frames[i][0]),
np.float32(sum_of_all_frames[i][14]))) * 100
        # 680 873
        #               divided_1               =               (cv2.divide(np.float32(sum_of_all_frames[i][4]),
np.float32(sum_of_all_frames[i][11]))) * 100
        # 840 and 746

    return all_div



def transition(spliced, initial):
    """ it returns time in a format of real minutes"."seconds microseconds.
        minutes are from 0 to 60, but we now that our measurements lasts 30 to 40 minutes.
    """
    minutes = int(spliced[1])
    minutes_initial = int(initial[1])
    minutes_real = minutes - minutes_initial

    if minutes_real < 0:
        minutes_real = 60 + minutes_real
    elif minutes == minutes_initial:
        minutes_real = 60
    return float(str(minutes_real) + "." + spliced[2] + spliced[3])



def extract_sums(temp, path):
    """ it averages the images and the time so that we have 1 point """
    # declarations start
    sum_of_all_frames = []
    sum_of_time = []
    sorted_table = functions_crop.table()[0]
    # declarations end

    # 1 if we use the default frame selection
    # otherwise if we use the algo that does frame selection based on the cropped images
    if temp == 1:
        images, filename = general_functions.load_images_from_folder(path + "selected//")
    else:
        images, filename = general_functions.load_images_from_folder(path +
"selected_cropped//")
    # the method is going to be the same we just change the folder from which the images are going
to be loaded
    for i in range(0, len(images), 25):
        sum_of_time.append(filename[i].split("_")[0])
        sum_of_frames = []
        for j in range(len(sorted_table)):
            sum_of_frames.append(images[j+i])
        sum_of_all_frames.append(sum_of_frames)
    sum_of_all_frames = np.array(sum_of_all_frames)
    sum_of_all_time = np.array(sum_of_time)
    return sum_of_all_time, sum_of_all_frames


def averaging(sum_of_divided_frames, sum_of_divided_frames_def, def_time, sum_of_time,
dimensions_mu):
    """Since we want to have 1 point on our graph in the case of more than one image with the
same higher P.S.N.R. score
       at the same minute we average the intensities that we found for each frame.
       After the processing a 2d diagram will appear in the screen that will visualize the
       intensity of hemoglobin absorption per minute (time point).
    """
    # declarations start
    average_all_mu = []
    time_point = []
    initial = sum_of_time[0].split("-")
    # declarations end

    for i in range(len(sum_of_time)):
        spliced = sum_of_time[i].split("-")
        for num in range(4):
            if len(spliced[num]) == 1:
                spliced[num] = "0" + spliced[num]
        time_point.append(transition(spliced, initial))
        image = sum_of_divided_frames[i]
        cropped_img_mu            =            np.average(image[dimensions_mu[0]:dimensions_mu[1],
dimensions_mu[2]:dimensions_mu[3]])
        average_all_mu.append(cropped_img_mu)

    def_cropped_img_mu                                                                   =
np.average(sum_of_divided_frames_def[dimensions_mu[0]:dimensions_mu[1],
                                            dimensions_mu[2]:dimensions_mu[3]])
    def_average_all_muc = def_cropped_img_mu
    avg_def_muc = np.array([def_average_all_muc], dtype=np.float32)
    avg_intensity, avg_time = between(time_point, average_all_mu, avg_def_muc, def_time)
    plt.plot(avg_time, avg_intensity, "ro-")
    plt.xlabel("Time (min)")
    plt.xlim(-1, 40)
    plt.ylabel("Haemoglobin Oxygenation (a.u.)")
    plt.title("Hemoglobin Oxygenation Dynamic Graph ")
    plt.show()



def method_div(path):
    """When the user clicks the "HOD Graph" button the calculation of the intensity of hemoglobin
absorption per minute
       (time point) will begin.
       We do that by dividing the two distinct spectral bands (in our case 680 and 873).
       The first time point is the reference time point before the application of the pharmaceutical
agent.
    """
    sum_of_all_time, sum_of_all_frames = extract_sums(1, path)
    sum_of_divided_frames = divided(sum_of_all_frames)
    sum_of_divided_frames_def, def_time = functions_default_image.default_div(path)
    dimensions_mu = functions_areas.read("mucosa", path)
    averaging(sum_of_divided_frames, sum_of_divided_frames_def, def_time, sum_of_all_time,
dimensions_mu)


def method_wave(path):
    """ 3-dimentional graph
        In each of the three graphs we normalize the intensity by dividing them with the
        same value that we found for the default image.
    """
    sum_of_all_time, sum_of_all_frames = extract_sums(1, path)
    sorted_selection, time_selection = functions_default_image.default_3d(path)
    average1 = []
    average2 = []
    # for int and time start
    intensity = []
    time_point_all = []
    for i in range(25):
        wave = []
        for j in range(len(sum_of_all_frames)):
            wave.append(sum_of_all_frames[j][i])
        int_v, time_point = averaging_wave(wave, sum_of_all_time, path)
        intensity.append(int_v)
        time_point_all.append(time_point)

    intensity = np.array(intensity)
    time_point = np.array(time_point_all)
    sorted_selection = np.array(sorted_selection)

    for i in range(25):
        avg1, avg2 = between(time_point[i], intensity[i], sorted_selection[i],
np.float64(time_selection))
```

```
        average1.append(avg1)
        average2.append(avg2)
    intensity = np.array(average1, dtype=object)
    time_point = np.array(average2)
    # for int and time end

    # sorted table start
    sorted_table, none = functions_crop.table()
    sort = []
    for i in range(25):
        x = []
        for j in range(len(average1[0])):
            x.append(sorted_table[i])
        sort.append(x)
    sorted_table = np.array(sort)
    # sorted table end

    # plot start
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    for j in range(len(sorted_table[0])):
        plt.plot(sorted_table[:, j], time_point[:, j], intensity[:, j]/intensity[:, 0], "g")
    ax.set_xlabel('Wavelength')
    ax.set_ylabel('Time (minutes)')
    ax.set_zlabel('Intensity')
    plt.title('Spectral Reflectivity Dynamics Graph ')
    plt.show()
    # plot end


def between(time_point, average, avg_def, def_time):
    """ averaging of time and intensity per minute in case we have more than one images with the
same P.S.N.R. score"""
```

```
# declarations start
new_time_point = []
new_average = []
time_point = np.array(time_point)
average = np.array(average)
all_pos = []
average1 = []
average2 = []
# declarations end

for i in range(34):
    b = time_point[time_point <= i + 1]
    c = time_point[time_point >= i]
    d = np.intersect1d(b, c)
    if len(d) != 0:
        new_time_point.append(d)
        pos = []
        for p in range(len(d)):
            result = list(np.where(time_point == d[p]))


            pos.append(result[0])
        all_pos.append(pos)
for i in range(len(all_pos)):
    average_small = []
    for j in range(len(all_pos[i])):
        average_small.append(average[all_pos[i][j]])
    new_average.append(average_small)

# initiate with default image start
average1.append(avg_def)
average2.append(def_time)
# initiate with default image end
```

```
    for i in range(len(new_average)):
        avg = sum(new_average[i]) / len(new_average[i])
        avg2 = sum(new_time_point[i]) / len(new_time_point[i])
        average1.append(avg)
        average2.append(avg2)
    return average1, average2



def averaging_wave(sum_of_wave_frames, sum_of_time, path):
    """ algorithm that can produce  3-dimentional graph"""
    # declarations start
    average_all = []
    time_point = []
    initial = sum_of_time[0].split("-")
    dimensions_mu = functions_areas.read("mucosa", path)
    # declarations end

    for i in range(len(sum_of_time)):
        spliced = sum_of_time[i].split("-")
        for num in range(4):
            if len(spliced[num]) == 1:
                spliced[num] = "0" + spliced[num]
        time_point.append(transition(spliced, initial))
        image = sum_of_wave_frames[i]
        cropped_img_mu          =          image[dimensions_mu[0]:dimensions_mu[1],
dimensions_mu[2]:dimensions_mu[3]]
        average_all.append(np.average(cropped_img_mu))
    return average_all, time_point
```

functions_areas.py

```
import cv2
import os
import functions_default_image
```

```
cropping = False
x_start, y_start = 0, 0


def read(name, path):
    """this function reads the file that contains the coordinates of the mucosa , black and white
areas"""
    file = open(path + "areas//"+name+".txt", "r")
    x = file.readlines()
    for i in range(len(x)):
        try:
            x[i] = int(x[i].split("\n")[0])
        except ValueError as error:
            print(error)
    file.close()

    return x


def save(path, name, ref_point):
    """
        this function will save the coordinates in a txt file
        :param path: the path that will save the coordinates
        :param name: name of the file that the coordinates will be saved
        :param ref_point: the list that contains the coordinates
    """
    if not os.path.exists(path + "areas"):
        os.makedirs(path + "areas")
    file = open(path + "areas//" + name + ".txt", "w")
    file.write(str(ref_point[0][1]) + "\n")
    file.write(str(ref_point[1][1]) + "\n")
    file.write(str(ref_point[0][0]) + "\n")
    file.write(str(ref_point[1][0]) + "\n")
    file.close()



def area(name, path):
    """ the player will double click in order to select the requested area on the image that will pop
after clicking
         inferior turbinate area button and then save the coordinates. if the user presses "q" the area
        selection will stop.
    """
    image = functions_default_image.default_areas(path)[0]

    def mouse_crop(event, x, y, flags, param):
        """ """
        # grab references to the global variables
        global x_start, y_start, cropping

        # if the left mouse button was DOWN, start RECORDING
        # (x, y) coordinates and indicate that cropping is being
        if event == cv2.EVENT_LBUTTONDBLCLK:
            x_start, y_start = x, y
            cropping = True
            ref_point = [(x_start - 2, y_start - 2), (x_start + 2, y_start + 2)]

            save(path, name, ref_point)

    cv2.namedWindow("image")
    cv2.setMouseCallback("image", mouse_crop)

    while True:
        i = image.copy()
        if not cropping:
            cv2.imshow("image", image)
        elif cropping:
            cv2.rectangle(i, (x_start - 2, y_start - 2), (x_start + 2, y_start + 2), (0, 0, 255), 2)
            cv2.imshow("image", i)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    # close all open windows
    cv2.destroyAllWindows()
```

functions_crop.py

```
import numpy as np



# CROPPING START
def initial_crop(sum_of_frames):
    sum_cropped = []
    for num in range(len(sum_of_frames)):
        temp = sum_of_frames[num]
        temp2 = np.delete(temp, [0, 1, 2], 0)  # rows
        temp = np.delete(temp2, [-3, -2, -1], 1)  # cols
        sum_cropped.append(temp)

    return sum_cropped



def square_crop(sum_of_frames):
    """ make the image dimensions square nxn """
    # definition start
    sum_of_frames = np.array(sum_of_frames)
    y = len(sum_of_frames[0])
    z = len(sum_of_frames[0][0])
    sum_cropped = []
    sum_cropped2 = []
    # definition end
    if z > y:
        times = int((z-y)/10)
        for num in range(len(sum_of_frames)):
            temp = sum_of_frames[num]

            for i in range(times):
                temp2 = np.delete(temp, [0, 1, 2, 3, 4, -5, -4, -3, -2, -1], 1)
                temp = temp2
            sum_cropped.append(temp)
        return sum_cropped

    elif z < y:
        times = int((y-z)/10)
        for num in range(len(sum_of_frames)):
            temp = sum_cropped[num]
            for i in range(times):
                temp2 = np.delete(temp, [0, 1, 2, 3, 4, -5, -4, -3, -2, -1], 0)
                temp = temp2
            sum_cropped2.append(temp)

        return sum_cropped2
    else:
        return sum_of_frames



def final_crop(sum_of_frames):
    sum_of_frames = np.array(sum_of_frames)
    z = len(sum_of_frames[0][0])
    sum_cropped = []
    times = int(z/50)
    for num in range(len(sum_of_frames)):
        temp = np.array(sum_of_frames[num])
        for i in range(times):
            temp2 = np.delete(temp, [0, 1, 2, 3, 4, -5, -4, -3, -2, -1], 1)
            temp = np.delete(temp2, [0, 1, 2, 3, 4, -5, -4, -3, -2, -1], 0)
        sum_cropped.append(temp)

    return sum_cropped
# CROPPING END



# PATTERN START
def pattern(sum_of_frames):
    """ this function creates the cube """
    cube = []
    for num in range(len(sum_of_frames)):
        frame = np.array(sum_of_frames[num])
        cube_of_frame = []
        for i in range(5):
            for j in range(5):
                cube_of_frame.append(frame[i::5, j::5])
        cube.append(cube_of_frame)
    big_cube_of_frames = np.array(cube)

    return big_cube_of_frames



def table():
    table_wave = np.array([[900, 909, 892, 882, 683],
                           [809, 821, 797, 784, 693],
                           [759, 772, 746, 732, 708],
                           [943, 949, 935, 927, 975],
                           [861, 873, 852, 840, 955]])
    table1d = table_wave.flatten()
    table_sort = np.sort(table1d)
    return table_sort, table1d



def sorted_cube(cube):
    """ this function sorts the cube in ascending order """

    table_sort, table1d = table()
    posit = []
    for i in range(25):
        result = np.where(table1d == table_sort[i])
        for j in range(len(result)):
            posit.append(result[0][j])
    sorted_array = []

    for i in range(len(cube)):
        z = []
        for j in range(len(cube[0])):
            z.append(cube[i][posit[j]])
        sorted_array.append(z)
    sorted_array = np.array(sorted_array)

    return sorted_array
# PATTERN END
```

functions_default_image.py

```
import functions_crop
import cv2
import numpy as np
import analysis_functions
import os
import functions_areas
```

```
def default_get(path):
    # selected image
    filename = os.listdir(path+"saved//")[0]
    selected_img = np.array([cv2.imread(path + "saved//" + filename, 0)])
    cube_selection = functions_crop.pattern(selected_img)
    sorted_table_selection = functions_crop.sorted_cube(cube_selection)
    return sorted_table_selection


def default_div(path):
    """ in this function we do all the necessary modifications to the default saved image in order to
be visualised
        in the 2d diagram at time point 0.
    """
    sorted_table_selection = default_get(path)
    all_div = analysis_functions.divided(sorted_table_selection)[0]
    def_time = 0

    return all_div, def_time


def default_areas(path):
    """ in this function we do all the necessary modifications to the default saved image in order to
    """
    sorted_table_selection = default_get(path)

    return sorted_table_selection[0]


def default_3d(path):
    """ in this function we do all the necessary modifications to the default saved image in order to
be visualised
        in the 3d diagram at time point 0.
    """
    # declarations start
    sorted_table_selection = default_get(path)
    def_time = 0.0  # we use 0 so that the separation  is going to be clear to the eye
    average_all = []
    dimensions_mu = functions_areas.read("mucosa", path)
    # declarations end

    for i in range(25):
        cropped_img_mu = sorted_table_selection[0][i][dimensions_mu[0]:dimensions_mu[1],
                                                   dimensions_mu[2]:dimensions_mu[3]]
        average_all.append(np.average(cropped_img_mu))
    return average_all, def_time
```

functions_time.py

```
import numpy as np



def transition(spliced, initial):
    """ transform time into the format minute.sec m sec
        time was not in the form of 0-60 but it had the actual time. for example 11:30
        so we transition it in a form that would be suitable for the graphs.
    """
    hour = int(spliced[0])
    minutes = int(spliced[1])
    hour_initial = int(initial[0])
    minutes_initial = int(initial[1])

    # the transition
    hour_real = hour - hour_initial
    minutes_real = minutes - minutes_initial
    # in case that the minutes_real is less than 0 means that the hour has changed for example from
9 to 10
    # in that case we need to add 60 in order that the result is positive and correct.
    if minutes_real < 0 and hour_real == 1:
        minutes_real = 60 + minutes_real

    return float(str(minutes_real) + "." + spliced[2] + spliced[3])



def time_points(sum_of_time):
    """
        This function returns the time in a suitable form, which is 0.0 to 60(we know that because
the examination
        takes place for about 0 to 40 minutes). It can work of course for more time!
    """
    # initial is a list of the starting time split in an array with its elements containing the initial hour
min
    # sec and m sec.
    # splice is a temporal list that contains for each moment the variables of time (hour minute sec
m sec)
    # the for loop adds a "0"  in the beginning of every variable of time(hour minute sec m sec if it
originally
    # had length of 1) because we want each variable to have at least a length of 2
    # for each element of sum_of_time after we use the function transition we load the modified
values in a new array
    # that will be our new time for now on.

    time_point = []
    initial = sum_of_time[0].split("-")

    for i in range(len(sum_of_time)):
        spliced = sum_of_time[i].split("-")
        for num in range(4):
            if len(spliced[num]) == 1:
                spliced[num] = "0" + spliced[num]
        time_point.append(transition(spliced, initial))

    return time_point



def per_minute(time_point, metric, sorted_table, sum_of_time):
    """ this function finds the frames and time points that score the best P.S.N.R. score every
minute and returns them
    """
    # definition start
    sum_of_all_time = []
    sum_of_all_frames = []
    initial = int(time_point[0])
    start_pos = 0
    sub_metric = []
    temp = 0
    # definition end

    for i in range(len(sorted_table)):
        if int(time_point[i]) == initial:
            sub_metric.append(metric[i])
            temp = 0
        elif int(time_point[i]) == initial + 1:
            meg = np.max(sub_metric)
            for k in range(start_pos, i-1):
                if metric[k] == meg:
                    sum_of_all_time.append(sum_of_time[k])
                    sum_of_all_frames.append(sorted_table[k])
            start_pos = i
            initial = int(time_point[i])
            sub_metric = [metric[i]]
            temp = 1

    if temp == 0:
        meg = np.max(sub_metric)
        for k in range(start_pos, len(sorted_table)):
            if metric[k] == meg:
                sum_of_all_time.append(sum_of_time[k])
                sum_of_all_frames.append(sorted_table[k])

    return sum_of_all_time, sum_of_all_frames
```

general_functions.py

```
import cv2
import numpy as np
import os
from tkinter import filedialog



def get_path():
    """ finding the path """
    path_ori = str(os.path.dirname(__file__))
    path_ori = path_ori.replace("/", "//")
    path_ori += "//"
    return path_ori



def file_opening_selection(filename):
    """
        this function opens the video with the name filename and saves all the frames in a list called
sum_of_frame
        :param filename: the name of the video
        :return: sum_of_frames which is a list that contains all the frames of the video
    """
    # definition start
    video = cv2.VideoCapture(filename[0])
    sum_of_frames = []
    num_frames = video.get(cv2.CAP_PROP_FRAME_COUNT)
    i = 0
    # definition end

    while i < num_frames:
        ret, frame = video.read()
        frame = frame[:, :, 0]
        sum_of_frames.append(frame)
        i = i + 1
    video.release()
    sum_of_frames = np.array(sum_of_frames)

    return sum_of_frames



def import_file(title):
    filename = np.asarray(filedialog.askopenfilenames(title=title))
    print('Selected:', filename)

    return filename



def time_calculation(file):
    split = file.split(" ")
    times = split[1]
    hour = int(times.split("-")[0])
    minutes = int(times.split("-")[1])
    sec = int(times.split("-")[2])

    return sec, hour, minutes
```

```
def single_video_opening(file):
    """ opens a video and returns the frames and the time of each frame """
    # definition start
    sum_of_frames = []
    sum_of_time = []
    i = 0
    video = cv2.VideoCapture(file)
    rate = video.get(cv2.CAP_PROP_FPS)
    num_frames = video.get(cv2.CAP_PROP_FRAME_COUNT)
    # definition end
    sum_s, sum_h, sum_m = time_calculation(file)
    while i < num_frames:
        ret, frame = video.read()
        frame = frame[:, :, 0]
        sum_of_frames.append(frame)
        sum_s = sum_s + 1 / rate
        if sum_s >= 60:
            sum_m = sum_m + 1
            sum_s = sum_s - 60
        if sum_m >= 60:
            sum_h = sum_h + 1
            sum_m = 0
        sec_all = str(round(sum_s, 4)).split(".")
        sec = sec_all[0]
        sec_mi = sec_all[1]
        if sec_mi == "0":
            sec_mi = "0000"
        sum_of_time.append(str(sum_h) + "-" + str(sum_m) + "-" + sec + "-" + sec_mi)
        i = i + 1

    return sum_of_frames, sum_of_time
```

```
def remove_files(my_dir):
    """ removes files from a directory """
    if not os.path.exists(my_dir):
        os.makedirs(my_dir)
    file_list = [f for f in os.listdir(my_dir) if f.endswith(".png")]
    for f in file_list:
        os.remove(os.path.join(my_dir, f))



def load_images_from_folder(folder):
    """ this function loads all the images stored in any folder. however we know that with this case
we are going
        to use it to load the images of the project we work that are on the sub_folder selected_images
and return the
        images and their names which are distinct.
    """
    images = []
    files = []

    for filename in os.listdir(folder):
        files.append(filename)
        img = cv2.imread(os.path.join(folder, filename))
        img = img[:, :, 0]  # gray images
        if img is not None:
            images.append(img)

    return images, files
```

multiple.py

```
import cv2
import os
import functions_default_image
import analysis_functions
import matplotlib.pyplot as plt
import numpy as np

cropping = False
x_start, y_start = 0, 0
point = 0



def read_mul(path, name):
    """this function reads the file that contains the coordinates of the mucosa , black and white
areas"""
    txt_files = []
    x_all = []

    for root, dirs, files in os.walk(path + "areas"):
        for file_i in files:
            if name == "mucosa":
                if "mucosa" in file_i:
                    # append the file name to the list
                    txt_files.append(os.path.join(root, file_i))
                    file = open(path + "areas//" + file_i, "r")
                    x = file.readlines()
                    for i in range(len(x)):
                        try:
                            x[i] = int(x[i].split("\n")[0])
                        except ValueError as error:
                            print(error)
                    x_all.append(x)
                    file.close()
            else:
```

```
            if "white" in file_i:
                # append the file name to the list
                txt_files.append(os.path.join(root, file_i))
                file = open(path + "areas//" + file_i, "r")
                x = file.readlines()
                for i in range(len(x)):
                    try:
                        x[i] = int(x[i].split("\n")[0])
                    except ValueError as error:
                        print(error)
                x_all.append(x)
                file.close()
    print("number of files:")
    print(len(x_all))
    return x_all


def save(path, name, ref_point, point):
    """
        this function will save the coordinates in a txt file
        :param path: the path that will save the coordinates
        :param name: name of the file that the coordinates will be saved
        :param ref_point: the list that contains the coordinates
        :param point: num of point
    """
    if not os.path.exists(path + "areas"):
        os.makedirs(path + "areas")
    file = open(path + "areas//" + name + "_" + str(point) + ".txt", "w")
    file.write(str(ref_point[0][1]) + "\n")
    file.write(str(ref_point[1][1]) + "\n")
    file.write(str(ref_point[0][0]) + "\n")
    file.write(str(ref_point[1][0]) + "\n")
    file.close()
```

```
def area(name, path):
    """ the player will double click in order to select the requested area on the image that will pop
after clicking
        inferior turbinate area button and then save the coordinates. if the user presses "q" the area
        selection will stop.
    """
    image = functions_default_image.default_areas(path)[0]
    size = 2

    def mouse_crop(event, x, y, flags, param):
        # grab references to the global variables
        global x_start, y_start, cropping
        global point
        # if the left mouse button was DOWN, start RECORDING
        # (x, y) coordinates and indicate that cropping is being
        if event == cv2.EVENT_LBUTTONDBLCLK:
            point += 1
            x_start, y_start = x, y
            cropping = True
            ref_point = [(x_start - size, y_start - size), (x_start + size, y_start + size)]

            save(path, name, ref_point, point)

    cv2.namedWindow("image")
    cv2.setMouseCallback("image", mouse_crop)

    while True:
        i = image.copy()
        if not cropping:
            cv2.imshow("image", image)
        elif cropping:
            cv2.rectangle(i, (x_start - 2, y_start - 2), (x_start + 2, y_start + 2), (0, 0, 255), 2)
            cv2.imshow("image", i)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    # close all open windows
    cv2.destroyAllWindows()


def method_div_mul(path):
    """When the user clicks the "DIVISION" button the calculation of the intensity of hemoglobin absorption per minute
        (time point) will begin.
        We do that by dividing the two distinct spectral bands (in our case 680 and 873).
        The first time point is the reference time point before the application of the pharmaceutical agent.
    """
    sum_of_all_time, sum_of_all_frames = analysis_functions.extract_sums(1, path)
    sum_of_divided_frames = analysis_functions.divided(sum_of_all_frames)
    sum_of_divided_frames_def, def_time = functions_default_image.default_div(path)
    dimensions_mu_array = read_mul(path, "mucosa")
    averaging_mult(sum_of_divided_frames, sum_of_divided_frames_def, def_time, sum_of_all_time,
                   dimensions_mu_array)


def averaging_mult(sum_of_divided_frames, sum_of_divided_frames_def, def_time, sum_of_time, dimensions_mu_array):
    """Since we want to have 1 point on our graph in the case of more than one image with the same higher P.S.N.R. score
        at the same minute we average the intensities that we found for each frame.
        After the processing a 2d diagram will appear in the screen that will visualize the
        intensity of hemoglobin absorption per minute (time point).
    """
    # declarations start
    average_all_div = []
    average_def_div = []
    time_point = []
    initial = sum_of_time[0].split("-")
    def_per_file_mu = []
    # declarations end
    # for dataset start
    for i in range(len(sum_of_time)):
        per_file_mu = []
        spliced = sum_of_time[i].split("-")
        for num in range(4):
            if len(spliced[num]) == 1:
                spliced[num] = "0" + spliced[num]
        image = sum_of_divided_frames[i]
        time_point.append(analysis_functions.transition(spliced, initial))
        for dimensions_mu in dimensions_mu_array:
            def_img_mu                                                    =
np.average(sum_of_divided_frames_def[dimensions_mu[0]:dimensions_mu[1],
                                   dimensions_mu[2]:dimensions_mu[3]])
            img_mu          =          np.average(image[dimensions_mu[0]:dimensions_mu[1],
dimensions_mu[2]:dimensions_mu[3]])
            per_file_mu.append(img_mu)
            def_per_file_mu.append(def_img_mu)
            average_all_div.append(np.average(per_file_mu))
    # for dataset end

    # for default image start
    for dimensions_mu in dimensions_mu_array:
        def_img_mu                                                        =
np.average(sum_of_divided_frames_def[dimensions_mu[0]:dimensions_mu[1],
                             dimensions_mu[2]:dimensions_mu[3]])
        def_per_file_mu.append(def_img_mu)
    average_def_div.append(np.average(def_per_file_mu))

    avg_def = average_def_div  # np.array([average_def_div], dtype=np.float32)
    # for default image end

    avg_intensity, avg_time = analysis_functions.between(time_point, average_all_div, avg_def,
def_time)
    # plot start
    plt.plot(avg_time, avg_intensity, "ro-")
    plt.title("Mucosa/white")
    plt.xlabel("Time (min)")
    plt.ylabel("Haemoglobin Oxygenation (a.u.)")
    plt.title("Hemoglobin Oxygenation Dynamic Graph ")
    plt.xlim(-1, 40)
    plt.show()
    # plot end
```

project_functions.py

```
from pathlib import Path
import shutil

""" this .py file is containing functions used to process the projects"""



def write(path, name):
    """ it appends the text file projects.txt which is default and contains all the names of projects
that have
        been saved
    """
    file = open(path + "def_images//" + "projects.txt", "a")
    file.write(str(name) + "\n")
    file.close()
```

```
def read(path):
    """ it reads the text file and returns in a type of a list all the names of the projects """
    file = open(path + "def_images//" + "projects.txt", "r")
    x = file.readlines()

    for i in range(len(x)):
        x[i] = x[i].split("\n")[0]
    if len(x) == 0:
        x = ["        "]
    else:
        x = x
    file.close()

    return x



def delete_project(path, name):
    """ delete the name of the project from the projects.txt and
        delete the data and the folder of that project
    """
    x = read(path)
    x.remove(str(name))
    file = open(path + "def_images//" + "projects.txt", "w")

    for i in range(len(x)):
        file.write(x[i]+"\n")
    file.close()
    path_ori = (path+"projects//"+name).replace("//", "/")
    dir_path = Path(path_ori+"/")
    try:
        shutil.rmtree(dir_path)
    except OSError as e:
        print("Error: %s : %s" % (dir_path, e.strerror))
        print("if the project folder was not created, there is no error")
```

selection_functions.py

```
import os
from skimage import metrics
import numpy as np
import cv2
import functions_crop
import general_functions



def metrics_func(sorted_table, type_cr, f, selected, filter_wave, path):
    """ metric calculation and saving the results """
    metric = []
    if not os.path.exists(path + "metrics"):
        os.makedirs(path + "metrics")
    for i in range(len(sorted_table)):
        metric.append(int(metrics.peak_signal_noise_ratio(selected, sorted_table[i][filter_wave],
                                                          data_range=None)))
    metric = np.array(metric)

    file = open(path + "metrics//" + str(f) + type_cr, "w")
    file.write(str(metric))
    file.close()

    return metric



def metrics_func_quick(sorted_table, selected, filter_wave):
    """ metric calculation and saving the results """
    metric = np.zeros(len(sorted_table), dtype=int)
    small_sum = 0
    for i in range(len(sorted_table)):

        metric[i] = int(metrics.peak_signal_noise_ratio(selected, sorted_table[i][filter_wave],
                data_range=None))
        small_sum = small_sum + metric[i]
    return metric, small_sum


def save_all(sum_of_all_frames, sum_of_all_time, path):
    """ save the selected frames """
    sum_of_all_frames = np.array(sum_of_all_frames)
    sum_of_all_time = np.array(sum_of_all_time)
    new_path = path + 'selected'
    sorted_table, table1d = functions_crop.table()
    if not os.path.exists(new_path):
        os.makedirs(new_path)
    else:
        general_functions.remove_files(path + "selected//")
    for i in range(len(sum_of_all_frames)):
        for j in range(len(sum_of_all_frames[0])):
            cv2.imwrite(path + "selected//" + sum_of_all_time[i] + "_" +
                str(sorted_table[j]) + '.png', sum_of_all_frames[i][j])


def pre_process(filename):
    """ the  pre-process of the frames and time so that we can process them and analyze them later
on the program """
    sum_of_frames, sum_of_time = general_functions.single_video_opening(filename)
    frames = np.array(sum_of_frames)
    cropped_frames = np.array(functions_crop.initial_crop(frames))
    square_cropped_frames = np.array(functions_crop.square_crop(cropped_frames))
    final_cropped_frames = np.array(functions_crop.final_crop(square_cropped_frames))
    cube = functions_crop.pattern(final_cropped_frames)
    sorted_table = functions_crop.sorted_cube(cube)

    return sorted_table, sum_of_time



def pre_process_select_image(filename):
    """ the  pre-process of the saved_image so that we can process it and analyze its later on the
program """
    video = general_functions.file_opening_selection(filename)
    frames = np.array(video)
    cropped_frames = np.array(functions_crop.initial_crop(frames))
    square_cropped_frames = np.array(functions_crop.square_crop(cropped_frames))
    final_cropped_frames = np.array(functions_crop.final_crop(square_cropped_frames))

    return final_cropped_frames
```