TECHNICAL UNIVERCITY OF CRETE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
ELECTRONICS LABORATORY

TECHNICAL UNIVERSITY OF CRETE

Development of a Software Platform for the Analysis of Hyperspectral Data

By

Spyros Pouros

Thesis Commitee

Costas Balas, Professor, Supervisor

Vasilis Samoladas, Assistant Professor

George Karystinos, Associate Professor

Chania, March 2019

Abstract

Hyperspectral imaging collects and processes information from across the electromagnetic spectrum. The goal of hyperspectral imaging is to obtain the spectrum for each pixel in the image of a scene, with the purpose of finding objects, identifying materials, or detecting processes. In hyperspectral imaging, the recorded spectra have fine wavelength resolution, cover a wide range of wavelengths and measures continuous spectral bands. The primary advantage to hyperspectral imaging is that, because an entire spectrum is acquired at each point, the operator needs no prior knowledge of the sample, and post processing allows all available information from the dataset to be mined. Hyperspectral imaging can also take advantage of the spatial relationships among the different spectra in a neighborhood, allowing more elaborate spectral-spatial models for a more accurate segmentation and classification of the image.

In this work, the goal is to create and develop a hyperspectral imaging processing suite which will be able to work with and provide to the user a spread set of tools so that hyperspectral images can be processed. It is about a framework highly equipped, a research tool that can be fully expandable. It is characterized by its compatibility, flexibility and execution speed. The suite provides a user interface which, compared to the other common suites on the market, is easier to work and has more distinguished and functional environment.

Acknowledgements

It is a great opportunity to bestow my heartful regards to all people who have been either directly or indirectly involved in the fullment of this diploma dissertation.

First and foremost, I would like to express my sincerest gratitude to my professor and supervisor, Professor Constantinos Balas, for giving me the opportunity to deal with such an interesting topic. Not only did he help me completing my studies, but also motivated me to work more efficiently and professionally by conducting a lot of extra research, being familiar with experimental devices and gaining valuable knowledge.

Besides my supervisor, I would like to thank the rest of my thesis commitee: Assistant Prof. Vasilis Samoladas and Associate Prof. George Karystinos, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

My sincere thanks also goes to the whole team of the Optoelectronics and Imaging Diagnostics Lab, Vardoulakis Emmanouil (M.sc Candidate), Tsapras Athanasios (PhD Candidate), Rossos Christos (PhD Candidate), Papathanasiou Athanasios (M.sc), Gkouzionis Ioannis (M.sc Candidate), Fragkoulis Logothetis (M.sc Candidate) and Anny Vastaroucha (M.sc Candidate) for their numerous brainstorming discussions, and useless advice.

Furthermore I would like to thank my best friends Vaggelis K., Adam, Vaggelis P., Giorgos, Mathios and Elena for the countless support in whatever decision I made and of course because they were by my side whenever I needed it.

Finally, I must express my very profound gratitude to my beloved parents, Giorgos and Maria for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The scope of this operation is to develop a suite which will be able to be competitive to other suites which are used in some similar areas. Nowadays there is a common interest in the fields of image processing and spectral imaging. More and more scientists, research groups and laboratories are heading to this areas and trying to develop several techniques in order to be competitive and ahead from the generation. Many challenges for a science field which applies in several areas like biomedical engineering, agriculture, astronomy etc.

So, listening to that concern it follows a framework, a research tool which comes to fill some gapes of the already existing hyperspectral suites and aspires to be competitive to them. In order to achieve these goals, there is a try of developing a suite which will be flexible, ease to be work with and will be distinguished to others in term of portability, execution speed and extensibility.

1.2 Thesis Outline

Chapter 2 provides a theoretical background about image processing and hyperspectral imaging.

Chapter 3 presents what exists nowadays and the main reasons why there was a specific interest to this suite development.

Chapter 4 provides the basic information about the workflow on which the suite was developed and also there are some information about the suite itself.

Chapter 5 provides general information about plugins and their implementation in this suite.

Chapter 6 presents the summary of what was analyzed before and provides some suggestions for future work.

Chapter 2
Theoritical Background

2.1 Imaging

Imaging is the visual representation or reproduction of an object. At this time, digital imaging is the most advanced and applicable method where data are recorded using a digital camera, such as a CMOS.

The amount of information that can be extracted from an image is determined by the quality of it. Image quality is determined by the following parameters:

- Spatial resolution determines the closest distinguishable features in an object. It depends mainly on the wavelength (λ), the numerical aperture (NA) of the objective lens, the magnification, and the pixel size of the array-detector. The last two play an important role because they determine the sampling frequency which must be sufficiently high to achieve full resolution. Spatial resolution also depends on the signal quality.
- Lowest detectable signal depends on the quantum efficiency of the detector (the higher the better), the noise level of the system (the lower the better), the NA (numerical aperture) of the optics (the higher the better), and the quality of the optics.
- Dynamic range of the acquired data determines the number of different intensity levels that can be detected in an image. It depends on the maximal possible number of electrons at each pixel and on the lowest detectable signal (basically it is the ratio of these two values). If, however, the measured signal is low, so that the CMOS well associated with a pixel is only partially filled, the dynamic range will be limited accordingly. As an example, if a CMOS well if fulfilled to only 10% of its maximum capacity, the dynamic range will be reduced to 10% of its nominal value.
- Field of view (FOV) determines the maximal area that can be imaged.

2.2 Image Processing

In computer science, **digital image processing** is the use of computer algorithms to perform image processing on digital images. It allows a wide range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

Digital image processing allows the use of much more complex algorithms, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means.

In particular, digital image processing is the only practical technology for:

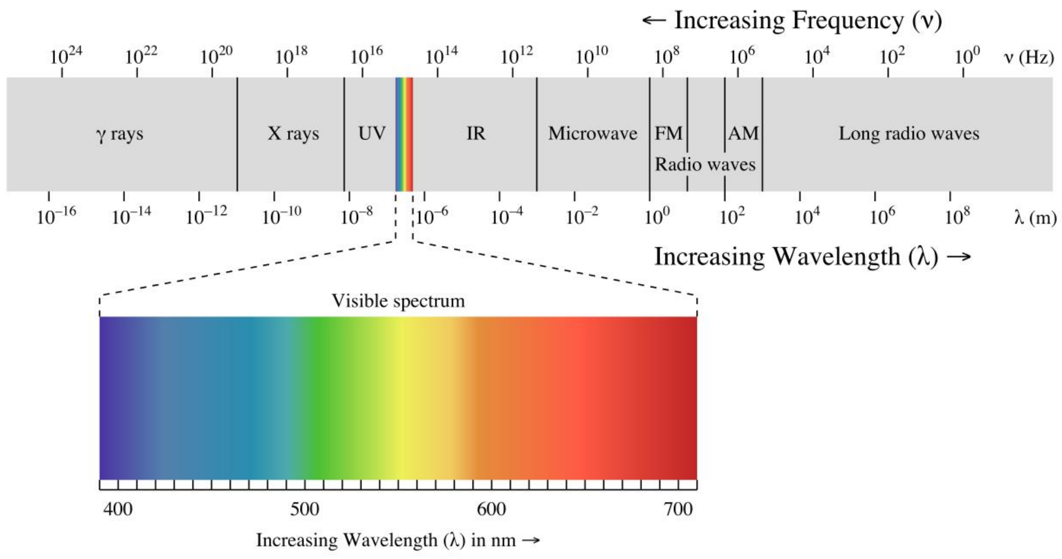
- Classification
- Feature Extraction
- Multi-scale signal extraction
- Pattern recognition

Some techniques which are used in digital image processing include:

- Image editing
- Image restoration
- Independent component analysis
- Linear filtering
- Neural networks
- Partial differential equations

2.3 Electromagnetic Spectrum

The electromagnetic spectrum is the complete range of wavelengths of electromagnetic radiation, beginning with the longest radio waves (including those in audio range) and extending through visible light (a very small part of the spectrum) all the way to the extremely short gamma rays that are a product of radioactive atoms. Nearly all types of electromagnetic radiation can be used for spectroscopy, to study and characterize matter.



← Increasing Frequency (ν)
10^24 10^22 10^20 10^18 10^16 10^14 10^12 10^10 10^8 10^6 10^4 10^2 10^0 ν (Hz)
γ rays
X rays
UV
IR
Microwave
FM
AM
Radio waves
Long radio waves
10^-16 10^-14 10^-12 10^-10 10^-8 10^-6 10^-4 10^-2 10^0 10^2 10^4 10^6 10^8 λ (m)
Increasing Wavelength (λ) →
Visible spectrum
400
500
600
700
Increasing Wavelength (λ) in nm →

FIGURE 1 ELECTROMAGNETIC SPECTRUM

The types of radiation are generally grouped by the kinds of chemical and physical effects they can produce on matter. For example, in a magnetic field, exposure to the low-energy radio frequency radiation only reorients nuclei, while exposure to the slightly higher-energy microwave region changes electron spin states of molecules with unpaired electrons. Microwave radiation can also change the rotational energy of molecules; this effect is used to heat food quickly in a microwave oven. In the middle regions of the electromagnetic spectrum, absorption of IR radiation causes changes in the vibrational energy of molecules. Visible (Vis) and ultraviolet (UV) radiations alter the electron energies of loosely held outer electrons of atoms and molecules. Higher-energy X-rays can cause electron transitions between inner electron levels, and gamma radiation produces changes within atomic nuclei. As all compounds absorb radiation in multiple regions of the spectrum, the information on molecular activity in each region provides complementary data for material characterization.

2.4 Range of the Spectrum

Electromagnetic waves are typically described by any of the following three physical properties: the frequency f, wavelength λ, or photon energy E. Wavelength is inversely proportional to the wave frequency, so gamma rays have very short wavelengths that are fractions of the size of atoms, whereas wavelength on the opposite end of the spectrum can be of thousand kilometers. Photon energy is directly proportional to the wave frequency, so gamma ray photons have the highest energy (around a billion electron volts), while radio wave photons have very low energy (approximately a femtoelectronvolt). These relations are illustrated by the following equations:

$f = \frac{c}{\lambda}$, or $f = \frac{E}{\lambda}$, or $E = \frac{h*c}{\lambda}$

The behavior of EM radiation depends on its wavelength. When EM radiation interacts with single atoms and molecules, its behavior also depends on the amount of energy per quantum (Photon) it carries.

2.5 Spectral Imaging (SI)

Spectral imaging is a combination of spectroscopy and photography in which a complete spectrum or partial spectral information (such as the Doppler shift or Zeeman splitting of a spectral line) is acquired at each position of an image plane. Spectral imaging allows extraction of additional information the human eye fails to capture with its receptors for red, green and blue. Applications related to astronomy, solar physics, analysis of plasmas in nuclear fusion experiments, planetology, and Earth remote sensing are sparked by the benefits of spectral imaging.

Various distinctions among techniques are applied, based on criteria including spectral range, spectral resolution, number of bands, width and contiguousness of bands, and application. The terms include Multi-Spectral Imaging, Hyper-Spectral Imaging, full spectral imaging, imaging spectroscopy or chemical imaging.

Important new developments in the field of Biomedical Optical Imaging (OI) allow for unprecedented visualization of tissue microstructure and enable quantitative mapping of disease-specific endogenous and exogenous substances. Spectral imaging (SI) is one of the most promising OI modalities.

A spectral imager provides spectral information at each pixel of an image sensor array. The SI Systems acquire a three-dimensional (3D) data set of spectral and spatial information, known as spectral cube. The spectral cube can be considered as a stack of images, each of them acquired at a different wavelength. Combined spatial and spectral information offers great potential for the non-destructive/invasive investigation of a variety of studied samples.

2.5.1 Multi-Spectral Imaging

Multi-Spectral Imaging (MI) is responsible for capturing image data at specific frequencies across the electromagnetic spectrum. The wavelengths may be separated by filters or by the use of instruments that are sensitive to particular wavelengths, including light from frequencies beyond the visible light range, such as infrared. MI images are the main type of images acquired by remote sensing (RS) radiometers. Dividing the spectrum into many bands, MI is the opposite of panchromatic, which records only the total intensity of radiation falling on each pixel. Spectral Imaging with more numerous bands, finer spectral resolution or wider spectral coverage may be called Hyper-Spectral or Ultra-Spectral.

There are several fields in which Multi-Spectral Imaging is able to be applied. Some of these are:

- Military Target Tracking
- Land Mine Detection
- Ballistic Missile Detection
- Space-based imaging
- Documents and artworks

2.5.2 Hyper-Spectral Imaging

Hyperspectral imaging (HSI) is an emerging field in which the advantages of optical spectroscopy as an analytical tool are combined with two-dimensional object visualization obtained by optical imaging. In HSI, each pixel of the image contains spectral information, which is added as a third dimension of values to the two-dimensional spatial image, generating a three-dimensional data cube, sometimes referred to as hypercube data or as an image cube. A simple, well-known example of a three-dimensional data cube is the common RGB color image, where each pixel has red, green, and blue color. Hyperspectral data cubes can contain absorption, reflectance, or fluorescence spectrum data for each image pixel. It is assumed that HSI data is spectrally sampled at more than 20 equally distributed wavelengths. The spectral range in hyperspectral data can extend beyond the visible range (ultraviolet, infrared). Multispectral imaging (MSI) is a term that should probably be reserved for imaging that simultaneously uses two or more different spectroscopy methods in the imaging mode (eg, wavelength and fluorescence lifetime). The result of imaging with a couple of color bands/filters should not be termed spectral imaging, much less multispectral, but unfortunately these terms have been somewhat misused, especially in the biological literature. With these reservations, in the following review MSI systems typically record images at fewer than 20 wavelengths and can include noncontiguous as well as wide and narrow spectral bands.

HSI applications in industry (machine vision) and remote sensing (including satellite reconnaissance) are relatively well-known because the technique was originally defined by Goetz in the late 1980s. However, within the past decade, a surge of interest in HSI technology has been seen in life sciences with applications in fields as diverse as agriculture, food quality and safety, pharmaceuticals, and particularly in healthcare.

The surface of the body is an excellent area for deployment of optical research methods, and HSI technology is being applied in ever more applications in dermatology for noninvasively targeting cancer detection, skin oxygenation mapping for diabetic ulcers, spectral unmixing of fluorescently labeled antigens, and more. In this chapter, we will introduce the tissue optics principles used for hyperspectral skin imaging. Different spectral imaging technologies for a variety of skin imaging applications will be described.

The several application in which we can use HIS are:

- Agriculture
- Eye care
- Food Processing
- Mineralogy
- Surveillance
- Astronomy
- Chemical Imaging
- Environment

2.6 Spectral Cube

The information that is primarily collected by spectral imagers and then appropriately processed based on the kind of application running, is stored in 3D data structures for further analysis. This sort of data structures are known as Spectral Cubes (SC). A spectral cube consists of the three dimensional projection of a great number of consecutive and registered sets of hyper-spectral or multi-spectral images. Being more specific, the first two dimensions respond to spatial dimensions, for account of pixel coordinates and the third one refers to spectral dimension, meaning a specific wavelength of the electromagnetic spectrum. A glance at Figure 2.4 can offer profound perception of how a spectral cube does look like.

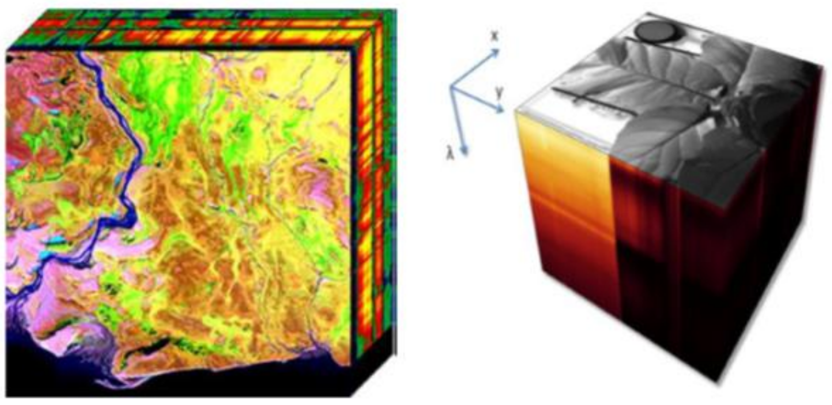


FIGURE 2 HYPER-SPECTRAL CUBES

Simply put, an imaging spectrometer acquires the spectrum of each pixel in a two-dimensional spatial scene. As shown in the figure below, the easiest way to think of such a scheme is as band sequential imaging, in which multiple images of the same scene at different wavelengths are acquired. A key point is that the spectra be sampled densely enough to reassemble a spectrum (commensurate with the need for analysis). There are many technological means to obtain these data. The images are typically stacked in a computer, from the lowest wavelength to the highest, to create an image cube of the data set. The spectrum of a selected pixel is obtained by skewering it in its third dimension, wavelength, as the inset in Figure 2.4 shows.

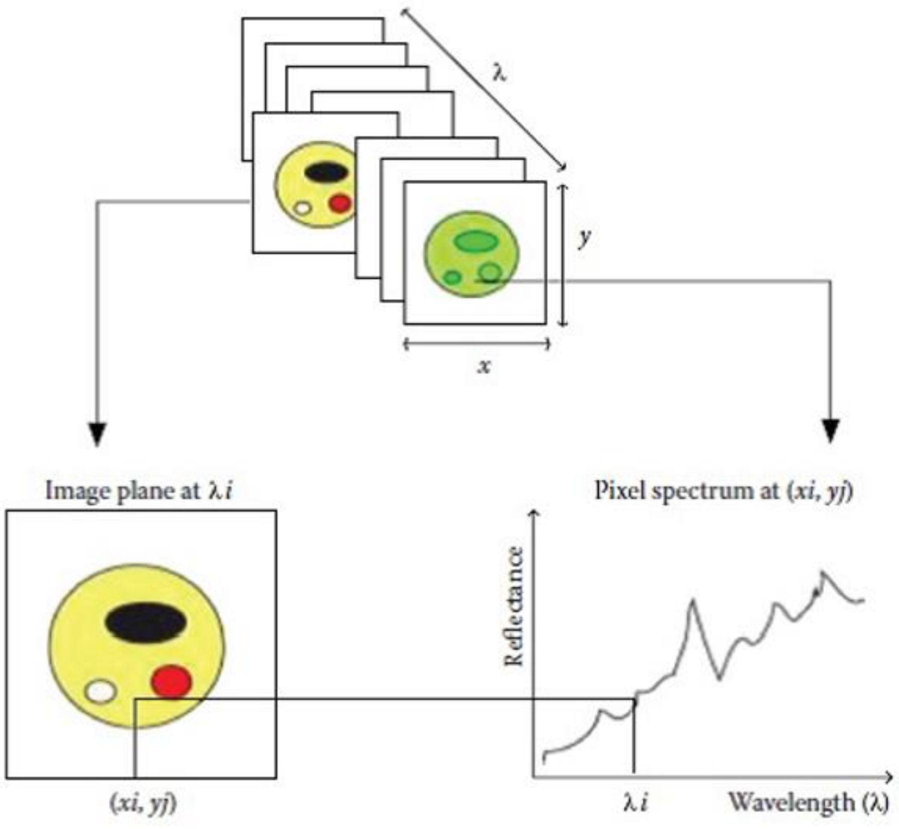


FIGURE 3 SCHEMATIC REPRESENTATION OF A HYPERCUBE SHOWING THE RELATIONSHIP BETWEEN SPATIAL AND SPECTRAL DIMENSIONS.

As an example, the hyper-spectral cube of a fish fillet acquired using reflectance mode is illustrated in Figure 2.5. The raw hyper-spectral cube consists of a series of contiguous sub-images one behind each other at different wavelengths Figure 2.5.a. Each sub-image provides the spatial distribution of the spectral intensity at a certain wavelength. That means a hyper-spectral image described as I(x; y; λ) can be viewed either as a separate spatial image I(x; y) at each individual wavelength (λ), or as a spectrum I(λ) at each individual pixel (x; y). From the first viewpoint, any spatial image within the spectral range of the system can be picked up from the hyper-spectral cube at a certain wavelength within the wavelength sensitivity Figure 2.5.b. The gray scale image shows the different spectral intensity of the imaged object at a certain wavelength due to the distribution of its corresponding chemical components. For example, an image within the hypercube at a single waveband centered at 980 nm with bandwidth of 5 nm Figure 2.5.b can relatively show the information of moisture distribution in the fish fillet, which is difficult to be observed in RGB image Figure 2.5.c. From the second viewpoint, the resulting spectrum of a certain position within the specimen can be considered as its own unique spectral fingerprint of this pixel to characterize the composition of that particular pixel Figure 2.5.d.

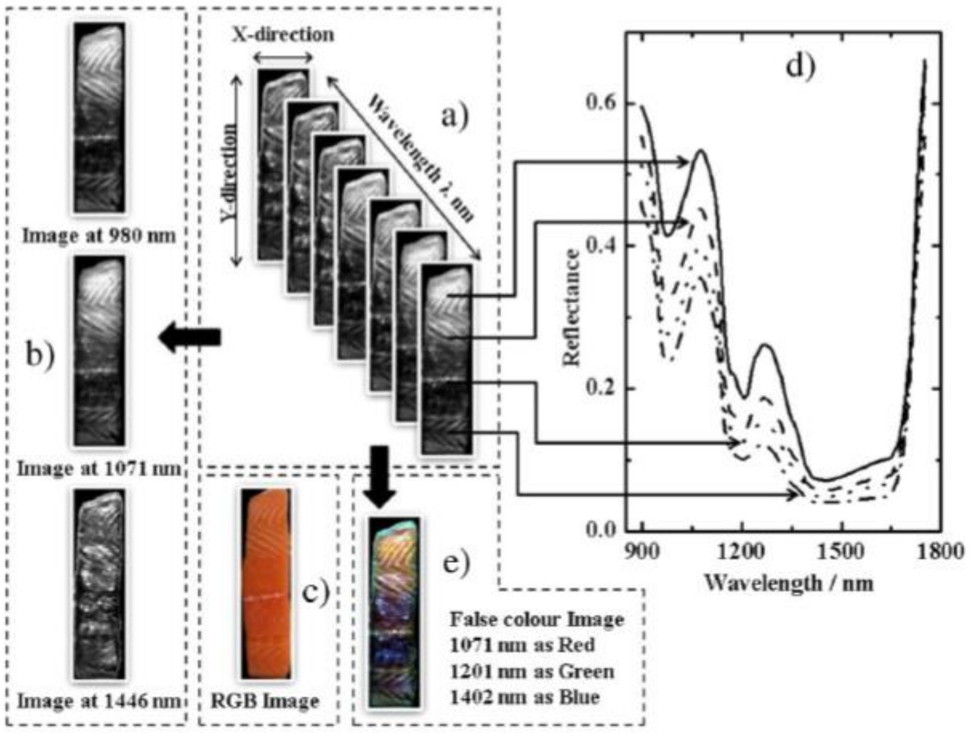


FIGURE 4 SCHEMATIC DIAGRAM OF HYPER-SPECTRAL IMAGE (HYPER-SPECTRAL CUBE) FOR A PIECE OF ﬁSH ﬁLLET

2.7 Hyper – Spectral Imaging Applications

As it has already been mentioned, there are numerous applications emerging from the spectral analysis that is being provided by hyper-spectral imaging. For nearly a decade, this technology was primarily used for purposes like surveillance, reconnaissance, environmental and geological studies. However, the application of hyper-spectral imaging in the biomedical area has been negligible due to high-instrumentation costs and problems arising from the clinical use of hyperspectral sensors. With recent achievements in sensor technology and increasing affordability of high performance spectral imagers, hyper-spectral imaging systems constitute one of the most important key areas in medical imaging. The early diagnosis of cancer, one of the thorniest medical problems, is now possible, since the evolution of hyper-spectral sensors allows the scanning of a patient's body to identify precancerous lesions or to provide critical spectral data through endoscopic procedures. The extension and improvement of hyper-spectral imaging in biomedical and clinical diagnosis is within the grasp of researchers. The advantages of this technology regarding diagnostic health care applications include a high resolution imaging of tissues either at macroscopic or cellular levels and the capability to generate highly accurate spectral information related to the patient, tissue sample, or any other disease condition. In particular, the vast investment of hyper-spectral imaging in medicine lies on the generation of wavelength-specific criteria for disease conditions on spectral features. As a consequence, an ideal technology for high-throughput patient screening and non-invasive diagnosis is begotten.

Due to their unparalleled ability to reveal abnormal spectral signatures, hyper-spectral medical instruments hold great potential for non-invasive diagnosis of cancer, retinal abnormalities and assessment of wound conditions, for instance diabetes. A portable hyper-spectral imager could also aid the analysis of human body fluids, such as blood, urine, saliva, semen and determine blood oxygenation levels of tissues, which could be of prime importance during surgeries. Yet importantly, it could perform diagnosis for dental diseases. It is a great advantage for a patient the fact that not only does an early diagnosis of an ailment can take place, but also an appropriate treatment may be applied at the same time.

Hyper-spectral signatures when combined with targeting algorithms would in essence offer unique diagnostic information. There is an increasing level of interest on the part of health care providers to investigate possible ways of reducing health care costs by providing timely treatments for many types of disease conditions. Hyper-Spectral scanning imaging is expected to contribute a lot in this pursuit.

2.8 Measures of Spectral Similarity

The similarity measure is the measure of how much alike two data objects are. Similarity measure in a data mining context is a distance with dimensions representing features of the objects. If this distance is small, it will be the high degree of similarity where large distance will be the low degree of similarity.

The similarity is subjective and is highly dependent on the domain and application. For example, two fruits are similar because of color or size or taste. Care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation. Similarity are measured in the range 0 to 1 [0,1].

In statistics and more generally in common spectral imaging suites the most well – known and more often used techniques for spectral similarity measuring are:

- Euclidean Distance: is the most common use of distance. In most cases when people said about distance, they will refer to Euclidean distance. Euclidean distance is also known as simply distance. When data is dense or continuous, this is the best proximity measure. The Euclidean distance between two points is the length of the path connecting them. The Pythagorean Theorem gives this distance between two points.
- Manhattan distance: is a metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates. In a simple way of saying it is the total sum of the difference between the x-coordinates and y-coordinates. Suppose we have two points A and B if we want to find the Manhattan distance between them, just we have, to sum up, the absolute x-axis and y – axis variation means we have to find how these two points A and B are varying in X-axis and Y- axis. In a more mathematical way of saying Manhattan distance between two points measured along axes at right angles. In a plane with p1 at (x1, y1) and p2 at (x2, y2). Manhattan distance = $|x1 - x2| + |y1 - y2|$. This Manhattan distance metric is also known as Manhattan length, rectilinear distance, L1 distance or L1 norm, city block distance, Minkowski's L1 distance, taxi-cab metric, or city block distance.
- Cosine similarity: finds the normalized dot product of the two attributes. By determining the cosine similarity, we would effectively try to find the cosine of the angle between the two objects. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90°

have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors.

- SID: is a measure derived from spectral information measure which models the spectral band-band variability as a result of uncertainty caused by randomness. The SID is derived from divergence theory and calculates the probabilistic behaviors between spectral signatures. Compared with SAM, which examines the geometrical characters between two spectral signatures or pixel vectors, SID computes the discrepancy between the probability distributions produced by the spectral signatures. Consequently, SID is more effective than SAM incapturing the subtle spectral ariability.
- SAM: is a popular and widely used spectral similarity measure in hyperspectral remote sensing. It calculates spectral similarity by measuring the angle between the spectral signature of two samples, I's and j's The measure determines the similarity between two spectra by calculating the spectral angle between them, treating them as vectors in a space with dimensionality equal to the number of spectral bands used (Kruse et al 1997). The spectral angle has a lower bound of 0 and has values always greater than 1. Unlike the distance metrics it is possible to have a zero spectral angle even when the two vectors are not identical. This technique is relatively insensitive to illumination and albedo effects because the angle between two vectors is invariant with respect to the length of the vectors.

Chapter 3

Creating a Spectral Suite

3.1 Purpose of creating/using a spectral suite

While multispectral images have been in regular use since the 1970s, the widespread use of hyperspectral images is a relatively recent trend. Hyperspectral imaging, also known as imaging spectrometry, is now a reasonably familiar concept in the world of remote sensing. However, for many biomedical specialists who have not had the opportunity to use hyperspectral imagery in their work yet, the benefits of hyperspectral imagery may still be vague and there is no specialized tool on the market nowadays to work on it. Undoubtedly, it cannot be specified that it is vitally important the existence of a hyperspectral suite which will provide specific tools and features appropriate for biomedical imaging and engineering.

3.2 Present Spectral Suites

Nowadays, there are some spectral suites on the market. ENVI, MultiSpec, Geomatica, Erdas Image are only a few of them. Although, the fact that it is very interesting for new scientists to lead to that field of science and way of data analysis, it is worth to point out that none of them, or only a small number, is specialized in biomedical data analysis. The majority of them are used in agriculture fields and geographical analysis. These suites have many features to analyze and to edit spectral images and spectral cubes. Although, the occurrence of these spectral suites, it is obvious that there is a lack of a framework suitable to biomedical needs. The referred suites provides no tools for biomedical research, so these field's specialists are facing difficulties to using these suites. Furthermore, the existing suites use some old fashioned GUIs which create more difficulties to the user.

3.3 Biomedical Applications in Electronics Laboratory

Undoubtedly, electronics laboratory has a specific interest in biomedical applications. And for that reasons, many attempts were made in order to develop a spectral suite which be able to be used in spectral data analysis. The effort ,that has been put on that direction, has purpose to create a universal suite which will be able, in the future, to be 100%

functional. Hematology, skin cancer and plenty of others analysis are topics in which our laboratory is interested in so there was a need of such a suite development.

3.4 Innovations of presented thesis

As I mentioned before, from a large and extended research in the field of spectral suites, it is obvious that the majority of the well-known and most popular suites for spectral data analysis are specialized in geographical and agriculture fields. Because of that, many features from these suites are not specialized for bio data. So, it is important to develop a platform which will be able to handle and analyze spectral data from biomedical staff. Except for that, this suite has the advantage of compatibility. Its design gives the opportunity of anyone else wants to use and add any feature, to follow the plugin pattern I made and finally add anything he wants. It is a very easy and useful way to expand this work, make it bigger and even more functional in order to be equal or even better compared to what is used until now. At last but not at least, after the research and checking the user interface from all of the platforms I refed before, obviously there is a need of change how UIs are set and how they looks like to the user. For sure, the environment in which someone works, is important to be friendly and considering that nowadays there are programs able to be used and design your UI quite more presentable and good-looking, it is imperative to use any access to do so. This suite is so easy to be used and gives the user the opportunity to handle all the information which extracts from the data he inserts.

Chapter 4

Basic Information about workflow

4.1 Introduction

Starting with the basic points of how I built this suite, it is very important to explain the tools which I used to construct the suite.

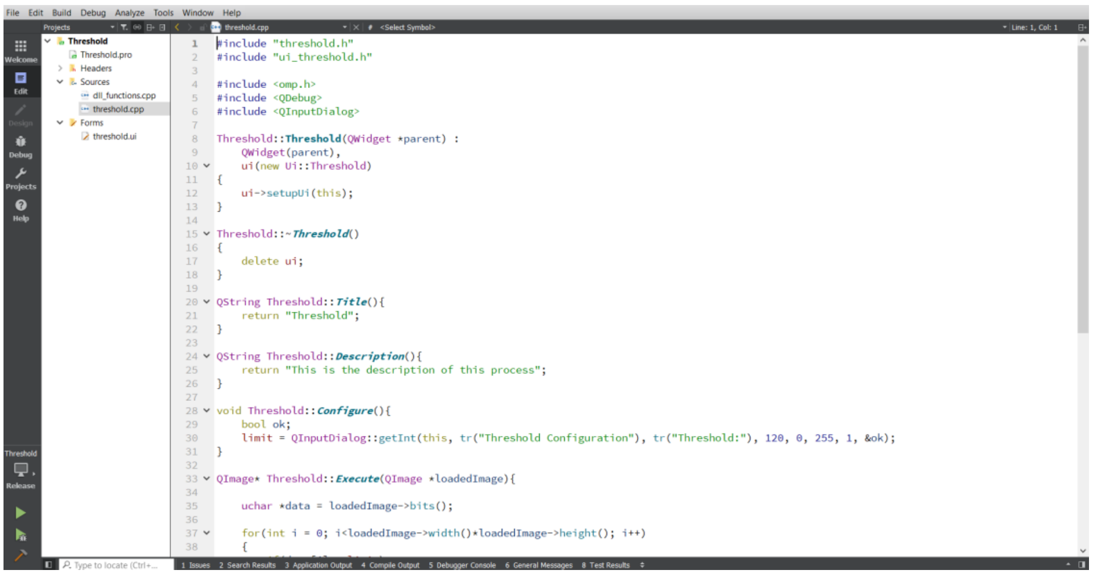
4.1.1 Qt

Qt is much more than just a cross-platform SDK - it's a technology strategy that lets you quickly and cost-effectively design, develop, deploy, and maintain software while delivering a seamless user experience across all devices.

Too many people, young scientists and pro engineers decides to go with Qt for 3 very specific reasons. The first one is that it’s fast. In software development, time really is money. That’s why Qt gives you a highly productive framework complete with cross-platform libraries, APIs and tools for faster time to market. Second one it’s easy. It is very important that with Qt’s easy – to – use and flexible IDE and design tool include ready-made controls and out-of-the box functionality for efficient UI design using drag and drop tools, declarative programming with QML or imperatively with c++. At last but not at least, it’s definitely future-proof. Due to requirements changes from time to time, Qt’s open extensible and modular c++ framework supports a cost-efficient software development life cycle.

In the end, Qt has many fields in which is already used. Some of them are shown below:

- Embedded Devices
- Application UIs and Software
- Internet of Things
- Mobile
- Automotive
- Automation
- Medical



```cpp
#include "threshold.h"
#include "ui_threshold.h"

#include <omp.h>
#include <QDebug>
#include <QInputDialog>

Threshold::Threshold(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Threshold)
{
    ui->setupUi(this);
}

Threshold::~Threshold()
{
    delete ui;
}

QString Threshold::Title(){
    return "Threshold";
}

QString Threshold::Description(){
    return "This is the description of this process";
}

void Threshold::Configure(){
    bool ok;
    limit = QInputDialog::getInt(this, tr("Threshold Configuration"), tr("Threshold:"), 120, 0, 255, 1, &ok);
}

QImage* Threshold::Execute(QImage *loadedImage){

    uchar *data = loadedImage->bits();

    for(int i = 0; i<loadedImage->width()*loadedImage->height(); i++)
    {
```

FIGURE 5 QT'S WORKFLOW

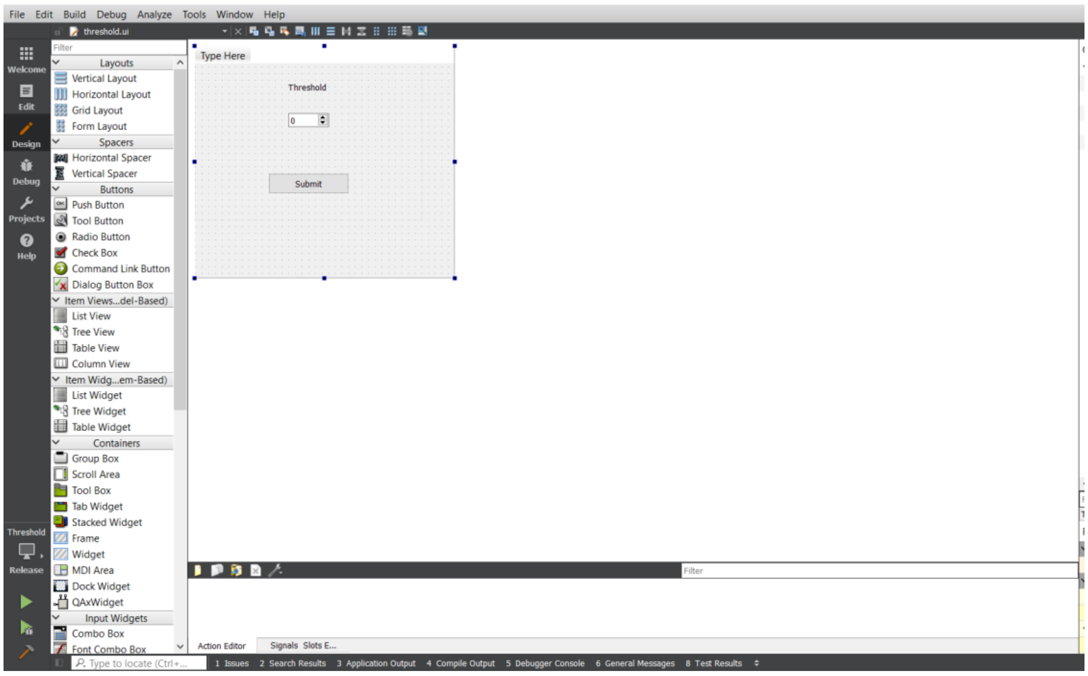


FIGURE 6 GUI'S DESIGNER

4.1.2 About Compiler and MinGW

MinGW (*Minimalist GNU for Windows*), formerly mingw32, is a free and open source software development environment to create Microsoft Windows applications. The development of the original MinGW project was halted in 2013, but an alternative called MinGW-w64 has been created by a different author to include several new APIs and provide 64-bit support.

MinGW includes a port of the GNU Compiler Collection (GCC), GNU Binutils for Windows (assembler, linker, archive manager), a set of freely distributable Windows specific header files and static import libraries which enable the use of the Windows API, a Windows native build of the GNU Project's GNU Debugger, and miscellaneous utilities.

MinGW does not rely on third-party C runtime dynamic-link library (DLL) files, and because the runtime libraries are not distributed using the GNU General Public License(GPL), it is not necessary to distribute the source code with the programs produced, unless a GPL library is used elsewhere in the program.[3]

MinGW can be run either on the native Microsoft Windows platform, cross-hosted on Linux (or other Unix), or "cross-native" on Cygwin.

4.1.3 About language and C++

C++ is widely used among the programmers or developers mainly in an application domain. It contains the important parts including the core language providing all the required building blocks including variable, data types, literals etc. It supports object-oriented programming including its features like Inheritance, Polymorphism, Encapsulation, and Abstraction. These concepts make the C++ language different and mostly in use for developing the applications easily and conceptualized.

There are several benefits of using C++ for developing applications and many applications product based developed in this language only because of its features and security. Please find the below sections, where uses of C++ has been widely and effectively used. Below is the list of top 10 uses of C++:

- Applications: It is used for development of new applications of C++. The applications based on graphic user interface, which are highly used applications like adobe photoshop and others. Many applications of Adobe systems are developed in C++ like Illustrator and image ready and Adobe

- Games: This language is also used for developing games. It overrides the complexity of 3D games. It helps in optimizing the resources. It supports multiplayer option with networking. uses of C++ allows procedural programming for intensive functions of CPU and to provide control over hardware, and this language is very fast because of which it is widely used in developing different games or in gaming engines. C++ mainly used in developing the suites of a game tool.
- Web Browser: This language is used for developing browser's as well. C++ is used for making Google Chrome, and Mozilla Internet browser Firefox. Some of the applications are written in C++, from which Chrome browser is one of them and others are like a file system, the map reduces large cluster data processing. Mozilla has other application also written in C++ that is email client Mozilla Thunderbird. C++ is also a rendering engine for the open source projects of Google and Mozilla.
- Database Access: This language is also used for developing database software or open source database software. The example for this is MySQL, which is one of the most popular database management software and widely used in organizations or among the developers. It helps in saving time, money, business systems, and packaged software. There are other database software access based applications used that are Wikipedia, Yahoo, youtube etc. The other example is Bloomberg RDBMS, which helps in providing real-time financial information to investors. It is mainly written in C++, which makes database access fast and quick or accurate to deliver information regarding business and finance, news around the world.

- **Compilers:** Most of the compilers mainly written in C++ language only. The compilers that are used for compiling other languages like C#, Java etc. mainly written in C++ only. It is also used in developing these languages as well as C++ is platform independent and able to create a variety of software.
- **Operating Systems:** It is also used for developing most of the operating systems for Microsoft and few parts of Apple operating system. Microsoft Windows 95, 98, 2000, XP, office, Internet Explorer and visual studio, Symbian mobile operating systems are mainly written in C++ language only.
- **Other Uses**: it is used for medical and engineering applications, Computer-aided design systems. These applications are like MRI scans machines, CAM systems that are mainly used in hospitals, local, state and national government, and other departments for construction and mining etc. applications of C++ is considered as a first preferred language to use among the developer when performance is considered for any developing application.

In conclusion, C++ is the language which is used everywhere but mainly in systems programming and embedded systems. Here system programming means for developing the operating systems or drivers that interface with Hardware. Embedded system means things that are automobiles, robotics, and appliances. C++ is having higher or rich community and developers, which helps in the easy hiring of developers and online solutions easily. Uses of C++ is referred to as the safest language because of its security and features. It is the first language for any developer to start, who is interested in working in programming languages. It is easy to learn, as it is pure concept based language. Its syntax is very simple, which makes it easy to write or develop and errors can be easily replicated. Before using any other language, programmers preferred to learn C++ first and then they used other languages. But most of the developers try to stick with C++ only because of its wide variety of usage and compatibility with multiple platforms and software.

4.2 Starting with the suite

The meaning of spectral cube’s was analyzed before. The suite uses XML files to extract information about images and the cube itself. So from the UI, user is able to select an XML file which is contained in the folder with the pictures.

The reason why we use XML files is that, it is a well-known and easy to be used markup language, which has native support from Qt. That’s why it is easy to save cube’s information in a text-based format which is readable both from algorithms and from developers and people themselves. Furthermore, it is very easy more information to be added and stored in those files with no need of code fixing.

Reading now the XML file, I extract information which are used subsequently in order to be easier for the user to handle the cube. The information that be extracted are

- Wavelengths
- The first band - wavelength
- The last band - wavelength
- Band Step
- The date when the cube was last modified
- And of course the name of all the images which will be loaded

The information, I just mentioned to, are available after the import in an action called “Cube Information” so that user will be able to know about them.

By reading the XML file, and keep the information to which I just referred to, I am able to access the cube, via images of course. After a set the images and the information a get from them (First/Last Band, Step etc) I keep the images in a QList in order to be able to process them and apply any filter, feature or clustering algorithm the suite gives me the ability to do so. After this process, UI has in the central Widget the mid Image from the cube. Right down from the image that is shown, there is label in which the wavelength in “nm” is mentioned.

As you see in the following image, there is a slider at the bottom of the main window. Sliding it, gives you the opportunity to move in every other image, from the loaded ones. The wavelength, I mentioned before, of the image you chose is updated. On the other hand if you just want to change the images one by one and not in a wider number, there are two arrow, one on the left and one on the right bottom side. After clicking the right one, for example, you move to the very image which's wavelength is up by step (The difference between two wavelengths, given that the difference among them is the same).

Since the cube is imported, images and information are kept in variables and arrays the image that is currently shown in the central widget, is always fit in the Graphics view widget so that the user don't miss any point of view.

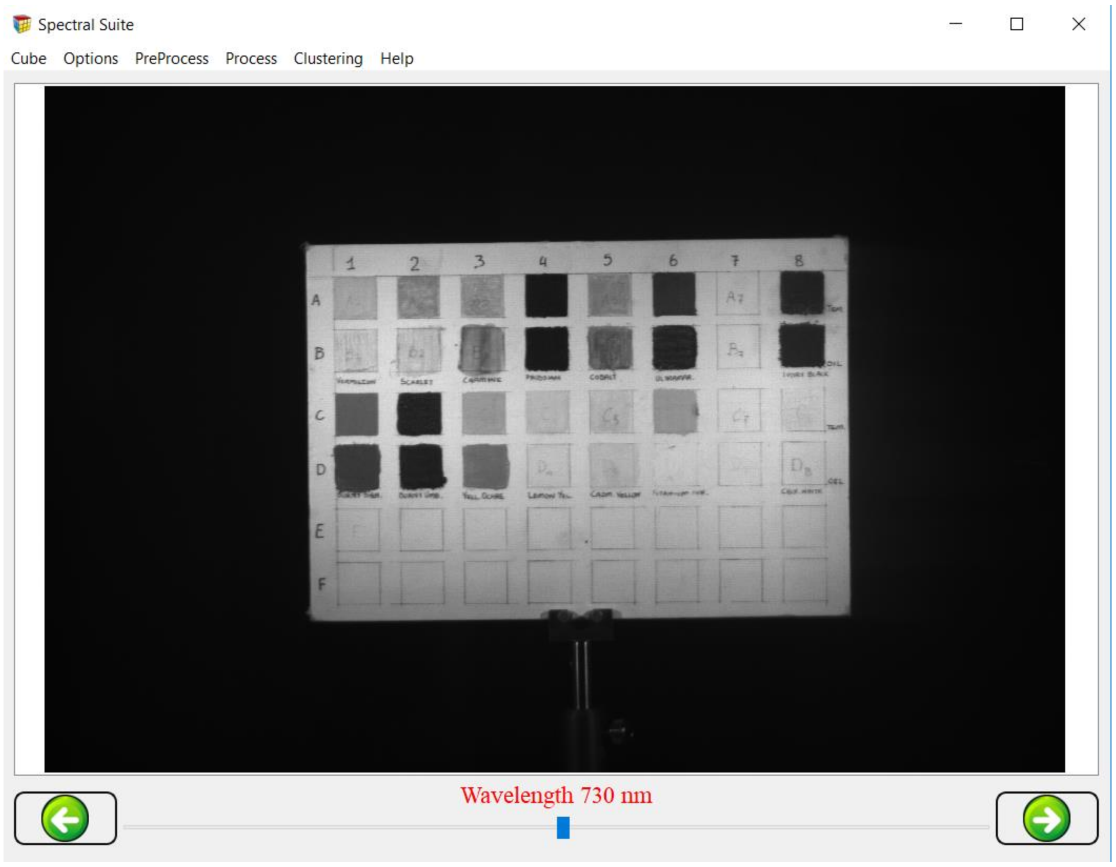


FIGURE 7 HYPERSPECTRAL SUITE GUI

4.3 Menu Bar and Options

The menu bar that is put on the top of the main window is developed in two separated parts. The first one is the branch that is created in a static way using Qt features. The options which are created apriori are:

1. The option “Cube”. This option has the following actions:
    - Import: Triggering this action a window is revealed, giving user the right to find the XML he wants and loaded in order to images and information be parsed.
    - Cube Information: At first this action is disabled. After a cube is imported, I set this action enabled and after the user clicks on it, a pop up window shows providing cube’s information about first and last band, step and date when the cube was last modified
    - Recent Files: In case this option is hovered by the cursor, a side list is revealed which contains the XML files from the last 5 cubes were imported
    - Export Cube: Clicking this button, a dialog window is opened so that the user is able to export, to save the whole modified cube after all the features he used during the suite explore.
    - Save: By triggering this action, a dialog window opens so that user can save the current image which is shown in the central widget.
    - Exit: At last, when the user clicks this button, he exits from the application at all.

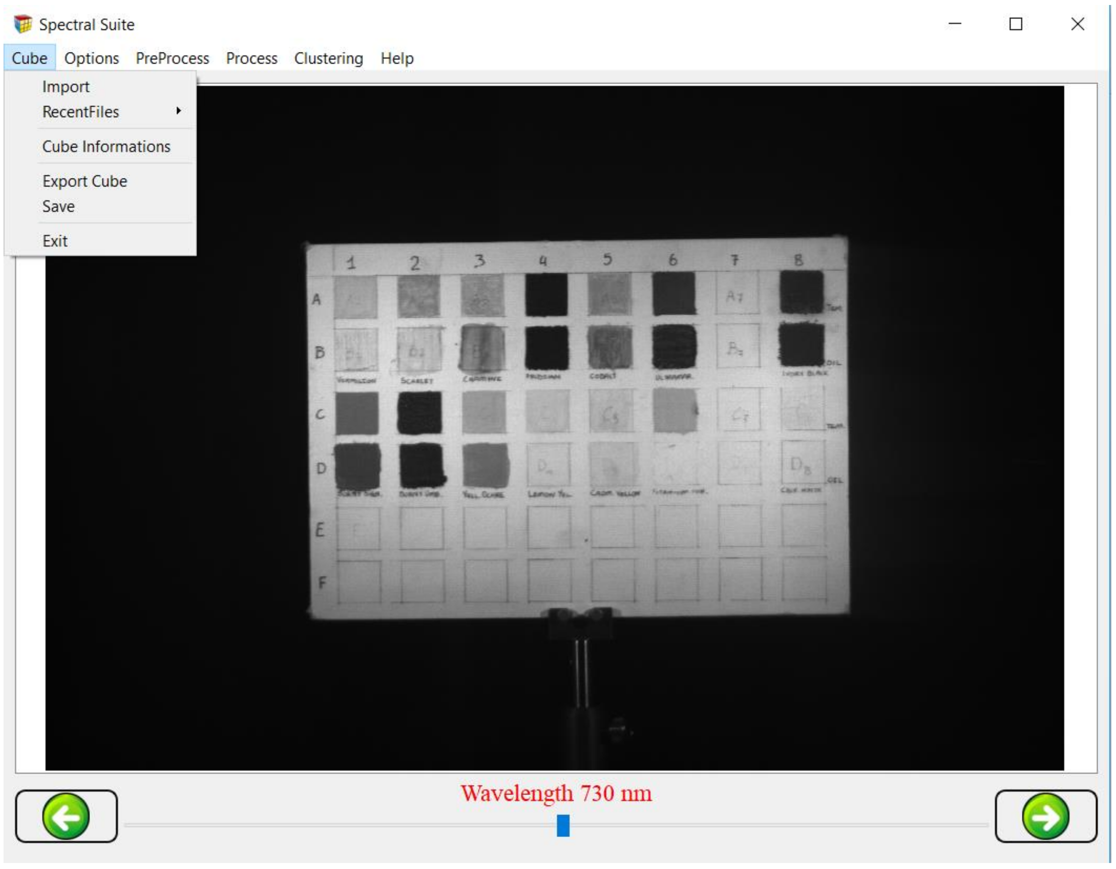


FIGURE 8 MENU OPTION "CUBE"

2. The option named “Options”. Here there are some options, with which the user is able to see the behavior either from the current image or the whole cube using plotting or histogram features, and edit the image by cropping, rotate etc. This option has the following actions:
   - **Plotting:** By triggering this option and hovering his cursor over the current image is able to see the graph which in xAxis has the wavelengths of the cube from first to last band by step, and in yAxis the intensities of the pixels. By moving the cursor over the image the graph updates its graph.

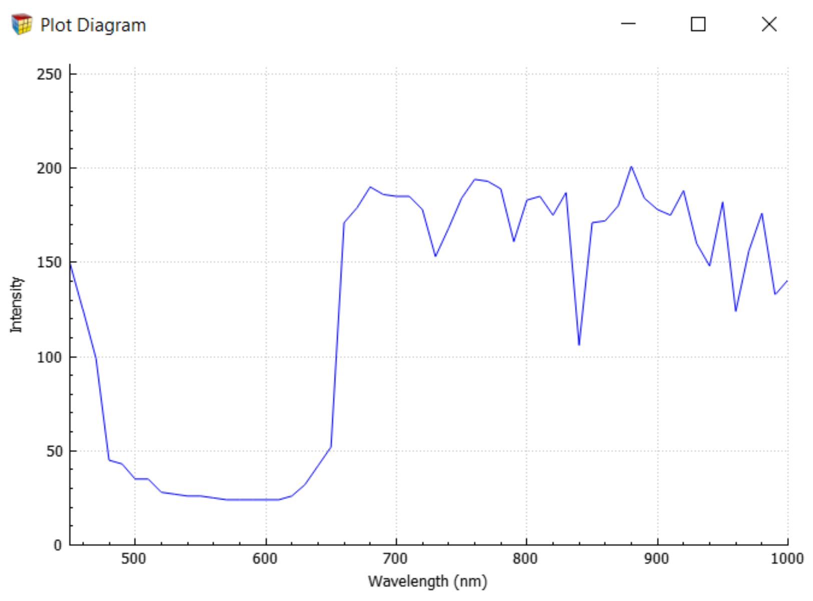


FIGURE 9 SPECTRUM PIXEL PLOT

- **Histogram:** This option creates a graph only for the current image and its purpose is to show how many pixels are in each intensity value from 0 to 255.

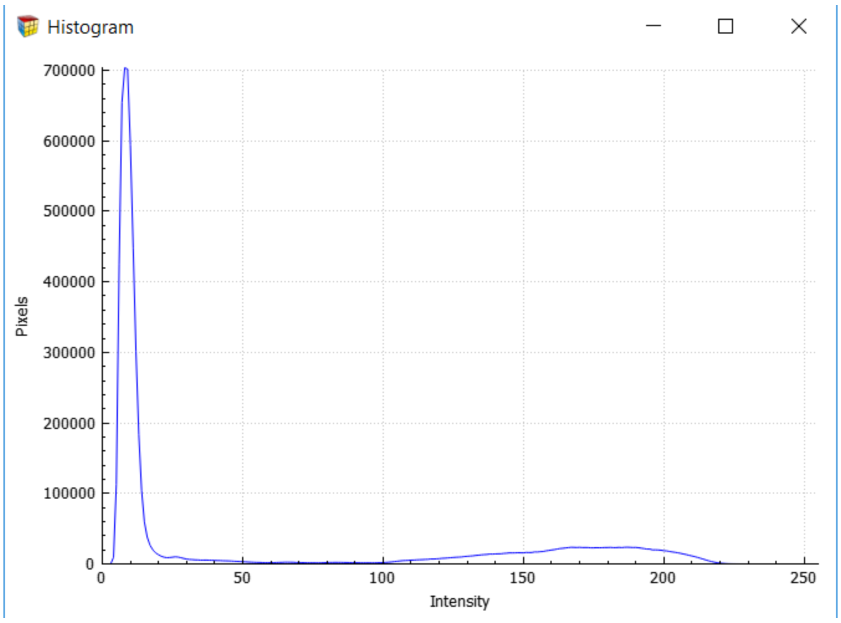


FIGURE 10 IMAGE HISTOGRAM

- **Reload Image:** After using some filters on the current image with this option the user can restore the original image from the cube.

- **Centroids:** By clicking this button another graph is revealed. Now the graphs are referring to some specific pixels which belong in different colors. Each graph refers to a different centroid with a different color so it can be distinguished. There is also an interactive legend where we can rename some centroids. Furthermore, if we want to clear the field and delete some graphs, there is a radio button that, when we want to do so, we set it enabled by click on it and then double click to the legend and delete the graph we selected.
- **Crop:** This is snipping tool which gives the user the opportunity, using his mouse, to mark a specific area on current image and create a new image which is the data within the area he chose.
- **Process:** This is a checkable process which provides to options called “Current” and “Cube”. By having checked the “Current” option all the filters are applied only in the current image and on the other hand if the option “Cube” is checked the filters the user selects, will be applied in all images of the cube.
- **Plot Profile:** By clicking this option, user is able to draw a line on the current image in the central widget and after that a new window pops up with a graph of the intensities from each pixel which is on the line the user just drawn.

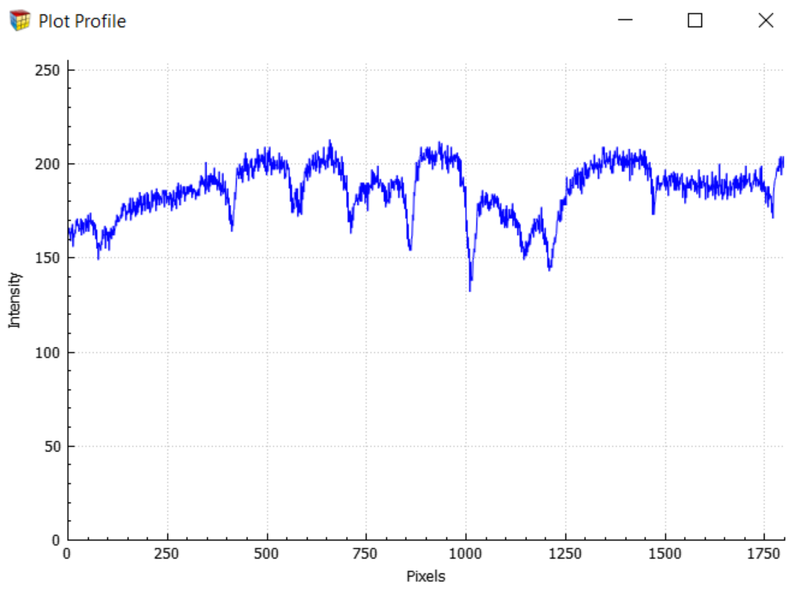


FIGURE 11 PLOT PROFILE

- **Surface Plot:** This option creates a 3D graph with the intensities of all the pixels within the current image.

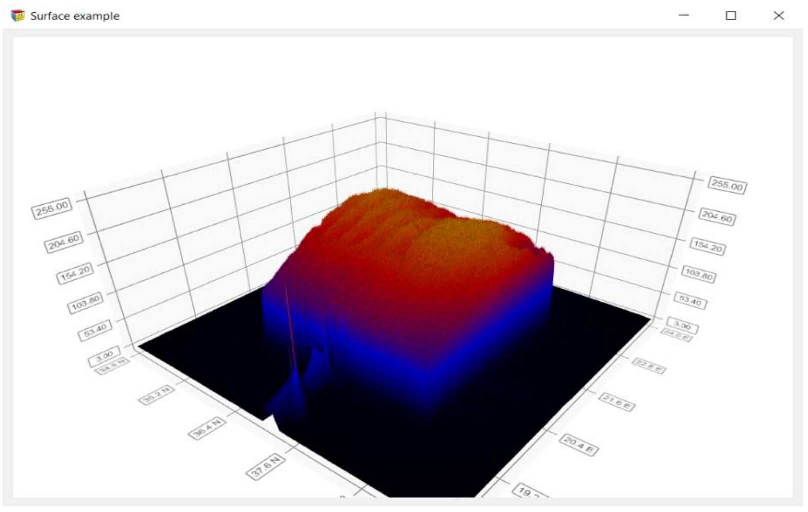


FIGURE 12 SURFACE PLOT

- **Transform:** This option has to different actions. The first one is "Flip", which inverts the current image (upside-down). The second option is "Rotate". Using this one the user can decide either clockwise or counterclockwise rotation for the current image.
- **Operations:** This option is referring to the pixels and its purpose is to replace each pixel value with the new result. There are four operations available:

- ✓ **Addition:** Click this one, a configuration window pops up, asking for a value which will be added in each pixel in all images of the imported cube. So, for each pixel of each image there is an addition with current pixel value plus the value which the user just inserts. If the result is not getting over 255, the result is replacing previous value. Otherwise, the new value will be 255.

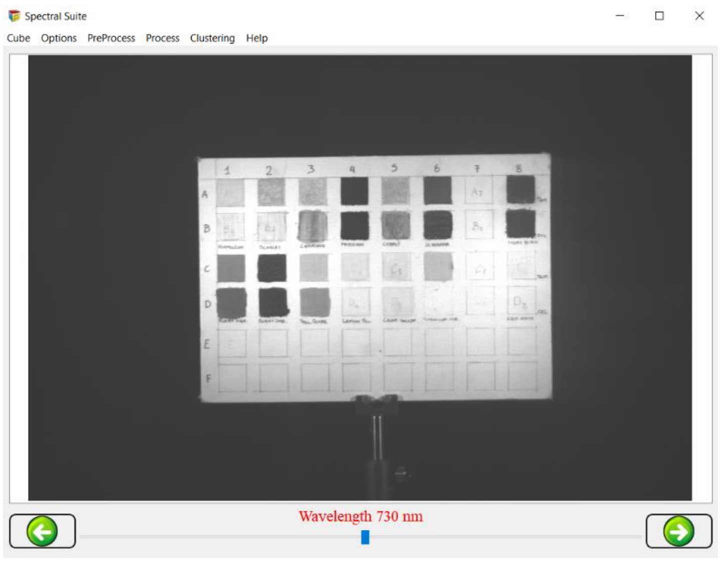


FIGURE 13 ADDITION (VALUE = 50)

- ✓ **Substraction:** In the same mood, there is a substraction, current pixel value minus the number which is inserted by the user. If the result is not less than zero, it replaces the current pixel value. Otherwise the new value is zero.

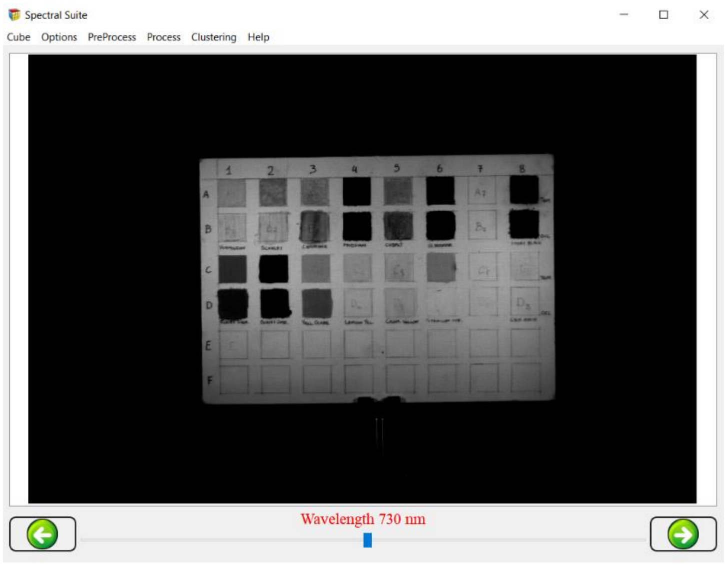


FIGURE 14 SUBSTRACTION (VALUE = 50)

✓ **Multiplication:** As previous, the inserted number is multiplied with the value of each pixel. If it overcomes number 255, the new pixel value is 255.

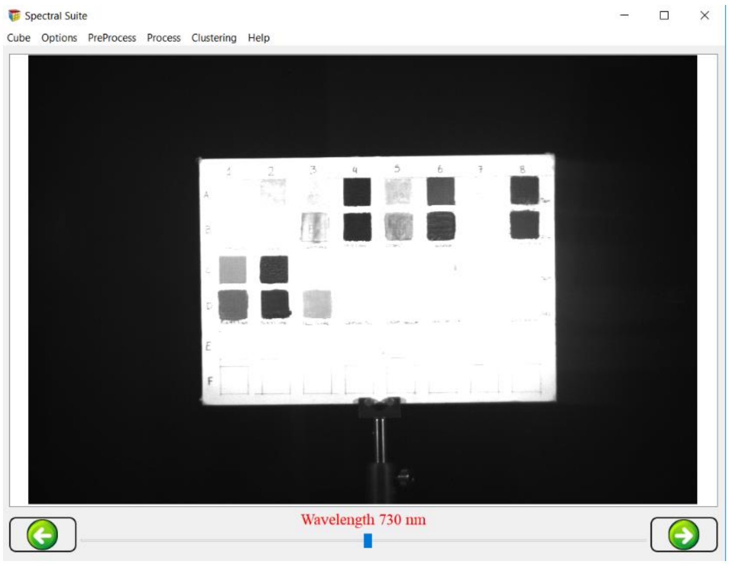


FIGURE 15 MULTIPLICATION (VALUE = 2)

✓ **Division:** Using the value which the user has inserted, there is a division between the intensity of each pixel and the inserted number. In this case like the others, the new value is replacing the old one.

- **Reload Cube:** Pushing this button, the cube, the user is currently handling, is being reloaded.
- **Substract Image:** On this action, there is a dialog window popping up, where the user is asked to choose one image, whichever he wants. After the loading of the image, there is a check on the dimensions of the selected image. If they match with the dimensions of the images from the loaded cube, there is a substraction between every pixel on the loaded image and the corresponding pixel in every image from the cube.
- **Merge Image:** On this action, there is a dialog window which pops up and there are 3 lists. One for each Red, Green and Blue channel. The user selects one image for each channel and the right pixel value is extracted. After that the three values are merged in a new pixel image, so in the end a new image is created and shown. There is a "Save" and "Exit" on the window where the merged image is shown.

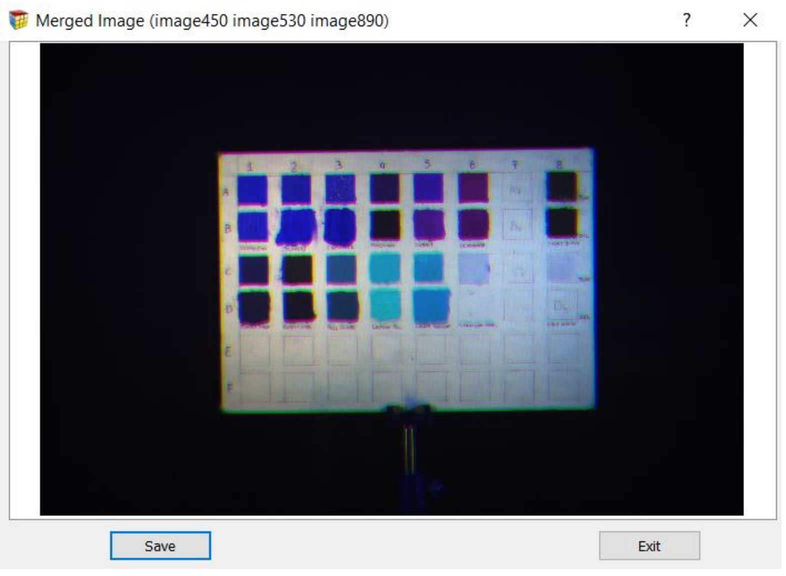


FIGURE 16 MERGED IMAGE

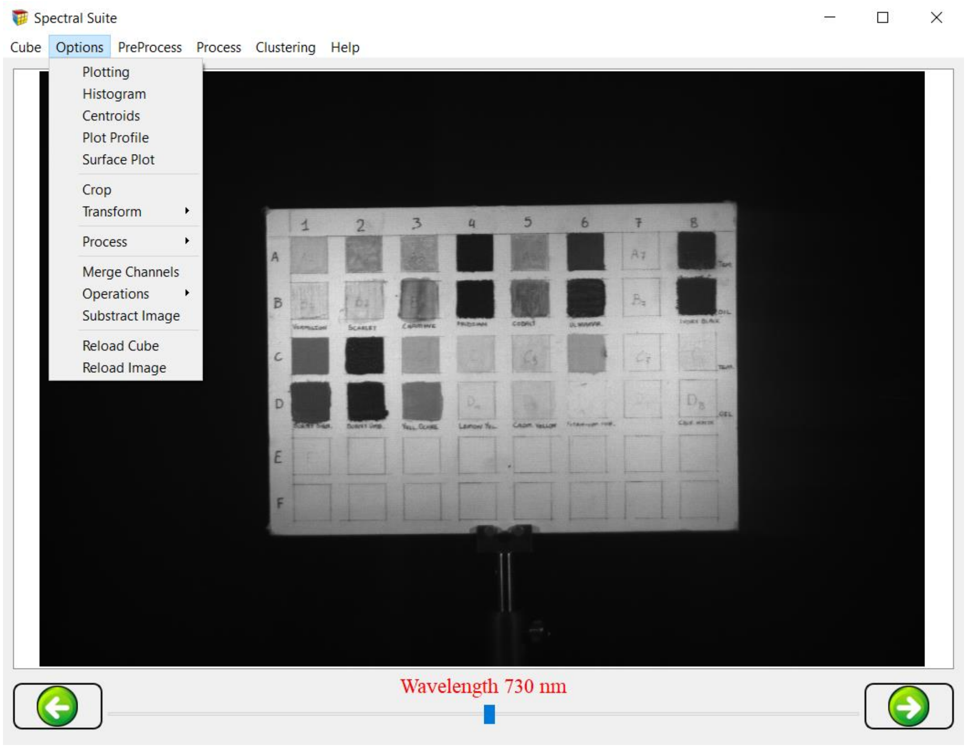


FIGURE 17 MENU OPTION "OPTIONS"

3. The option named “Help”. This option is divided into two different parts.
    - **“About this Suite…”:** Clicking this button and a new pop up window is revealed. In that, there are some information about the creator of this spectral suite, the supervisor professor and the copyrights.
    - **“About the plugins…”:** By having this button triggered, there is a pop up window which includes information about its one plugin that was parsed. There is information about the author of the plugin and the version of it.

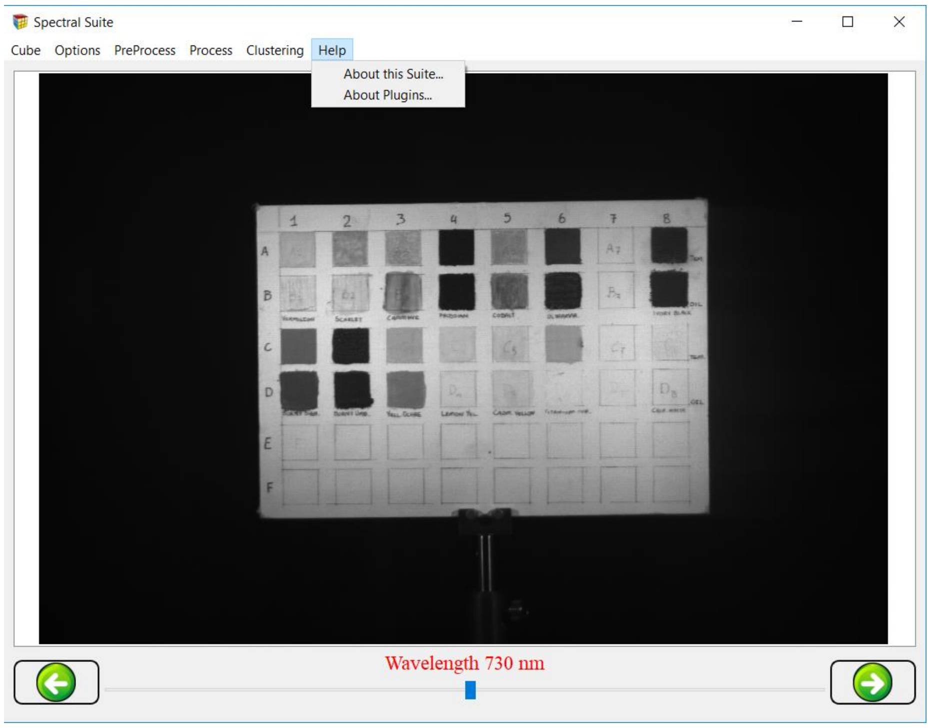


FIGURE 18 MENU OPTION "HELP"

There were the menu actions and their options which was added in the suite statically using tools and features which Qt Creator provides. In the next chapter comes the plugin logic and implementation of this spectral suite.

Chapter 5

Plugin Logic and Implementation

5.1 About plugins

In computing, a **plug-in** is a software component that adds a specific feature to an existing computer program. When a program supports plug-ins, it enables customization.

Web browsers have historically allowed executables as plug-ins, though they are now mostly deprecated. (These are a different type of software module than browser extensions.) Two plug-in examples are the Adobe Flash Player for playing videos and a Java virtual machine for running applets.

A theme or skin is a preset package containing additional or changed graphical appearance details, achieved by the use of a graphical user interface (GUI) that can be applied to specific software and websites to suit the purpose, topic, or tastes of different users to customize the look and feel of a piece of computer software or an operating system front-end GUI (and window managers).

The host application provides services which the plug-in can use, including a way for plug-ins to register themselves with the host application and a protocol for the exchange of data with plug-ins. Plug-ins depend on the services provided by the host application and do not usually work by themselves. Conversely, the host application operates independently of the plug-ins, making it possible for end-users to add and update plug-ins dynamically without needing to make changes to the host application.

Programmers typically implement plug-in functionality using shared libraries, which get dynamically loaded at run time, installed in a place prescribed by the host application. For example, HyperCard supported a similar facility, but more commonly included the plug-in code in the HyperCard documents (called *stacks*) themselves. Thus the HyperCard stack became a self-contained application in its own right, distributable as a single entity that end-users could run without the need for additional installation-steps. Programs may also implement plugins by loading a directory of simple script files written in a scripting language like Python or Lua.

Plug-ins appeared as early as the mid 1970s, when the EDT text editor running on the Unisys VS/9 operating system using the UNIVAC Series 90 mainframe computers provided the ability to run a program from the editor and to allow such a program to access the editor buffer, thus allowing an external program to access an edit session in memory.[14] The plug-in program could make calls to the editor to have it perform text-editing services upon the buffer that the editor shared with the plug-in. The Waterloo Fortran compiler used this feature to allow interactive compilation of Fortran programs edited by EDT.

Very early PC software applications to incorporate plug-in functionality included HyperCard and QuarkXPress on the Macintosh, both released in 1987. In 1988, Silicon Beach Software included plug-in functionality in Digital Darkroom and SuperPaint, and Ed Bomke coined the term *plug-in*.

Applications support plug-ins for many reasons. Some of the main reasons include:

- to enable third-party developers to create abilities which extend an application
- to support easily adding new features
- to reduce the size of an application
- to separate source code from an application because of incompatible software licenses.

5.2 Why to use plugins in this suite

As it was mentioned before, the purpose of this spectral suite is to create and develop a new framework which will be able to differ in some field to the other spectral suites. One purpose is to create a suite which will be well designed and easier to be used in contrast to the others. A second field is time. There is a specific effort by developing an optimal code to have real time results in applied algorithms. But the most important and vital purpose of this suite is compatibility and grow-up prospects. By creating a platform that is appropriate to use plug-ins, there is an obvious determination and prospect to deliver a framework fully dynamically designed in order, for itself, to be able to be used from other scientists and engineers in the future. People who will catch the pulse of their time needs and develop tools and features which will be able to be applied and reach results and goals vital to the future of the next generations.

So it is clear that by choosing to go with plugins the suite is a fully dynamically environment, easy to be used, easy to be grown up. Anyone in the future, after reading and understanding what the suite already provides, he will be able to add whatever he wants and anything he needs. He just need to follow the generic pattern which the base class follows and all the rest will be easily done.

5.3 Pattern of Plugins – Base Class

Now, speaking about the pattern which base class follows, it is composed of 8 virtual function which all will be implemented in each plugin main.cpp file.

The 8 virtual functions and their meaning is as follows:

- Title(): This function return a QString which is the name of the dll. This name will be shown in the menu bar in each category.
- Description(): There is again a QString return value with the description of each plugin.
- Configuration(): All plugins in this suite, and in the most features in general, needs some parameters to be more compatible. That's why, as you are going to see below, after the user clicks on a plugin, at first he has to trigger configuration window and add the values which are meant to be filled, and after that to proceed to the execution
- Execution(): This is the main function. Here goes the implementation of each plugin.
- Tested(): This is a function that helps the user to be sure about the compatibility of each design
- Author(): Here goes the name of the designer of each dll.
- Version(): Here goes the version of each plugin.

5.4 Plugins in this suite

The plugins, that are added in this suite, are separated to 3 different categories based on the purpose of their implementation

5.4.1 PreProcess Plugins

In this category, there are plugins intended to be applied before any clustering algorithm.

1. **Binning:** This is a plugin which helps the user to make some sort of combination between a group of pixels in order to shorten image's information so that it will be easier to be handled. The configuration window asks the user to choose either a 2*2 or 4*4 "window", which will be applied on the current image. The execution function gets this configure value as an argument and applies it on the image. The new image is loaded to the central widget.

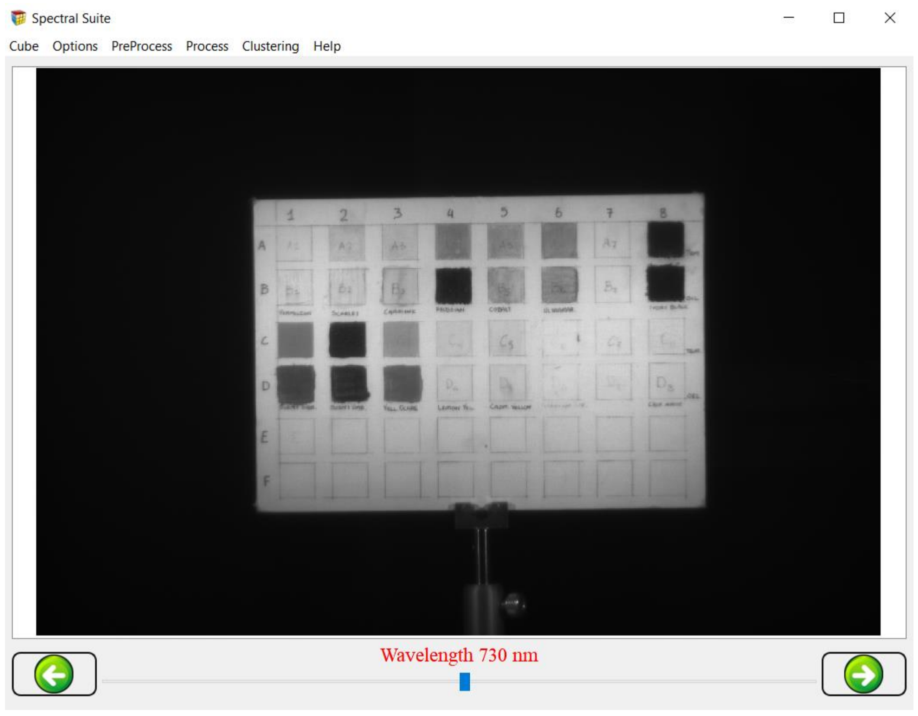


FIGURE 19 BINNING (2*2)

2. **Mean Filter:** Mean filtering is a simple, intuitive and easy to implement method of *smoothing* images, *i.e.* reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images. The idea of mean filtering is simply to replace each pixel value in an image with

the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (*e.g.* 5×5 squares) can be used for more severe smoothing. For the implementation of this filter the OpenCV's mean filter function is used.

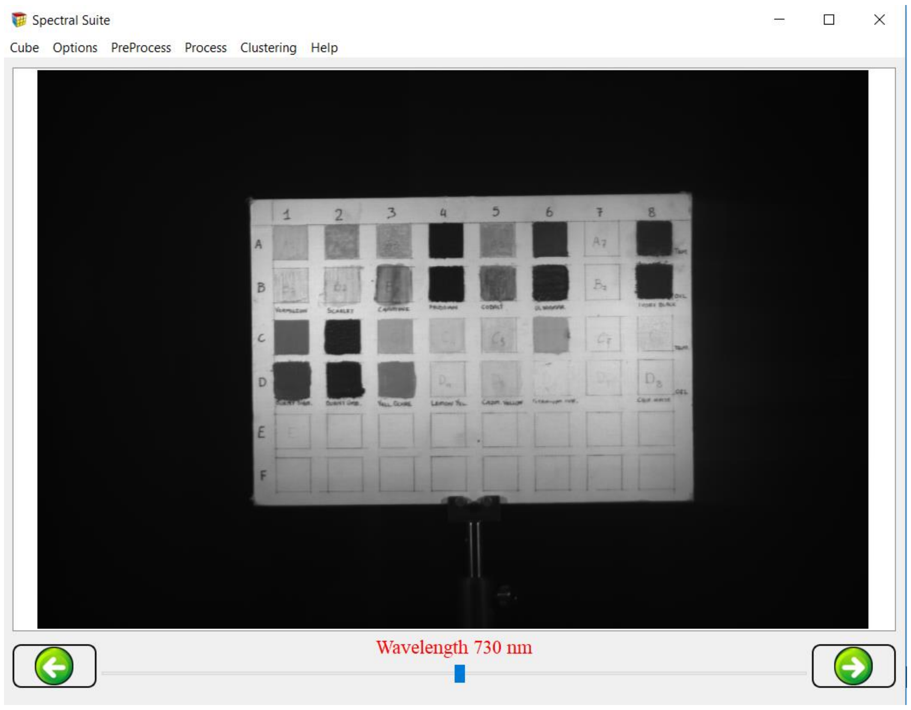


FIGURE 20 MEAN FILTER

3. **Median Filter:** This is a nonlinear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise (but see discussion below), also having applications in signal processing. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the

"window", which slides, entry by entry, over the entire signal. For 1D signals, the most obvious window is just the first few preceding and following entries, whereas for 2D (or higher-dimensional) signals such as images, more complex window patterns are possible (such as "box" or "cross" patterns). Note that if the window has an odd number of entries, then the median is simple to define: it is just the middle value after all the entries in the window are sorted numerically. For an even number of entries, there is more than one possible median. For this function, an OpenCV function was used, too.

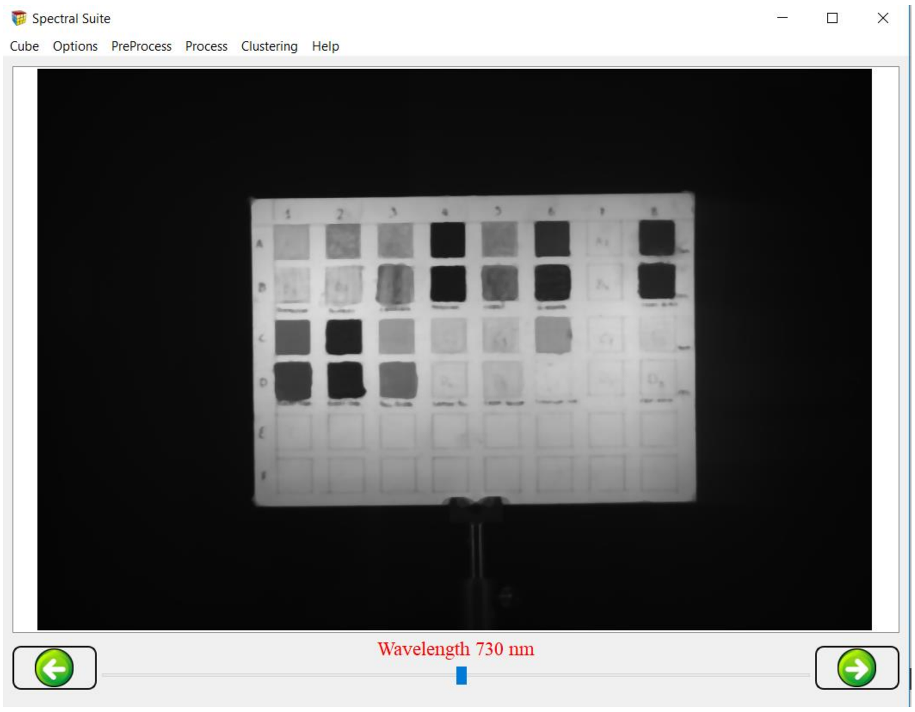


FIGURE 21 MEDIAN FILTER

4. **Despeckle Filter:** One of the fundamental challenges in image processing and computer vision is image denoising. What denoising does is to estimate the original image by suppressing noise from the image. Image noise may be caused by different sources (from sensor or from environment) which are often not possible to avoid in practical situations. Therefore, image denoising plays an important role in a wide range of applications such as image restoration, visual tracking, image registration, and image segmentation. While many algorithms have been proposed for the purpose of image denoising, the problem of image noise suppression remains an open challenge, especially in

situations where the images are acquired under poor conditions where the noise level is very high. There are two main types of noise in the image:

- **Salt and pepper noise** : It has sparse light and dark disturbances. Pixels in the image are very different in color or intensity from their surrounding pixels; the defining characteristic is that the value of a noisy pixel bears no relation to the color of surrounding pixels. Generally this type of noise will only affect a small number of image pixels. When viewed, the image contains dark and white dots, hence the term salt and pepper nose.
- **Gaussian noise**: "Each pixel in the image will be changed from its original value by a (usually) small amount. A histogram, a plot of the amount of distortion of a pixel value against the frequency with which it occurs, shows a normal distribution of noise. While other distributions are possible, the Gaussian (normal) distribution is usually a good model, due to the central limit theorem that says that the sum of different noises tends to approach a Gaussian distribution.

For this function, an OpenCV function was used, too.

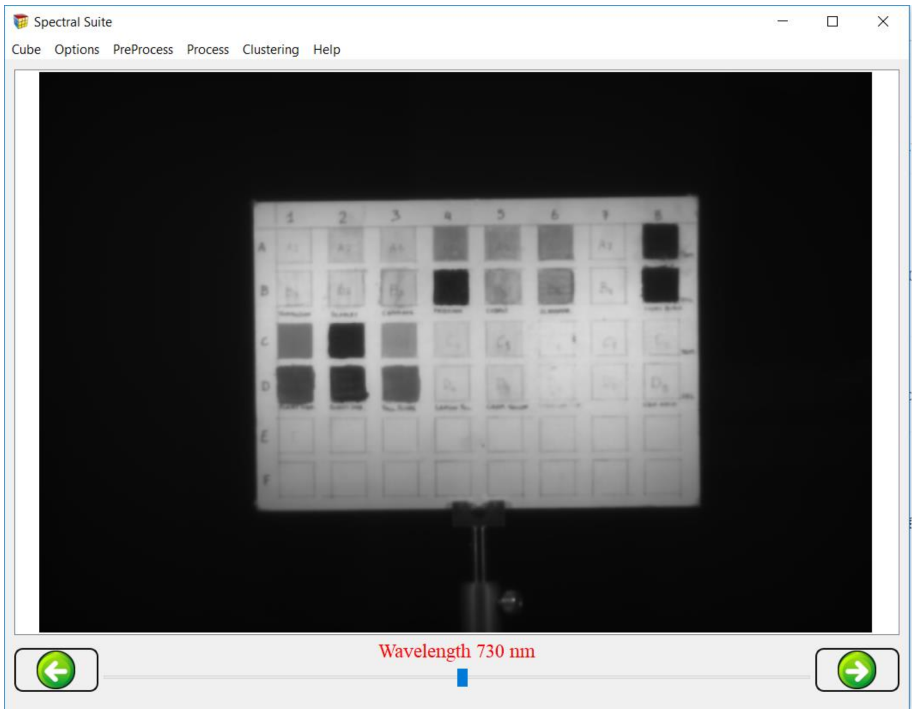


FIGURE 22 DESPECKLE FILTER

5. **Smoothing Filter:** Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noises. It actually removes high frequency content (eg: noise, edges) from the image. So edges are blurred a little bit in this operation. (Well, there are blurring techniques which doesn't blur the edges too). OpenCV provides mainly four types of blurring techniques.

    - **Averaging:** This is done by convolving image with a normalized box filter. It simply takes the average of all the pixels under kernel area and replace the central element.
    - **Gaussian Blurring**: In this, instead of box filter, gaussian kernel is used.
    - **Bilateral Filtering:** This is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters. We already saw that gaussian filter takes the a neighbourhood around the pixel and find its gaussian weighted average. This gaussian filter is a function of space alone, that is, nearby pixels are considered while filtering. It doesn't consider whether pixels have almost same intensity. It doesn't consider whether pixel is an edge pixel or not. So it blurs the edges also, which we don't want to do.
    - **Median Blurring:** Here, this technique takes median of all the pixels under kernel area and central element is replaced with this median value. This is highly effective against salt-and-pepper noise in the images. Interesting thing is that, in the above filters, central element is a newly calculated value which may be a pixel value in the image or a new value. But in median blurring, central element is always replaced by some pixel value in the image. It reduces the noise effectively. Its kernel size should be a positive odd integer.

For this function, an OpenCV function was used, too.

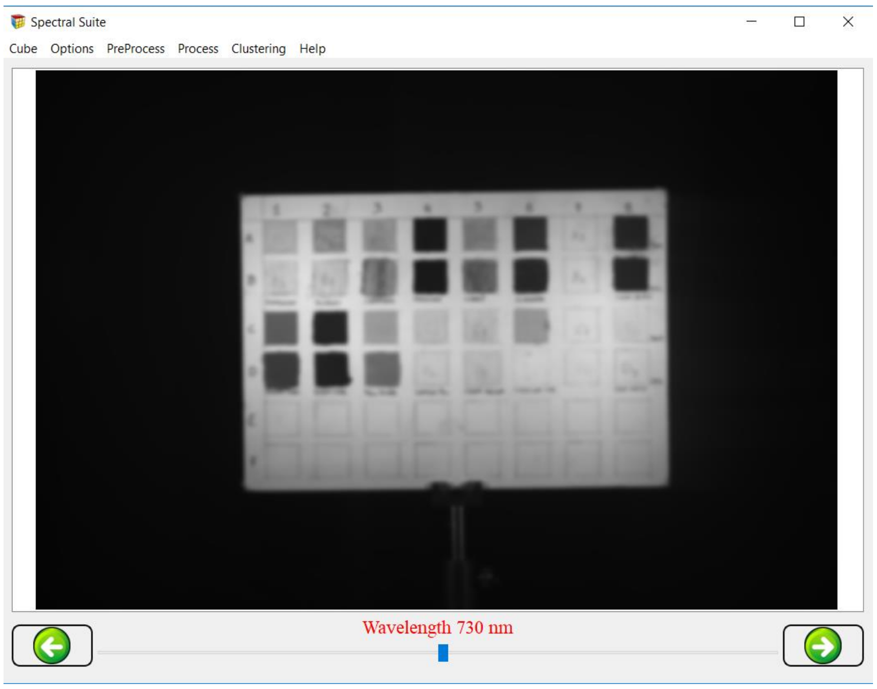


FIGURE 23 SMOOTHING FILTER - GAUSSIAN

5.4.2 Clustering Plugins

5.4.2.1 About Clustering

Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters. We can show this with a simple graphical example:

In this case we easily identify the 4 clusters into which the data can be divided; the similarity criterion is *distance*: two or more objects belong to the same cluster if they are "close" according to a given distance (in this case geometrical distance). This is called *distance-based clustering*. Another kind of clustering is *conceptual clustering*: two or more objects belong to the same cluster if this one defines a concept *common* to all that objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

5.4.2.2 The Goals of Clustering

So, the goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs.
For instance, we could be interested in finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties ("natural" data types), in finding useful and suitable groupings ("useful" data classes) or in finding unusual data objects (outlier detection).

5.4.2.3 Possible Applications

Clustering algorithms can be applied in many fields, for instance:

- Marketing: finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records
- Biology: classification of plants and animals given their features;
- Libraries: book ordering
- Insurance: identifying groups of motor insurance policy holders with a high average claim cost; identifying frauds
- City-planning: identifying groups of houses according to their house type, value and geographical location
- Earthquake studies: clustering observed earthquake epicenters to identify dangerous zones
- WWW: document classification; clustering weblog data to discover groups of similar access patterns.

5.4.2.4 Requirements

The main requirements that a clustering algorithm should satisfy are:

- scalability
- dealing with different types of attributes
- discovering clusters with arbitrary shape
- minimal requirements for domain knowledge to determine input parameters

- **ability** to deal with noise and outliers
- **insensitivity** to order of input records
- **high dimensionality**
- **interpretability** and **usability**.

5.4.2.5 Problems

There are a number of problems with clustering. Among them:

- current clustering techniques do not address all the requirements adequately (and concurrently)
- dealing with large number of dimensions and large number of data items can be problematic because of time complexity
- the effectiveness of the method depends on the definition of “distance” (for distance-based clustering)
- if an *obvious* distance measure doesn’t exist we must “define” it, which is not always easy, especially in multi-dimensional spaces
- The result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways.

5.4.2.6 Well known Clustering Algorithms

K-Means is probably the most well know clustering algorithm. It’s taught in a lot of introductory data science and machine learning classes. It’s easy to understand and implement in code.

Mean shift clustering is a sliding-window-based algorithm that attempts to find dense areas of data points. It is a centroid-based algorithm meaning that the goal is to locate the center points of each group/class, which works by updating candidates for center points to be the mean of the points within the sliding-window. These candidate windows are then filtered in a post-processing stage to eliminate near-duplicates, forming the final set of center points and their corresponding groups.

DBSCAN is a density based clustered algorithm similar to mean-shift, but with a couple of notable advantages. Suppose we have a set of points, it can group together points that are nearby neighbors and also marks the outliers points that lie on a big distance.

5.4.3 Clustering Algorithms in this suite

5.4.3.1 K-Means

In this suite K-Means is implemented. The implementation of K-Means is as follows in steps:

- **Initialization:** The first thing k-means does, is randomly choose K examples (data points) from the dataset as initial centroids and that's simply because it does not know yet where the center of each cluster is. (a centroid is the center of a cluster).
- **Cluster Assignment:** Then, all the data points that are the closest (similar) to a centroid will create a cluster. If we're using the Euclidean distance between data points and every centroid, a straight line is drawn between two centroids, then a perpendicular bisector divides this line into two clusters.
- **Move the Centroid:** Now, we have new clusters that need centers. A centroid's new value is going to be the mean of all the examples in a cluster. We'll keep repeating step 2 and 3 until the centroids stop moving, in other words, K-means algorithm is converged.

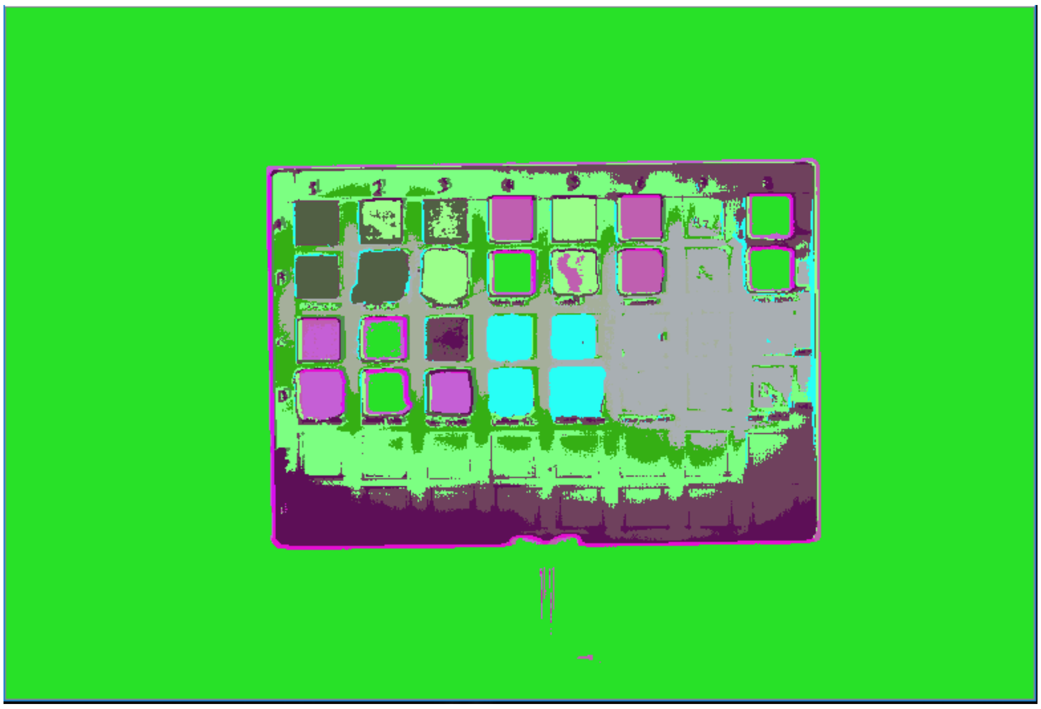


FIGURE 24 K-MEANS RESULT

5.4.3.2 DBSCAN

Except for K-Means, DBSAN is also implemented. It is about a density-based clustering algorithm where, given a set of points in some place, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). The DBSCAN algorithm basically requires 2 parameters:

- eps: the minimum distance between two points. It means that if the distance between two points is lower or equal to this value (eps), these points are considered neighbors.
- minPoints: the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region.

5.4.4 Process Plugins

These plugins are headed to be applied after the preprocessing filters like smoothing, median etc. In this category the algorithms which are implemented are:

- Contrast Enhancement: Contrast enhancement is a significant factor in any subjective evaluation of image quality which used to enhance the overall quality of the medical image for feature visualization and clinical measurement. This study presents a number of contrast enhancement techniques for medical images analysis. These techniques were applied on different type of medical images such as: MRI, CT-Scan and X-ray to improve image quality and come up with an acceptable image contrast. The proposed method included different enhancement techniques: logarithm and Exponential equations was created to improve the illumination and contrast of medical images, Image quality coefficients were extracted and compared with image quality coefficients for the same images, which were processed by the modified filter, and showed that the proposed method gave better results.

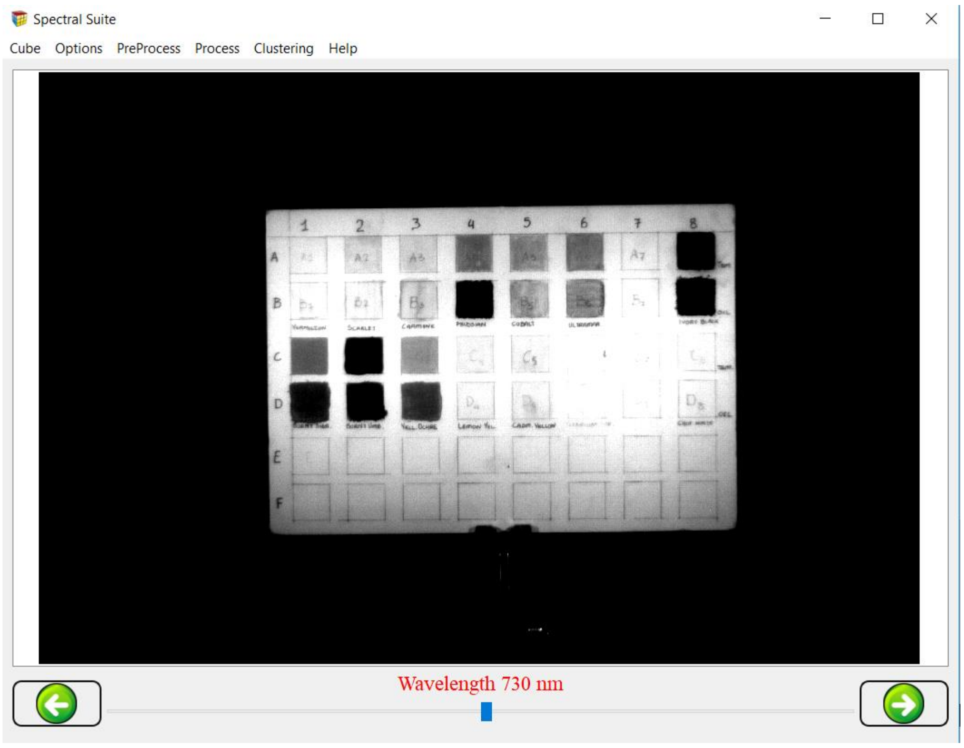


FIGURE 25 CONTRAST ENHANCEMENT (LIMIT 100)

- **Threshold:** In this plugin, there is again a configuration window which asks for a threshold value. Then, this value is compared to every pixel in the current image in order to get the result. If a pixel's value is lower than the value that the user gave, that pixel's new value is 0, otherwise the value remains as it was.

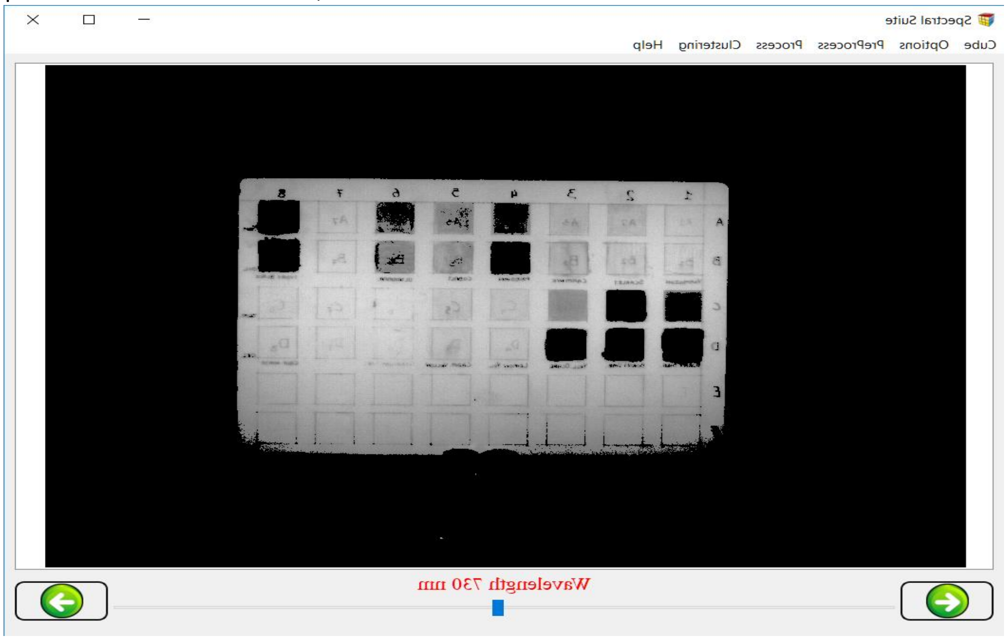


FIGURE 26 THRESHOLD (LIMIT = 120)

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Starting with what it was presented and analyzed in the former pages, at first there was an introduction in spectral imaging in general. We saw the definitions of some highly important concepts. Image processing, electromagnetic spectrum, spectral cube were presented so that anyone will be able to have an idea of what he is dealing with.

Furthermore, there was an extensive reference to multi and hyperspectral imaging. What hyperspectral imaging really is and the applications in which it can be applied, were collocated. Undoubtedly, there are countless applications and the importance of heading to this field of science and engineering deemed necessary.

After that spectral cubes were presented. It is vital for this work and the suite, anyone to understand what a spectral cube really means, how it is presented and what information he/she is able to extract from it. The whole spirit of this suite is handling spectral cubes and editing them in a specific way, given the features which are provided.

Subsequently, the importance of developing a hyperspectral suite for data analysis like this one was analyzed. Other similar software which are on market nowadays were presented. Also, there was a reference in similar applications which take place in TUC Electronics Laboratory and most important, the reasons why this suite must be developed given the fields which it covers.

After all that, an introduction on the way this suite works and the main features which were implemented, took place. There are several features for image processing and many different ways to apply these techniques either in a single image or the whole cube.

But the clue of this suite development and the reason why this suite is different and maybe with some addendums will be able to compete others, is the plugin thing. What a plugin is, the importance of having a dynamically created suite which any plugin, that follows the base class pattern, can be added. All these were analyzed so I hope now is clear why we were headed to that direction.

So, base class implementation and the dynamically created menus for PreProcess, Process and Clustering and their features were analyzed and specified.

6.2 Future Work

From now on, taking for granted that there is stable base with this suite and given the fact of portability of it, the suite can be expanded and many other plugins can be added. Other engineers who will use this suite, are able to add other features and use what already exists.

Moreover, it can be a new group of plugins, which can be added, based on GPU, like CUDA and OpenCl. Also, in this suite could be added algorithms which are used in Electronics Laboratory, like Hematology etc. In the end, more static features can be put on this suite for operations which are not be supported from this suite yet and maybe the GUI can be updated with menu bars and features so that it will be easier to work with.