**Universidade da Beira Interior**
Departamento de Informática

# Project - 2017: *Biomedical Image Analisys* : A Preparatory Study

Prepared by:

**Miguel Pereira 34323**

Supervisor:

**Professor Paulo Fazendeiro**

27 of June of 2017

# Acknowledgments

First of all, I express my gratitude to Professor Paulo Fazendeiro. In this gratefulness, I wish to acknowledge his ethical and pedagogy that I intend to honor, as well as all the suggestions and availability throughout this investigation.

To all my professors for sharing their knowledge, sharing resources, support, guidance and availability which contributed to the construction and finalization of this project.

To disentangle, thank all colleagues for the exchange of views and suggestions discussed in the course of the present investigation, a special thanks for Socia Lab students.

To my family and friends, my understanding of my occasional absence, and especially my patience and collaboration demonstrated in my non-availability throughout this investigation.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **API** | Application Programming Interface |
| **BSD** | Berkeley Software Distribution |
| **CMY** | Cyan Magenta Yellow |
| **FCM** | Fuzzy C-Means |
| **FCMFP** | Fuzzy C-Means with Focal Point |
| **FP** | Focal Point |
| **GPU** | Graphics Processing Unit |
| **GUI** | Graphical User Interface |
| **XB** | Inverse Xie-Beni |
| **iOS** | iPhone Operating System |
| **MRI** | Magnetic Resonance Imaging |
| **NM** | Nanometre |
| **OpenCL** | Open Computing Language |
| **OpenCV** | Open Source Computer Vision Library |
| **PFCM** | Penalized Fuzzy C-Means |
| **RGB** | Red-Green-Blue |
| **SKFC** | Kernel-Based Fuzzy Clustering with Spatial Constraints |
| **SA** | Segmentation accuracy |

| | |
|---|---|
| **SFCM** | Spatial Fuzzy C-Means |
| **TFP** | Total Factor Productivity |

# Chapter 1

# Introduction

## 1.1 Scope of the work

Within the latest years the use of the Internet has drastically changed the way we work, spend our leisure time and communicate with one another.Access to the Internet is almost universal in developed countries and although usage rates are much lower in the developing world, they are increasing.

Technologies affect growth in the economy. Traditionally, capital and labor are the "factors of production" that drive growth in the economy. Growth occurs when the stock of capital or labor increase, or when they are used more efficiently. The growth that comes from innovations and technological change in the economy is captured in Total Factor Productivity (TFP). Economists have always thought of new technologies as driving growth through their ability to enhance TFP. This made sense for the technologies that we have seen until now.

Today, we are witnessing the take-off of another transformative set of technologies, commonly referred to as Artificial Intelligence (AI). AI can replicate labor activities at much greater scale and speed, and to even perform some tasks beyond the capabilities of humans. Not to mention that in some areas it has the ability to learn faster than humans, if not yet as deeply. For example, by using virtual assistants, 1,000 legal documents can be reviewed in a matter of days instead of taking three people six months to complete. Similarly, AI can take the form of physical capital such as robots and intelligent machines. And unlike conventional capital, such as machines and buildings, it can actually improve over time, thanks to its self-learning capabilities.

The key factors of making AI grow are unlimited access to computing power and the growth in big data, with the current improvements in this fields AI technologies are emerging.

Hospitals have deployed swarms of doctor whose task is to look at a image and provide diagnosis with a specific disease. They may be looking to hundreds of images per day and doing a repetitive task all over again. With the growth of AI, why not make a Intelligent system to do these diagnosis by himself?

## 1.2 Motivation

My personal interest in this project lies in the foundations used: Mathematics, AI and Medicine. Since I've been learning more about Computer Science these past years, more enjoyment I've come to gain from seeing inventions or ideas related to AI, I find them fascinating. One project that really struck my mind was from professor Luís Alexandre [3]. Basically it analyzed Thorax images and a program with AI would make conclusions on whether the person image had liquid in it's lungs or not.

Having this in mind I thought on alternatives that could do a similar process. I also like Medicine and one of my favorite organs is the brain, therefore it seemed logical to adapt the problem from X-Ray to Magnetic Resonance Imaging (MRI), later I find tumors to be the best option to deal with. Since this project has a limited time to development only segmentation had been considered.

I think not only this project has academic interest but it has also high value due to the work it provides can replace a repetitive job from a doctor whose task is just looking at images and making these conclusions all over again.

This being said I think if I get good results with my experiments, getting a certain percentage of correct diagnosis, will lead this project to selling it to hospitals, reducing employer costs on that particular task. If the results get close to perfect it will be amazing as this will prove to be more efficient than doctors eyes and we can prevent diseases and respective treatment more effectively.

## 1.3 Objectives

In order to achieve the business objectives briefly presented in the last section we will need to accomplish the following goals: Implement a clustering algorithm to perform image segmentation and extract characteristics later on.

Since the time was limited, the focus on this project is strictly towards image segmentation and the comparison between two clustering algorithms. FCM and FCMFP. After the implementation of these algorithms, if the results are greater than already existent ones, great value is added to this research.

## 1.4 Document Organization

In order to reflect the work that has been done, this document is structured as follows:

1. The first chapter– **Introduction** – presents the project, the motivation for its choice, the framework for it, its objectives and the organization of the document.

2. The second chapter – **State of art** – with some degree of detail, explains thoroughly the theory behind image segmentation.

3. The third chapter – **Technologies Used** – describes the most important concepts within this project as well as the technologies used during the application development.

4. The fourth chapter – **Implementation and tests** – explains with technical terms how the clustering algorithms implemented work and the results obtained.

5. The fifth chapter – **Conclusions and Future work** – describes the problems faced in the development of the project and also discusses what could be improved and be made in the future.

# Chapter 2

# State of Art

## 2.1 Introduction

In this chapter I present the theory behind how the algorithms developed are working. Some areas are more important than others therefore being more thoroughly explained such as color and the concept of a image. At the end of the chapter the algorithms used are going to be explained graphically so that is easier to comprehend.

## 2.2 Theoretical Reference

### 2.2.1 Digital Image

Using information in the following document [1], we try to resume the following theory:

A image can be considered as the group of colored points or levels of grey. This images are named bitmaps. Any color can be defined as the quantity of each primary color, or in the other case, the scale goes from black to white.

We can also consider that a image is not made by points but by predefined objects, like squares, circles, ellipses,etc. This objects can receive numeric designations according their size, position, color,etc. We say in this case that the image is vectorial, meaning that it is made of defined objects and their numerous characteristics. Most images we work on have been through a process named digitalization that gives the image a digital representation.

A bitmap image is constituted by a group of horizontal and vertical points like lines or columns of a table. A pixel is the smallest element of the image in which we can associate a color. All images reproduced in the screen are bitmap images, the same way images obtained by a scanner or camera.

The resolution of a image(number of pixels by unit of width) influences the final quality of the image. On the other hand, growth of the number of pixels by unit of width increases the information to process leading to a increase of file size.

### 2.2.2 Light and Color

The light is a electromagnetic wave. The visible light is a electromagnetic wave ranging between 400 Nanometre (NM) and 700 NM. Its color is characterized by the length of wave of light. The white light contains all colors of the rainbow.

Nowadays it is known that the retina is composed by not just nerve endings sensible to colors (between 6 to 7 millions of cone cells mainly grouped in the center of the retina), but also by cells specialized in the capture of luminous bases distributed around all the surface of the retina, the rod cells (between 115 and 120 millions). The rod cells are sensible to luminous intensity in all gamma of wave-length that the human eye is sensible. Therefore, they can't distinguished between light received in different wave-lengths, meaning they don't detect color.

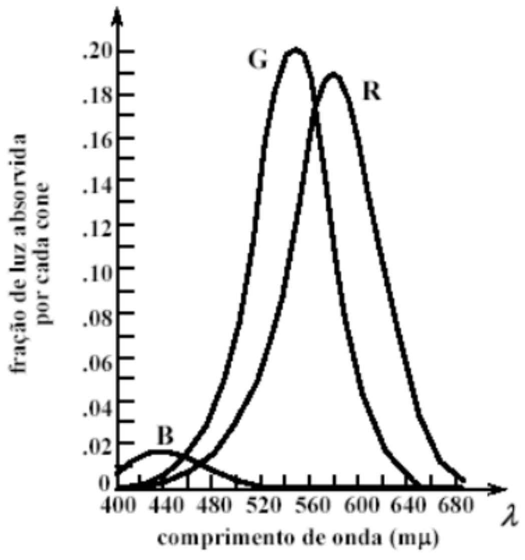


Figure 2.1: Light received

| **Type of cone cells** | **Main Color** | **Relative Distribution (%)** | **Detected Gamma (NM)** | **Bigger sensibility of λ(NM)** | **Fraction of light absorbed at λmax (%)** |
|---|---|---|---|---|---|
| β | blue | 4 | 350-550 | 440 | 2 |
| γ | green | 32 | 400-660 | 540 | 20 |
| ρ | red | 64 | 400-700 | 580 | 19 |

Table 2.1: Characteristics of cone cells

The table 2.1 shows the wavelength ranges at which the three types of cone cells are sensitive, the wavelength in which each of them presents greater sensitivity, its mean relative distribution and the incident light fraction which absorbs. The table 2.1 and the figure 2.1 allow us to take some notes:

- It is verified that the type of cone cells of type $\rho$ existing in the retina is almost two times larger than the number of $\gamma$-type cone cells. The number of cone cells of type $\beta$ is much lower than the number of cone cells of any of the other types.
- The highest sensitivity of the human eye should be in the range of wavelengths detected by the $\gamma$ and $\rho$ cone cells, the intermediate zone between green and red (+- 550 NM - yellow).
- It is in the lower wavelength range (blue zone) where it exists less sensitivity to color from all types of cones.
- The wavelengths between 400 and 600 NM are, in general, detected by all three types of cone cells, but each type detects a given wavelength with a different sensitivity. For example, the light with wavelength of 500 NM (cyan color) correspond to sensitivities of $\beta$ type cone cells of 20%, $\gamma$ cone cells of 30% and $\rho$ cone cells of 10%. For a wavelength of 550 NM (yellow), we will have 0% to $\beta$, 99% to $\gamma$ and 80% to $\rho$.
- It is the difference of the answers of these three type of cone cells that allow to interpret different wavelengths as corresponding to different colors.

In the next section, some color models will take advantage of the characteristics we just analyzed.

### 2.2.3 Digital color models

Numerous representation models of color spaces use a decomposition on a plane representing the luminosity and two planes representing the color information.

The meanings and definitions of these three planes varies slightly from one model to another, but its interpretation remains somewhat stable.

The plane of luminosity allows to represent the luminous intensity of a independent point of color. It thus models the operation of rod cells of the human eye. We can therefore work on the luminosity of a point without changing its color. This plan corresponds to the black and white version of a color image. It was really for compatibility reasons between the color screens and the black and white screens that appeared these types of models. Conversely, the two planes of color allow the representation of the independent tone to be luminosity. They are usually built to explore the perceptual properties of the human eye.

For compression applications these types of models can be beneficial. Several studies have shown that our eye is much more sensitive to brightness information than tone information. Thus, in the process of compression with losses we can introduce many more losses in these color planes than in the plane of luminosity.

#### 2.2.3.1 RGB Model

This is the most known model. It is used for the colors of our screens or televisions. Each color is obtained by additive synthesis of the three primary colors: red, green, and blue.

In this model a tone is obtained by a linear combination $\theta = P*R+\gamma*G+\beta*B$, with $(\rho,\gamma,\beta) \in [0,1]$. The triples (0,0,0) and (1,1,1) correspond respectively to black and white.

A color is defined by the numerical value assigned to each primary color, three integers. If these values are all at the maximum value the resulting color is the white. In contrast if the values are all null we get the color black.

If the value of the three primary colors is identical we obtain gray more or less dark depending on the distance of these values to the maximum. An image in 16 million colors mode is known as true color image. In view of the human eye that is capable of distinguish only 350 000 different colors this value is more than enough.

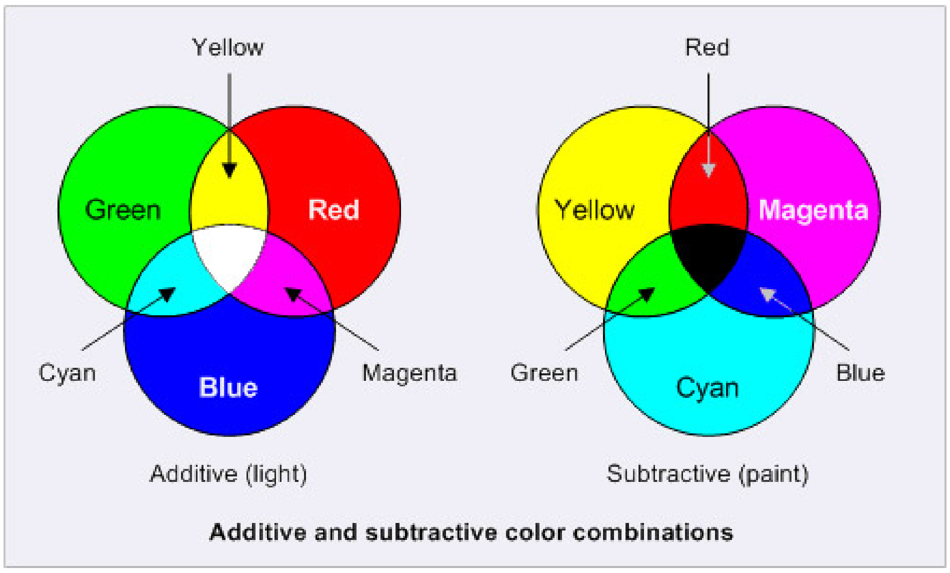


Figure 2.2: Primary and secondary colors of RGB and CMY color model

### 2.2.4 Image Segmentation

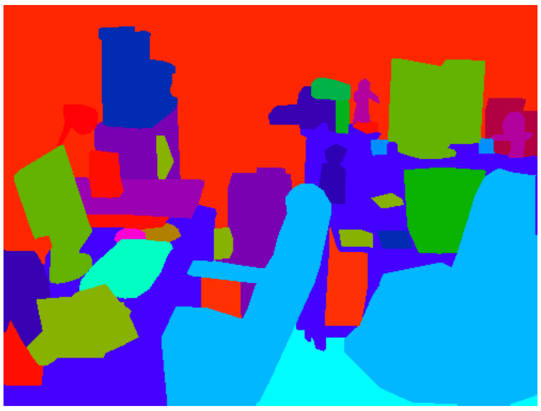
According to [4]: Image segmentation is an important and challenging problem and a necessary first step in image analysis as well as in high-level image interpretation and understanding such as robot vision, object recognition, and medical imaging.

The objective of image segmentation consists in separating a image into several regions with similar attributes such as intensity, color, tonality, texture,etc. The approaches in image segmentation algorithms can be divided in four categories: thresholding,clustering,edge detection and region extraction.

In this project we will use clustering methods, and we can define clustering as the process to identify patterns so that we can have classes that include samples within them. Hard clustering scheme in an image a pixel only has 1 class, on the other hand fuzzy clustering scheme can have a pixel within multiple classes. The last is more efficient dealing with noise in the image and among the fuzzy clustering methods, FCM is the more popular method used in image segmentation, although it has a limitation by not having information about spatial context.

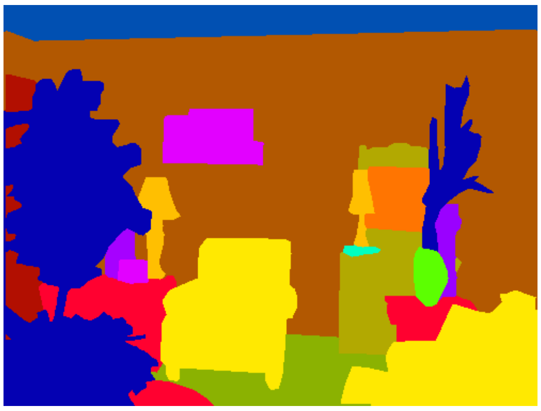
(a) First image

(b) Example segmentation of first image

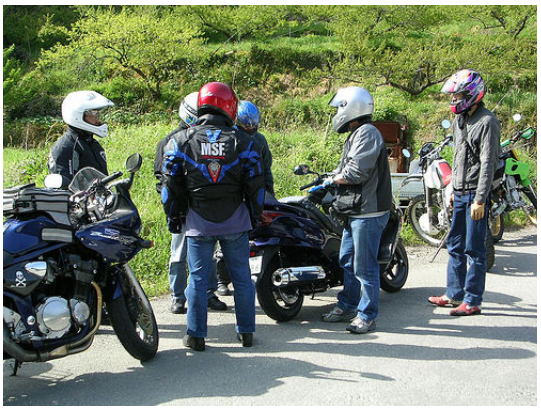


(c) Second image

(d) Example segmentation of second image



(e) Third image



(f) Example segmentation of third image

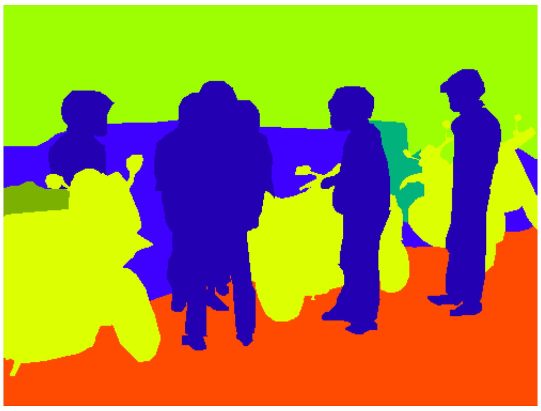


Figure 2.3: Examples of segmented images

### 2.2.5 Clustering

Clustering is the process of partitioning a group of data points into a small number of clusters. This clusters can be seen as classes to the extent of having inside of them data points with the same characteristics, physical or abstract. On the contrary data points in different clusters share more different characteristics.

For instance if we consider the products in a market, we can consider water, soda, beer to be group in drinks. This is a qualitative kind of partitioning. A quantitative approach would be to measure certain features of the products, like percentage of sugar or other component, and products with high percentage of this component would be grouped together.

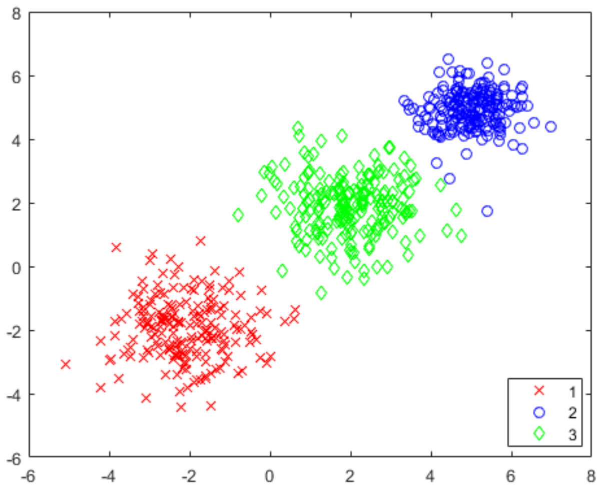


Figure 2.4: Clustering example

When we analyze figure 2.4 we notice two properties. Data points that have the more alike characteristics and are in the same cluster tend to be more close to each other, **Homogeneity**. And we notice that data points from different clusters tend to be far away due to having distinct characteristics, **Separation**.

### 2.2.6 K Means algorithm

K means algorithm is a Partitional clustering technique that partitions the data points into a specified number of groups. Given a data set with **n** objects, each one will belong only to one cluster.

When we start we define how much clusters we want to have and we initialize them, in this case we used K-Means in 2 dimension test case so it will be only 2 coordinates (x,y).

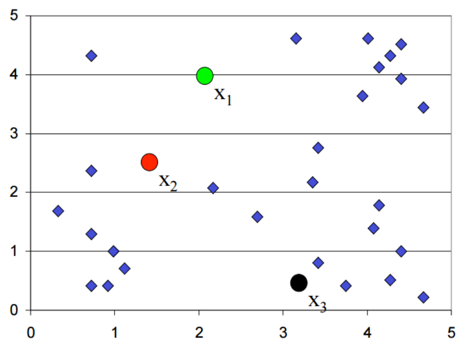


Figure 2.5: K means algorithm example phase 1

After this is done we will look at a pixel (0,0) or in this example the first x point and determine which cluster is the closest to it. When we look at all pixels we are done, and have a flag related each one of the pixels to understand which clusters belongs to it. We notice in fig. 2.6 that we already have all pixels classified by having a color on them.

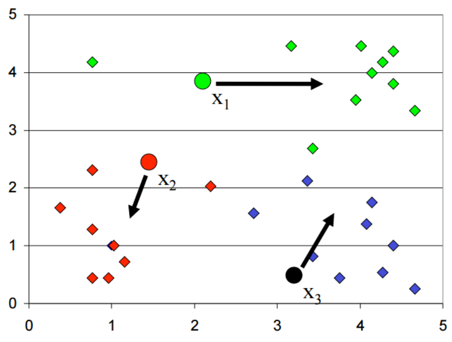


Figure 2.6: K means algorithm example phase 2

Then we calculate the center of the cluster by doing a sum on the coordinates x and y that belong to that cluster and divide by total number of points that belong to that cluster, mean of the cluster. We notice in fig. 2.7. that the clusters x1,x2 and x3 have moved because of this calculations.

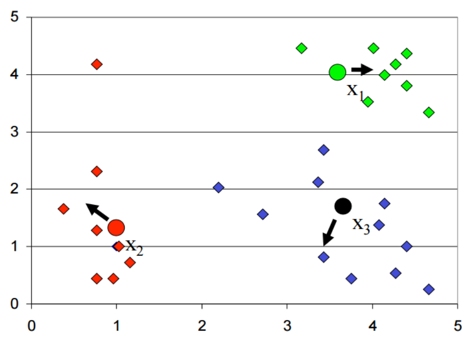


Figure 2.7: K means algorithm example phase 3

To the exception of the first run, if after running a cycle the clusters didn't change position, we end the algorithm and assume convergence is found. In fig. 2.8. we can see that we ran the algorithm again and what happened is that all the points continued with the same closest cluster. Since all the points remained the same, the calculation of the center of the clusters didn't change position, hence ending algorithm.

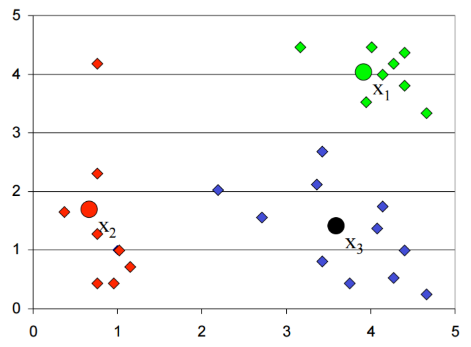


Figure 2.8: K means algorithm example phase 4

Have in mind that in this project the calculation towards distance cluster in K-Means was done using index values to the calculations(x,y) and FCM and FCMFP was done with color space coordinates, RGB.

### 2.2.7 FCM algorithm

The biggest difference between FCM and K-Means is that in FCM a given data point can belong to a cluster but also belong to other. It has a certain correlation mainly based on distance, it ranges from 0 to 1. When this value is 1 we imply that the element is in the exact position of the cluster whereas 0 the element is in the exact position of other cluster. The higher is the value the more alike is the element towards that cluster.

We start off defining how much clusters and we initialize them in a random space, since RGB uses 3 channels we will consider 3 dimensions(x,y,z).

Then we will look at every pixel of the image, starting at pixel (0,0) and look at cluster 1. Calculate the degree of membership to the partition matrix then look at cluster 2, repeat. When we fill this information for all the clusters we go to the next pixel and we repeat all process. To calculate the degree of membership of the partition matrix we are using Euclidean Distance between the point RGB values and the cluster RGB and doing some additional calculations, we will talk more in depth in chapter 4.

When we finish this calculation we will go through all clusters and calculate a variation of mean coordinates based on the correlation the pixels have towards that cluster.

Finally we calculate the difference between the last cluster position and the actual one, we do this for all clusters and save the difference in a variable. We will compare it with a constant we initialize at beginning of the program $\epsilon$ which represents when the program should stop. When variable > $\epsilon$ the cycle ends. High values of $\epsilon$ tend to be less precise, and when the value is low we get more precise results,this will be shown at chapter 4.

### 2.2.8 FCMFP algorithm

First of all we would like to reference [2], without the help of this article it would be impossible to have code this algorithm.

The concept is similar to FCM, however some details are different. One of them is that the n-dimensional vector space becomes n+1. So if in FCM we are using (x,y,z) in FCMFP we use (x,y,z,k). To implement this all data points coordinates are (x,y,z,0) and we generate clusters randomly. A Focal Point (FP) is created by calculating the Barycenter of all coordinates, so we will sum all data point specific coordinate divide by total number of points.

The purpose of this FP is to pull points from irrelevant clusters making them disappear with a algorithm that we will discuss later.

## 2.3 Conclusions

We have seen that image segmentation is important, since segmentation accuracy determines the recognition of certain objects or regions ans also the eventual success or failure of computerized analysis procedure.

This analysis can perform critical operations like the diagnosis of diseases, computer-guided surgery, pedestrian detection, face detection, brake light detection, locate objects in satellite imagery, agricultural imaging-crop disease detection, fingerprint and iris recognition.

# Chapter 3

# Technologies and Tools Used

## 3.1 Introduction

In this section I present the general conditions of the fulfillment of the project.
Operative System - Linux Mint 18.1 Cinnamon 64 bit.
Developed in a simple text editor in Python language, Version 2.7.12.
Extra resources used:

- Numpy
- Open Source Computer Vision Library (OpenCV)
- Matplotlib
- Scikit-image
- Scipy

## 3.2 NumPy

NumPy is a library for the Python programming language, adding support for N-dimensional arrays and matrices objects, along with a large collection of high-level mathematical functions to operate on these arrays, tools for integrating C/C++ and Fortran code and also useful linear algebra, Fourier transform, and random number capabilities. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

## 3.3 OpenCV

OpenCV is released under a Berkeley Software Distribution (BSD) license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iPhone Operating System (iOS) and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with Open Computing Language (OpenCL), it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

OpenCV has Image Processing and Video I/O modules, Data Structure, Linear Algebra, Basic Graphical User Interface (GUI) with independent window system, Mouse and keyboard control, as well as more than 350 Computer Vision algorithms such as: Image filters, camera calibration, object recognition, structural analysis and others. Its processing is in real time of images.

## 3.4 Matplotlib

Matplotlib is a Python 2D plotting library and and its numerical mathematics extension NumPy. It produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For a sampling, see the screenshots, thumbnail gallery, and examples directory

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

It provides an object-oriented Application Programming Interface (API) for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine, designed to closely resemble that of MATLAB. SciPy makes use of matplotlib.

## 3.5 Scikit-image

Scikit-image is a Python package dedicated to image processing, and using natively NumPy arrays as image objects. It can also be used to efficiently and easily convert a image color space.

Scikit-image (skimage) is a collection of algorithms for image processing and computer vision. The main package of skimage only provides a few utilities for converting between image data types.

## 3.6 Scipy

SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering. A collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics and much more. In particular, these are some of the core packages:

The scikit-image SciKit (toolkit for SciPy) extends scipy.ndimage to provide a versatile set of image processing routines. It is written in the Python language.

This SciKit is developed by the SciPy community.

## 3.7 Conclusions

All the work done in this project was not possible without the use of every single package, API, library present in the past chapters. It's very important to note that the time that they had to be developed is in general a great amount and for that reason we should pay tribute in some way.

Python has proven to be a great tool for fast developing at image processing, particularly image segmentation. Using OpenCV to convert a image to a matrix and the operations that we do with them are very easy to do.

# Chapter 4

# Implementation and testing

## 4.1 Introduction

In this chapter we will discussed the algorithms implemented, the theory behind them and later we will present results. In this project, we look at three algorithms namely K Means clustering, FCM algorithm and the FCMFP and we will compare them for image segmentation. We also present ways to improve this algorithms and methods that allow us to evaluate our solution.

## 4.2 Segmentation algorithms

### 4.2.1 K Means Clustering Algorithm

The Lloyd's algorithm, mostly known as k-means algorithm works as follows.

In general, we have n data points x$_i, i = 1...n$ that have to be partitioned in k clusters. The goal is to assign a cluster to each data point. The K-means clustering uses the square of the Euclidean distance $d(x, \mu_i) = ||x - \mu_i||_2^2$.

1. Decide the number of clusters k
2. Initialize the center of the clusters:
   $\mu_i$ = some value, $i = 1, 2, .., k$
3. Attribute the closest cluster to each point:
   c $_i = \{j : d(x_j, \mu_i) \leq d(x_j, \mu_l), l \neq i, j = 1, ..., n\}$
4. Calculate the mean position of each cluster based on the points that are within that cluster:
   $\mu_i = \frac{1}{|c_i|} \sum_{j \in c_i} x_j, \forall i$

5. Repeat steps 3-4 until no point has changed cluster (convergence)

### 4.2.2 Fuzzy C-Means Clustering Algorithm

The algorithm is an iterative clustering method where X = $x_1, x_2, ..., x_n \subseteq R^p$ is the data set in the p-dimensional vector space, n is the number of data items, c is the number of clusters with $2 \leq c < n, u_{ik}$ is the degree of membership of $x_k$ in the $i^{th}$ cluster, q is a weighting exponent on each fuzzy membership, $v_i$ is the prototype of the centre of cluster i, $d^2(x_k, v_i)$ is a distance measure between object $x_k$ and cluster center $v_i$. An iterative process, is described as follows:

1. Set values for c, q and ε.
2. Initialize the fuzzy partition matrix U = $[u_{ik}]$.
3. Set the loop counter b = 0.
4. Calculate the c cluster centers $\{v_i^{(b)}\}$ with $U^{(b)}$:
   $v_i^{(b)} = \frac{\sum_{k=1}^{n}(u_{ik}^{(b)})^q x_k}{\sum_{k=1}^{n}(u_{ik}^{(b)})^q}$
5. Calculate the membership $U^{b+1}$. For k = 1 to n, calculate the following: $I_k = \{i|1 \leq i \leq c, d_{ik} = ||x_k - v_i|| = 0\}$, /I; for the $k^{th}$ column of the matrix, compute new membership values:
   (a) if $I_k = \emptyset$, then
   $u_{ik}^{(b+1)} = \frac{1}{\sum_{j=1}^{c}(\frac{d_{ik}}{d_{jk}})^{\frac{2}{(q-1)}}}$
   (b) else $u_{ik}^{(b+1)} = 0$ for all $i \notin I$ and $\sum_{i \in I_k} u_{ik}^{(b+1)} = 1$; next k.
6. If || $U^{(b)} - U^{(b+1)}$ || < ε, stop; otherwise, set b = b + 1 and go to step 4.

The main difference between K-means and FCM is that while in K-means a pixel belongs to a single cluster, in FCM a pixel has a coefficient to any of the clusters, ranging from 0-1, this way it becomes more accurate and forces a partition matrix to store all coefficient between pixels and clusters.

#### 4.2.2.1 Pixel with same position as clusters

In the beginning we had this idea, if the pixel is in the same coordinates as a cluster, the correlation between the two would be 1 and 0 to the rest of the clusters.

Later on we find this to not be the best approach, so we will still consider 1 if the pixel is in same position as cluster but for the rest of the clusters the correlation between them will be $\frac{1}{x}$ being x the total number of pixels that are on the same position of the cluster.

This question arrived when we had several pixels in the same position as the clusters.

### 4.2.3 The FCMFP algorithm

The problem can be converted into a constraint-free version through the use of Lagrange multipliers. The augmented objective function reads as

$$Q_{\mathcal{L}} = \sum_{i=1}^{C_{max}} \sum_{j=1}^{n} u_{ij}^m D_{ij} + \zeta \sum_{i=1}^{C_{max}} D_{iP} + \lambda(\sum_{i=1}^{C_{max}} u_{ij} - 1) \tag{4.1}$$

where $D_{ij} = d_{ij}^2 = ||\mathbf{x}_j - \mathbf{v}_i||_A^2 = (\mathbf{x}_j - \mathbf{v}_i)^T A(\mathbf{x}_j - \mathbf{v}_i)$ and $D_{iP} = d_{iP}^2 = ||\mathbf{P} - \mathbf{v}_i||_A^2 = (\mathbf{P} - \mathbf{v}_i)^T A(\mathbf{P} - \mathbf{v}_i)$ for an inner product norm, $A$ being a $(d \times d)$ positive definite norm-inducing matrix.

The minimization of $Q_{\mathcal{L}}$ can be obtained as follows.
For $i = 1, 2, \ldots, C_{max}$ and $j = 1, 2, \ldots, n$, consider $I_j = \{i \mid d_{ij} = 0\}$; assume $||.||_A$ to be an inner product induced norm; fix $m \in (1, \infty)$ and $\zeta \in [0, \infty)$ and let $\mathbf{P}$ be a point in the data space. Then $(U, V) \in (F_{C_{max}n} \times R^{C_{max}d})$ may be globally minimal for $Q_{\mathcal{L}}$ only if

$$u_{ij} = \begin{cases} \frac{1}{\sum_{k=1}^{C_{max}} \left(\frac{D_{ij}}{D_{kj}}\right)^{\frac{1}{(m-1)}}}, & \text{for} \quad I_j = \emptyset \\ \frac{1}{|I_j|}, & \text{for} \quad I_j \neq \emptyset,\ i \in I_j \\ 0, & \text{for} \quad I_j \neq \emptyset,\ i \notin I_j \end{cases} \tag{4.2}$$

$$\mathbf{v}_{ik} = \frac{\sum_{j=1}^{n} u_{ij}^m \mathbf{x}_{jk} + \zeta \mathbf{P}_k}{\sum_{j=1}^{n} u_{ij}^m + \zeta}. \tag{4.3}$$

**Algorithm 1** Fuzzy C-means with focal point in a higher dimensional space

```
Let $\mathbf{X}= \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset R^d$ be a finite set of unlabeled data.
Initialize the clusters' prototypes $\mathbf{V} \in R^{d \times c}$.
Set $C_{max}$, $m > 1$, $\mathbf{P} \in R^w (w \geq d)$ and $\zeta \geq 0$.
Extend X and V into $R^w$ by introducing $(w - d)$ null coordinates per element.
Repeat the following steps until a termination criterion has been met.
    Step 1 - For $i = 1, 2, \ldots, C_{max}$ and $j = 1, 2, \ldots, n$, update the partition matrix, U, according to (4.2).
    Step 2 - For $i = 1, 2, \ldots, C_{max}$, update the prototypes, V, according to (4.3).
Project the prototypes into the original feature space $R^d$.
```

**Algorithm 2** Iterative fuzzy C-means with focal point

```
Set $c', C_{max}$, $(1 < c' < C_{max})$, $\zeta \geq 0$, and $\Delta\zeta > 0$.
Repeat the following steps until the number of candidate clusters is smaller than $c'$.
    Step 1 - Apply the FCMFP algorithm.
    Step 2 - Remove neglectable clusters (clusters without any typical datum).
    Step 3 - Compute the validity measure for the remaining candidate clusters.
    Step 4 - Update $\zeta \leftarrow \zeta + \Delta\zeta$.
Output the partition(s) optimizing the considered validity measure.
```

## 4.3 Cluster quantity

One of the most significant problems in image segmentation is to determine the correct number of clusters in a image. If we run any of the segmentation algorithms with a different number of clusters than the original image the results are proven to be less precise. Therefore we must use some method to answer this question.

The effectiveness of this choice is verified by cluster validity analysis. One possible way of performing a classical cluster validity analysis consists in running the clustering algorithm for different values of C, several times with different initialization. The validity of the obtained partitions is assessed by validity measures: the number C which optimizes one of these measures (or a combination of some of them) is chosen as the optimal one.

Currently there are a wealth of internal validity indices to evaluate (fuzzy) partitions. Some theoretical and empirical studies show that the Xie–Beni (XB) index tends to provide one of the best responses in what concerns the validation of partitions produced by FCM, over a wide range of hyper-parameters values (i.e., $c$).

The objective is to seek clustering schemes where most of the vectors of the data set exhibit high degree of membership to a particular cluster while maintaining the clusters' centers as far as possible from each other. The Xie–Beni index is computed as the ratio of the compactness of the fuzzy partition of a data set to its separation. In this work, for convenience in the visualization of the results we use the inverse of this index:

The Xie–Beni index aims at identifying compact well separated clusters. Noting that the value of Xie–Beni index is infinite when the outputs are undesirable solutions, for convenience of computer calculations and presentation, we adopted here the inverse Xie–Beni index ($XB^{-1}$):

$$XB^{-1} = \frac{n \min_{i \neq j} ||\mathbf{v}_i - \mathbf{v}_j||^2}{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m ||\mathbf{x}_j - \mathbf{v}_i||^2}$$

It is clear that large values of $XB^{-1}$ are expected for compact and well-separated clusters, moreover we note that the maximization of $XB^{-1}$ corresponds to the minimization of the FCM objective function.

Using this index we will run our algorithm in the exact conditions but using different cluster number. Throughout the runs whichever has the lowest index value means that it is the correct number of clusters.

## 4.4 Cluster Relevance

This new method allowed the algorithm to be faster and not to compromise quality. Works as follows:

We will pick one point at a time and then look for the highest degree of membership of all the clusters. When we find the cluster with this property we will change a flag related to that cluster, meaning that the specific cluster is relevant. When we go through all points we will remove all the irrelevant clusters.

One interesting result is that whenever we use this code with FCM the cluster quantity was never decremented. This means all the clusters stay relevant. After some time debugging the program it was confirmed that sometimes clusters had a few data points that belonged to them, hence staying relevant.

In order to also solve this problem we used FCMFP. Since we introduce another coordinate, the FP will grab data points which are on clusters almost irrelevant and pull them away from the cluster. This way after some runs this cluster will have zero elements that belong to him, making him irrelevant.

## 4.5 Classification of Pixels

One interesting result in [4] demonstrated several algorithms and compared the number of misclassified pixels in the segmentation, the algorithm with the less number would be the more efficient one. This seemed to be a great way to classify segmentation algorithms therefore we decided to implement this feature.

At first we simply compared two pixels, one from the original image and the other from the result of the segmentation algorithm, if they were equal a counter would be incremented. At the end the counter would be divided by number of pixels and we would multiply by 100 to get a percentage. The Segmentation accuracy (SA) is defined as follows:

$$\text{SA} = \frac{\text{Number of correctly classified pixels}}{\text{Total number of pixels}} \times 100$$

What we would verify at first is that we would get 0 percentage meaning the result of our segmentation would never be the same as the original image. This happens because if the RGB values from the original image are (20,130,223) the results from our segmentation would be (20.000001,130.0000356,223.000057). We also tried to use some sort of standard deviation, but that proven to not effective as we do not know what value we should consider to make our segmentation classification accurate.

After some research we came across the term human segmented images, here is some example:

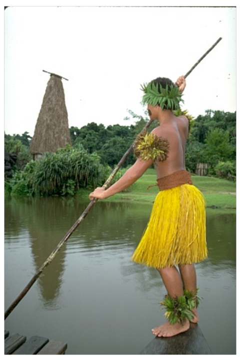


Figure 4.1: Original image

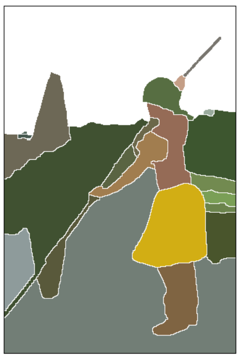


Figure 4.2: Human Segmented Image

The second approach was to pick the human segmented image , use the result of our segmentation on the original image and do the process described earlier. This way we would compare the RGB values that in theory would be more equal.

This prove to be slightly better as we could have some percentage of correct classified pixels although, we still suffer from the problem described in the first approach.

From researching data set we also notice another type of human segmented images that we already have seen in this project.

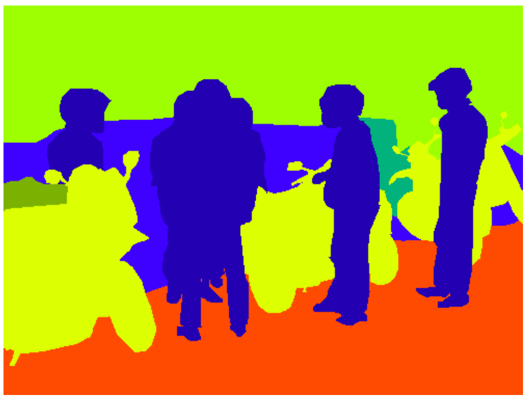


Figure 4.3: Example segmentated image

This types of images have completely nullified the last approach because cluster regions are painted in a random color, what matters is the region itself. We can view this as the regions can be represented by a number of letter, something abstract.

Now after facing this problem we have come to our current approach which is to map the colors between the human segmented images and the result of our algorithm on the original image. This can be done by doing the following:

First we declare a empty list and go through all pixels on the human segmented image and every time we found a RGB value of the pixel we are on, we check if the triple already exists in the list. If it does we continue else we add the triple in the list.

Second step is to declare a matrix and the objective is to fill it with the cluster number that the image pixel has. We fill this information by looking through all the pixels and comparing it's RGB value to the RGB values of the list declared in the first step. When both this values match we extract the index. The index is the cluster number that point has.

Third step is declaring a matrix that corresponds to the result of our segmentation on the original image. We will fill this matrix with the cluster that has the most degree of membership towards that point.

The final step is the hardest, and in here we try to map the clusters from the two images. A more simple example will be given to understand properly how the method works:

| | | | |
|---|---|---|---|
| A | A | B | B |
| B | A | B | B |
| B | B | A | A |
| B | B | B | A |

Table 4.1: Partition matrix of human segmented image

| | | | |
|---|---|---|---|
| C | C | C | D |
| D | C | D | D |
| D | D | C | C |
| D | D | C | C |

Table 4.2: Partition matrix of FCM segmented image

Now we Assume C is A and obviously D is B. If we make this prediction and look at the table 4.3. we will increment a counter everytime the supposition between matrix are correct. To be more easy we will put 1 and 0 whenever the mapping is correct.

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |

Table 4.3: Partition matrix of mapping both matrix C->A, D->B

With this table 4.3 we can conclude that this supposition has 14 correct classified pixels. So now we have other possibility left: C is B and D is A. Here's the result:

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |

Table 4.4: Partition matrix of mapping both matrix C->B, D->A

With this table 4.4 we can conclude that this supposition has 2 correct classified pixels. Since the last combination has a highest number of correct classified pixels we will assume the real cluster mapping is C being A and D being B.

Have in mind that in the real case, we test images with 7 or more clusters, when we are doing suppositions we want to do a loop throughout all possible combinations, which takes more or less 15 minutes.

Another note is that the matrix instead of having char A,B,C,D it has integers ranging from 0 to cluster number.

## 4.6 Experimental Results

In this section I present the results I obtained with my program that implemented the FCM algorithm. Here emerged the difficulty in finding a method to be able to evaluate the competence of my implementation. Therefore I decided to paint all pixels of the image read. If the image had 225x255 pixels I would paint the same number of pixels. The corresponding color would be determined by the degree of membership toward a cluster, if in a point the bigger degree was cluster 3 then the color would be the color of cluster 3(e.g. red). With this method we can have a graphical vision of how the algorithm behaves. Besides this aspect, we can include the time of execution of the program to compare with a larger number of clusters and of other algorithms that will be discussed further ahead.

It also arise some difficulty in comparing different color spaces results with the same algorithm because even if the parameters of the program are the same, since RGB has 3 coordinates and with Cie lab we used 2 coordinates, it is not possible to have the exact same conditions on the experiment.

For all cases, unless otherwise stated, the weighting exponent q = 2.0, e = 0.00001. All the algorithms are coded in a simple text editor using Python Version 2.7.12 and are run on a AMD Phenom(tm)II X4 925 Processor*4 personal computer with a memory of 2.0 GB.

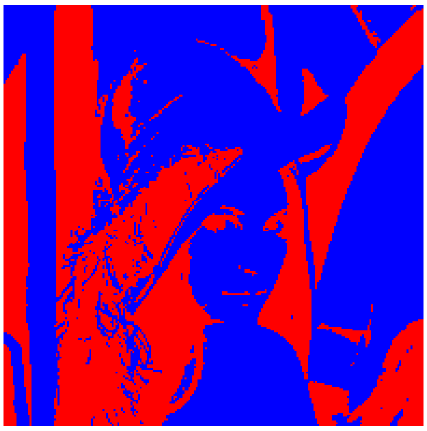


Figure 4.4: FCM algorithm using Lena image, 2 clusters and RGB colorspace

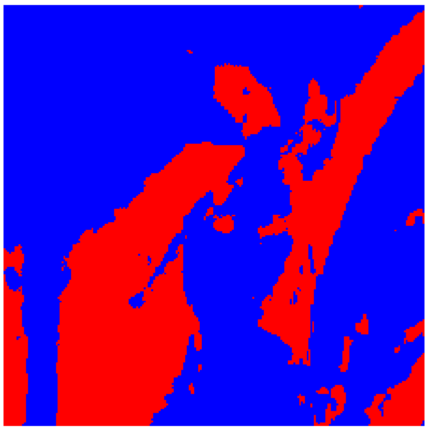


Figure 4.5: FCM algorithm using Lena image, 2 clusters and LAB colorspace

| **Figure** | **Execution Time** | **Initial cluster** | **Final cluster** |
|---|---|---|---|
| 4.4 | 62.3s | (59,145,227) | (89,130,100) |
| 4.5 | 23.2s | (-25,72) | (-106,89) |

Table 4.5: Results of FCM algorithm using Lena image,2 clusters and RGB and Lab colorspaces.

After some time testing the algorithm I noticed that painting with a predefined color would make it look like the segmentation of the image was not precise. Therefore the use of the mean RGB values (cluster values) seemed the most appropriate to paint per pixel. The result is visible in the following figure.

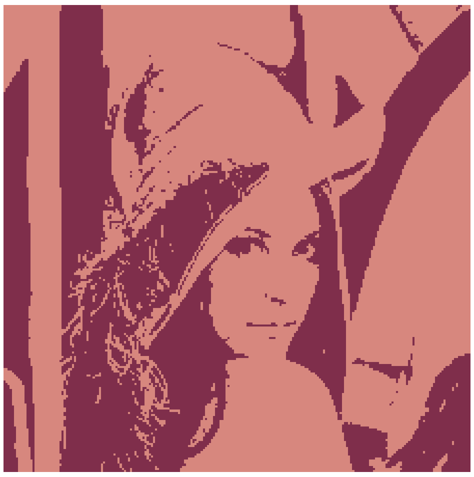


Figure 4.6: FCM algorithm using Lena image, 2 clusters and using its values in RGB

### 4.6.1 XB Results

In here we will provide the results of comparing the results of the index XB with the visualization of the result of the algorithm. In order to demonstrate this difference we used different values of $\epsilon$. 0.0001,100,1000, respectably.

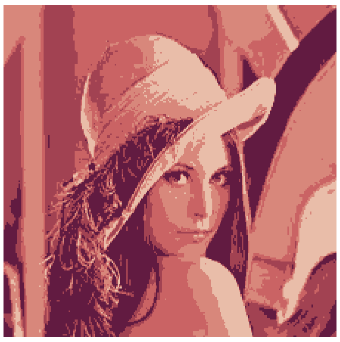


Figure 4.7: Lena image using FCM with 5 clusters, $\varepsilon = 0.0001$

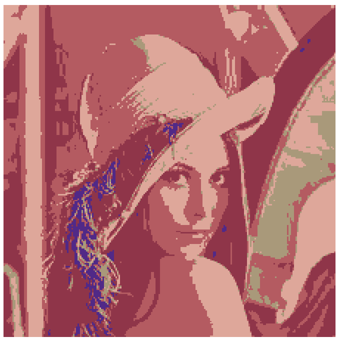


Figure 4.8: Lena image using FCM with 5 clusters, $\varepsilon = 100$

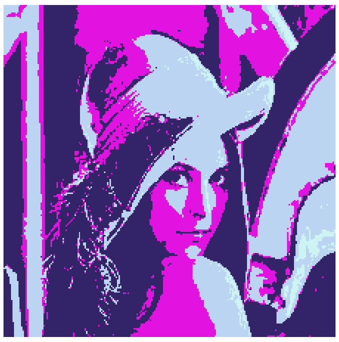


Figure 4.9: Lena image using FCM with 5 clusters, $\varepsilon = 1000$

| Figure | ε | Execution Time | XB |
|---|---|---|---|
| 4.7 | 0.0001 | 619.7s | 5.69845 |
| 4.8 | 100 | 41.8s | 4.752438 |
| 4.9 | 1000 | 28.3s | 0.272292 |

Table 4.6: Results of FCM algorithm using Lena image,5 clusters,RGB colorspace with different ε

Analyzing table 4.7 we conclude that having a low value to $\epsilon$ will lead to more precision and high values to low precision of the algorithm segmentation. This happens because whenever we calculate the cluster coordinates to establish his mean, if we move very little and have a high $\epsilon$ it's more likely to end the algorithm by supposing the cluster didn't move.

We can also conclude that having higher precision with our segmentation algorithms leads to a higher XB index and therefore we can agree that this is indeed one of the best responses in what concerns the validation of partitions produced by FCM, over a wide range of hyper-parameters values.

#### 4.6.1.1 Different Cluster Number

In here we present the most important motive on why XB index exists, **discovering the correct number of clusters**.

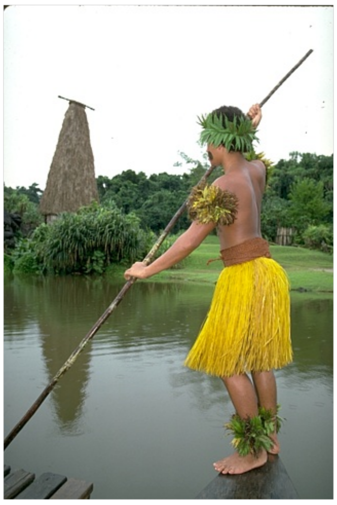


Figure 4.10: Original image

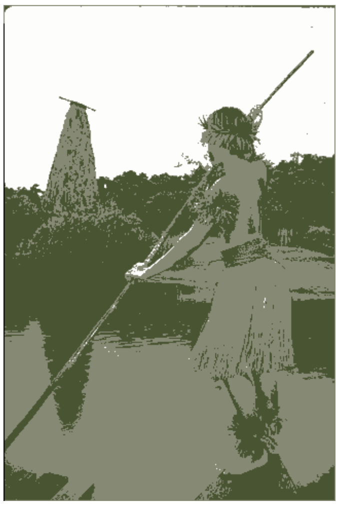


Figure 4.11: Result from FCM using 3 clusters

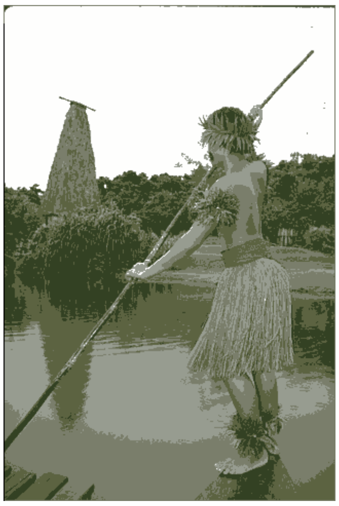


Figure 4.12: Result from FCM using 5 clusters

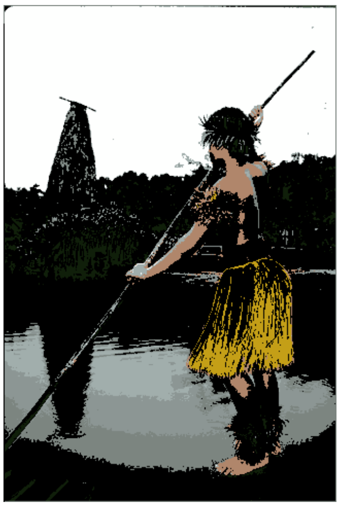


Figure 4.13: Result from FCM using 7 clusters

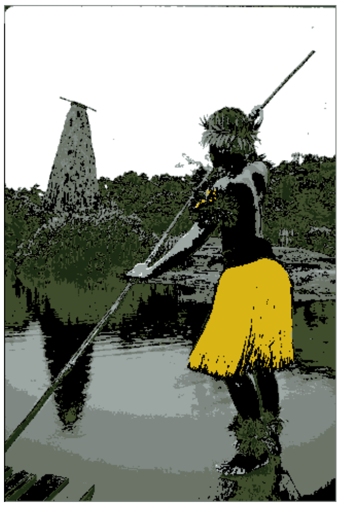


Figure 4.14: Result from FCM using 9 clusters

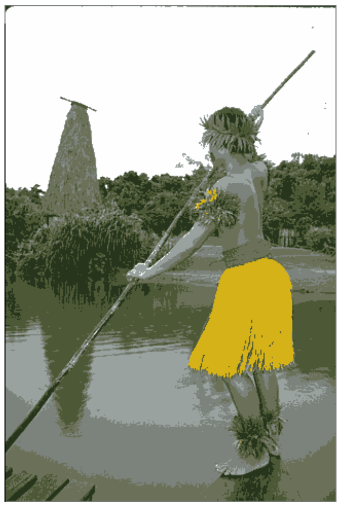


Figure 4.15: Result from FCM using 15 clusters

| Figure | ε | Execution Time | XB |
|---|---|---|---|
| 4.11 | 0.0001 | 619.7s | 3.70 |
| 4.12 | 0.0001 | 3663.6s | 5.94 |
| 4.13 | 0.0001 | 5175.3s | 7.24 |
| 4.14 | 0.0001 | 12389.7s | 8.23 |
| 4.15 | 0.0001 | 21300.9s | 11.63 |

Table 4.7: Results of FCM algorithm using images 4.11-4.15

Analyzing table 4.7 we conclude that having a higher number of cluster increases the precision since the original image has a lot of colors. This can be verified with the index XB that whichever value is the highest the more precision the images tend to get. What we did not verified is the XB index decreasing with the increase of clusters. We estimated the image to have 25 clusters and therefore we couldn't test with more clusters since another problem that arises is the amount of time needed to run the algorithm.

### 4.6.2 Classification of Pixels Results

After we demonstrated how the algorithm that evaluate the pixels works we provide the result we got and the SA obtained in order to analyze the results. We present two images: a simple one which we assume the original image and human segmented image are the same. And a more difficult one that has a lot more complex space of features. Also note that the human segmented image of this type is not completely correct as the image itself serves to segment objects into categories.

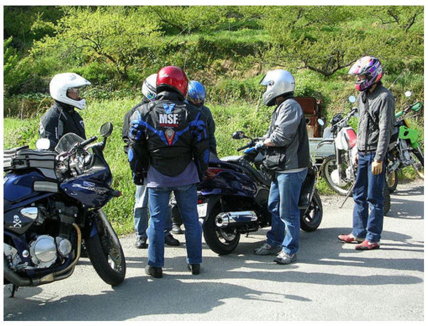


Figure 4.16: Original image

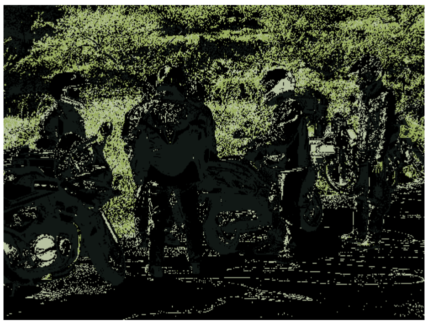


Figure 4.17: Result from FCM from the original image, using 7 clusters

The SA in this experiment was 31.69 %. This result was to be expected since the image is very complex.

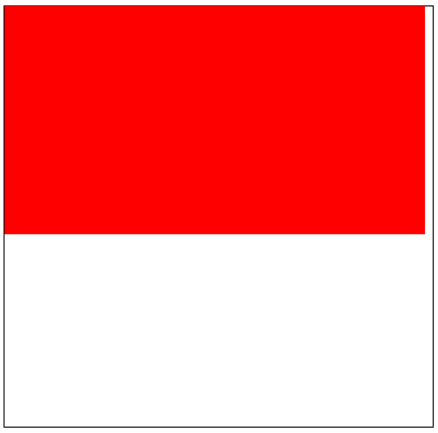


Figure 4.18: Original image

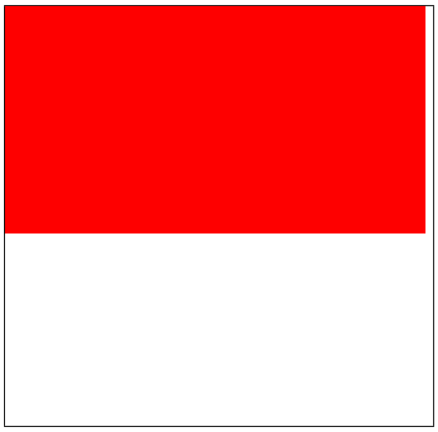


Figure 4.19: Result from FCM from the original image, using 2 clusters

The SA in this experiment was 100 %. Since the segmentation was simple it was accurate and the percentage confirmed that we we're doing things correctly. A border was added to this image in order to see the white region in the image, the report contrasted its white.

### 4.6.3 FCMFP results

This section shows very disappointing results from a algorithm that has a lot of potential, the problem is that when we finished implementing the algorithm we had few time to make our experiments. Another factor is that we hardly could finish a run of FCMFP because it would take from a day to a week to run with few clusters and $\epsilon$.

The key to having good results from FCMFP is having very low increments of $\zeta$ which made the clusters move. The problem was that, the much lower was this increment the more time was needed to finish.

Unfortunately, since we were unable to finish a test with a low increment, we will show a result obtained from a very high increment and 2 clusters. Whenever a run was finished $\zeta$ would increase by 10.

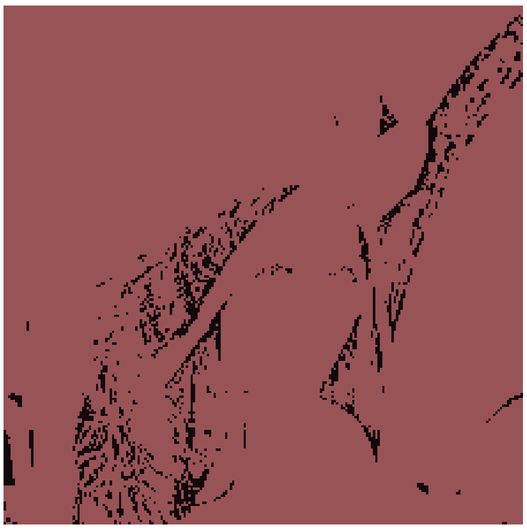


Figure 4.20: Result obtained from Lena image using FCMFP with 2 clusters

### 4.6.4 Breast Experiments

The premise towards this experiment is estimate how much milk does a woman have, by analyzing its breast volume and using thermodynamic data to see areas that are more hot than others.

This can be done with 3 steps.

- Cropping the image to obtain just the breast area
- Eliminate the artifact
- Apply a segmentation algorithm with 2 clusters to estimate milk quantity

Among our data set we will differentiate two different types of image: The first type is distinguished by having their upper member lift horizontal at the shoulder line, whereas the second type will have the members along it's body.

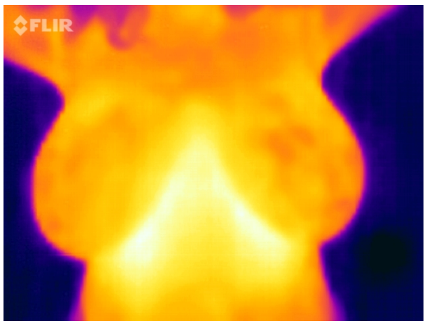


Figure 4.21: First type of image from breast data set

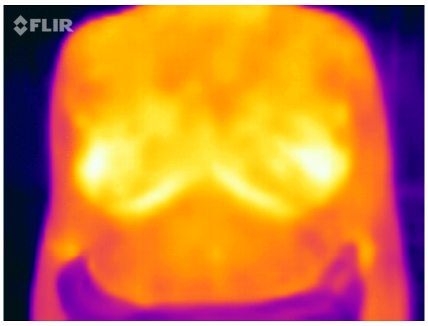


Figure 4.22: Second type of image from breast data set

#### 4.6.4.1 Cropping image

Even though the image has already been cropped, we want to strictly look at the muscle pectoralis Major. We have two types of images in the data set: People that have their arms raised and people who has arms near the body.

For the first case it's more simple. We only have to look for the axilliary medium point which is the highest and draw a horizontal line and eliminate what is in the upper part. We also notice in the image often people have a towel in the lower parts, since we want to eliminate this, we will draw a horizontal line in the abdominal, and we will eliminate what is lower to that line.

For the second case, we will remove the following regions constituting the upper members: carpal(wrist), antebrachial(forearm), antecubital(front of elbow), brachial(arm), acromial( point of shoulder). The breast zone will lead up to the sternal. The lower part remains the same, remove the towel and remove the abdominal extension.

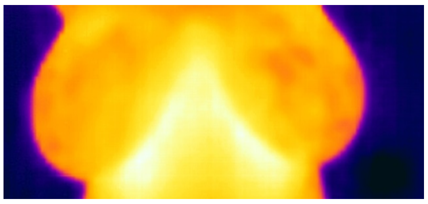


Figure 4.25: Result of cropping the first type of image from breast dataset

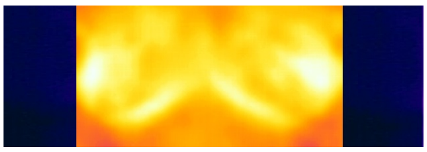


Figure 4.26: Result of cropping the second type of image from breast dataset

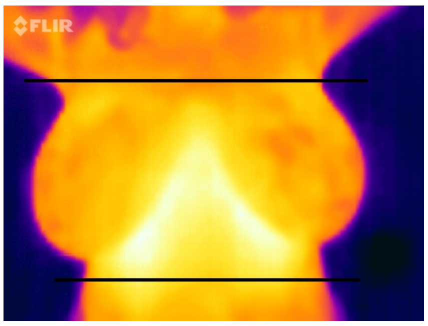


Figure 4.23: Graphically crop the first type of image from breast dataset

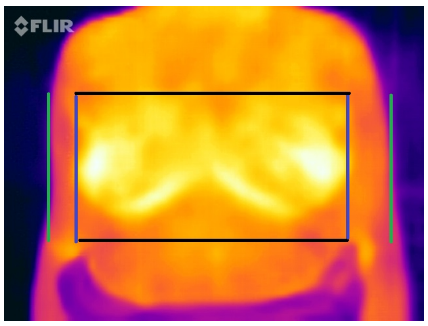


Figure 4.24: Graphically crop the second type of image from breast dataset

Since we want to obtain the same width in the cropped image we will color between green and blue line with the same color as the background, this way it will be more alike to the first type of images. After we crop the images with the algorithm described earlier we will get this result.

#### 4.6.4.2 Removing Artifact

We notice that in almost all images under the breast we have a zone that is very hot and is not necessarily part of the breast. This happens because the contact between breast and skin creates a hot area and therefore it will be displayed in the image, although we don't want to consider it in the estimation.

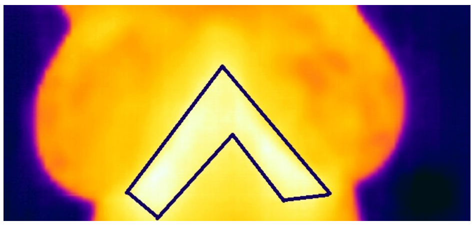


Figure 4.27: Defining artifact on breast image type 1

So if we verify that the image has it, we will look for the breast limits and the end of hot abdominal zone. Then we fill this zone with the background color so that the cluster will get this zone as well, this way the algorithm will consider the outside of body and this zone to not be consider.

We notice this zone in figure 4.4 and here's how we will remove it:

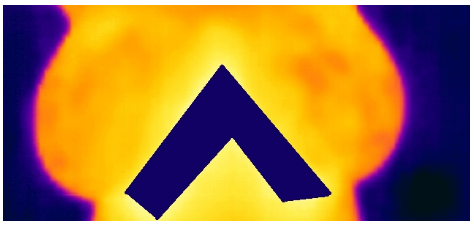


Figure 4.28: Removing artifact on breast image type 1

#### 4.6.4.3 Estimation of milk

In this final step, we will use FCM or FCMFP on already cropped images. We will define 3 clusters: background, hot zones and rest.

After we update the partition matrix that indicates the coefficient that a pixel has towards a cluster, we will use count how many elements has the cluster that represents hot area. We will do the same to the other cluster and ignore the background cluster.

After this is done we will simply divide between both of the numbers and as a result a estimation will be presented that corresponds to the average milk estimated.

Since we had limited time we were not capable of counting the units on each cluster and therefore not estimating the milk quantity.

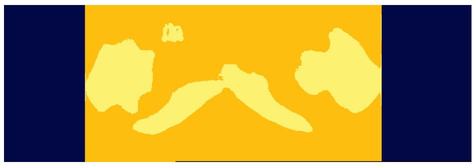


Figure 4.29: Result of FCM with 3 clusters in cropped breast image with artifact

This figure represents the result of our FCM if we didn't remove the artifact. What we are trying to demonstrate is that the calculations will not be accurate if we take in account this zone, therefore we present a segmentation where we tried to take this portion out, although not very successful.

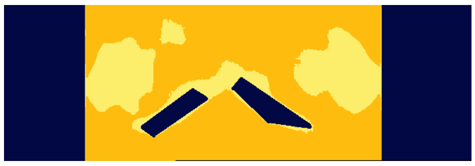


Figure 4.30: Result of FCM with 3 clusters in cropped breast image without artifact

The point here, even if it's not perfect, the purpose of this method is to reduce the quantity of pixels that may lead to error regarding hot zones and calculating a more exact quantity of milk.

#### 4.6.4.4 Defining Hot region

The next problem we had was to identify what region was the hottest and the intermediate region. We accomplish this assuming the most hot zone is white or in RGB values (255,255,255) and therefore we sum all coordinates and find the biggest one. We do the opposite to get the background and ignore it. We use the remaining one as the intermediate zone.

After this was done, what was left to do is to sum all the pixels that belong to each cluster and divide one by the other count.

## 4.7 Conclusions

We have presented the major results obtained using both FCM and the FCMFP. We think the results are astonishing as they prove to be far greater than any expectation on segmenting images although they don't reach the standards on some implementations of other algorithms. Also an additional investigation regarding woman's breasts, which prove to be a successful feat that can lead to applications that monitor milk quantity.

# Chapter 5

# Conclusions and Future Work

## 5.1 Main Conclusions

With the end of this work I respected much more image analysis. How can some algorithms have such a good accuracy since we still don't understand perfectly how the human eye and brain works? It was incredibly satisfying seeing the algorithm paint a similar image having seen the original one.

Regarding the implementation of the algorithms, they work although not very optimized. A low-scale example was done by hand, and every single calculation done was correctly made since it matched with the algorithm output. So, from a developer standpoint we feel proud to present it.

The experimental results can show the work that was put into this project, although, it does not give that much credit the hard work putted into. Since a image can be seen and interpreted in 10 seconds, the algorithm made to produce that result can take weeks if not more to be correctly implemented.

The learning lesson overall is that even if you are focusing on a specific task that seems that is not going to have a purpose or application, can be later used by someone doing something very different from you (in this case automatic image diagnosis) and your work may proven to be very useful.

Although the precision with all our algorithms are less efficient than others this helped me grow as a programmer and as a scientific researcher.

## 5.2 Future Work

As stated early in this document, one of the most important parts of the work was left behind, which was focusing in one particular disease and make a automatic diagnosis based on the interpretation of the image. Due to the time limit it was not possible to focus on this. Since segmentation and its study is capable of taking enormous time on implementation, nothing remains to be done.

What also could be interesting to implement would be neural network in order to train the algorithms on a data set and compare the results with the previous algorithms.

One of the greatest problems regarding the implementation of the algorithms was the execution time. Whenever was chose a high number of clusters or a very small $\epsilon$, the execution time would increase exponentially.

In order to solve this problem we could implement the code to run in a Graphics Processing Unit (GPU) to make the calculations faster. Or another alternative and perhaps simpler is to implement hash tables to make the program more efficient and reduce algorithm complexity.

We also didn't have time to code FCMFP and FCM to estimate milk quantity. It's fairly simple and it would be improved since we start with 4 clusters in FCMFP and would calculate the milk estimation whenever we face a cluster transition, so from 4 to 3 clusters and from 3 to 2 clusters. We would like to analyze differences in values in this transitions and why does this happens.

We also started by exploring different colors models, but at first we thought the results were bad, but since OpenCV uses RGB predefined. So maybe what we need to do is after doing the calculations in Cie Lab we should have converted to RGB to print in the screen.

# Appendix A

# Weekly Progress

This section documents what was made in every single week of the development of the project. It demonstrates the sequential progress of the project from start to finish.

- **Week 1** - Week of the 30th of January of 2017

  The first week I suggested my idea of for the project to the professor: a program that could analyze a MRI and could conclude if a certain person had a tumor in the brain.
  The work done in this week was searching for public databases of MRI. I also went to Hospital Pêro da Covilhã, Gabinete de Apoio ao Ensino in order to get part of their database. Although the people responsible towards this department didn't give access to the database they suggested we could make a meeting with a director of another department and gave me the contact so we could meet later.

  I also started to program in python since I knew I would be using it on the project.

- **Week 2** - Week of the 6th of February of 2017

  In this week I met with the professor since he suggested the project, we talked about what methods we could use to make this possible, giving me his insight on the area. Later on the week the professor sent me a link containing the grand challenges in Biomedical Image Analyses, the objective was that I would choose a specific challenge so we could use their database and publish my results if I had a good result. I continued to enhance my skills with python.

- **Week 3** - Week of the 13th of February of 2017

It was this week I met with the professor and agreed in all aspects of the project, I've also had my fist contact with the lab. At first I spent time on installing correctly OpenCV so I could use it with python programs. Afterwards I consulted the documentation of OpenCV and started to follow tutorials to open and writing images. Used Numpy library to do operations and manipulating in a matrix the RGB channels individually, this way I would be applying operations to the images.

In the final days created the latex standard model of the report to the project.

- **Week 4** - Week of the 20th of February of 2017

  In this week I continued to study more advanced OpenCV tutorials on python and writing the report. In the middle of the week I've been given an article related to my project. It was very helpful, since my knowledge in the area was very limited, it also made it clearer in how I could achieve making this program.

  I'll try to resume next what the article was about and what I learned from it.[4]

  This article starts be defining the concept on image segmentation and clustering approaches. They also present the most popular clustering algorithm FCM but given it's limitation (does not incorporate spatial context) we find a lot of FCM algorithm variations. At the end we are confronted with results comparing these algorithms. FCM, Spatial Fuzzy C-Means (SFCM),Kernel-Based Fuzzy Clustering with Spatial Constraints (SKFC) and Penalized Fuzzy C-Means (PFCM). The last is the one with better results (good computing time and less wrong classification).

- **Week 5** - Week of the 27th of February of 2017

  It was in this week I started developing in python clustering algorithms, specifically K-means algorithm. When I was almost finishing my solution, I met with the professor and he explained me thoroughly, some parts of the article I have previously been reading and wasn't understanding clearly. Then I tried to implement FCM algorithm, which was a bit different.In fact, pixels could be categorized in several clusters, and calculating the center of the cluster was also different so that we need to take in account the coefficient of the pixel towards a specific cluster.

- **Week 6** - Week of the 6Th March of 2017

  In this week i continued to develop the FCM algorithm, although I notice the structure I had in K means couldn't be the same in FCM. This is because in

K means algorithm I have a matrix and within each point I use a int to define which cluster belongs to the pixel. So in the FCM the pixel can belong to multiple clusters. I had two options here, I could either expand the matrix so that it's size were cluster * points but I choose to make a matrix just like before except the content is a list with the coefficients of the formula given. The list size is the number of clusters. After this structure was correctly made I adapted the coefficient calculations, and also the calculations to get the center pixel.

To test the algorithm I created a image with the dimension of the matrix and I coloured all the points based on the higher coefficient of cluster. Since the number of initial clusters of the image is chose before execution, it would be better to have some method to discover the best number of clusters. I have done some research and found among others a article that may be useful in the future. The article is the following, [2].

- **Week 7** - Week of the 13Th March of 2017

  In this week I tested the algorithm extensively in order to confirm it was well made. In order to do this I created another program that instead of reading an image, created a small matrix(3x3) so that the calculations on the formula were smaller and I could do it on paper to compare it later. After some time the results matched which implied my way of thinking and the algorithm itself were doing the same thought process. The only part I was unsure about were cases where the pixel of the matrix was exactly on the cluster coordinates. Should this point have value 1 in the membership matrix (have a strong correlation towards this cluster), or have value 0? Since I was unable to conclude from the article I figured I should ask the professor and also some more information to advanced topics of the project. After I got my answers I assumed whenever x points are exactly on the cluster coordinates the points have 1/x membership value towards that cluster and 0 value towards any other cluster. I tried to implement this process but I didn't confirm it yet.

  Another issue I was having in this project was that we weren't actually using the image properties and we were only considering the matrix row and column index, this way, any image we may use the result was not related to the image. After discussing this with the professor I changed the approach of calculating the membership value based on Euclidean Distance between matrix indexes to RGB values of the image. I also had to change the Euclidean Distance itself to do the calculations for 3 dimension coordinates and clusters so that they had 3 dimensions and vary from 0 to 255. When I did these alterations I had pretty amazing results as I could clearly see my image test with different colors within the different regions. This was done by painting a pixel with a defined color if the pixel had more membership value to this cluster. I also tried to convert my color model from RGB to CIE L*a*b*, in this case the results was unimpressive since all the membership matrix was one-sided as in a cluster would have more coefficient to all points, hence having my draw all from one colour. After a while I compared the pixel values of this color model to the CIE L*a*b* and I realized the process I was doing to convert the image from one color model to another was incorrect, since OpenCV method was not being precise. I decided to search and ultimately I found scikit-image library which made this conversion correct. After this was done, another problem was that whenever I tried to open the image in color space CIE L*a*b* a random image that resembled an abstract painting appeared. After some research I found out the function imshow of OpenCV opens strictly a image in RGB color space. So logically, if I try to open a Cie Lab image, obviously something strange will be the result. When I implemented the latest algorithm with the Cie Lab color space, using the coordinates a and b I got the result which may be slightly worse than RGB colorspace.

- **Week 8** - Week of the 20Th March of 2017

  This week I spent the time improving the report of the project, and trying to read more about the theory behind it. I also read the article from professor Paulo Fazendeiro,[2]. With the program I tested several images with different number of clusters to make conclusions.

- **Week 9** - Week of the 27Th March of 2017

  I continued the work done last week mainly in the report, and used the time to get results from tests I got from the project. I also optimized some parts of the code, as I can now test FCM algorithm with a maximum of 10 clusters.

- **Week 10** - Week of the 22Th May of 2017

  This week I gathered, as suggested by professor, a image with well-defined regions and tested it in the algorithm. The regions with RGB colorspace would stay well-define as expected and depending on the cluster number, different regions with similar color would be assembled. Lab colorspace overall got bad results, this may be due to only accounting to parameters a* and b*.

  I also began to develop FCMFP algorithm, increasing dimension to the cluster and points the image. I also adapted the formula to calculating the cluster center, accounting for this FP and a new constant Teta which can be described has the zoom of the image. What's left besides confirming what was done is to calculate the FP initial coordinates.

- **Week 11** - Week of the 29Th May of 2017

  This week I assembled all different parts of the report to make a big leap towards having a well written document. I also tried to contact Yong Yang, which is the author of [1], in order to request the image database used in his article. Although not having a response, I found out a link within the article which provided with MRI. Unfortunately the images are on a binary extension and therefore I could not test it yet.

  Relatively to the code of FCMFP the cluster number is staying the same after FCM runs, after some debugging each of the clusters have a point in which the partition matrix is the maximum towards that cluster and therefore it stays a relevant cluster. Since in all the clusters this happens, all clusters are relevant and the cluster number stays the same.

  At first glance the function seems to be written correctly, if the image property makes this phenom or a bug in the program, I must discuss it with professor.

  Relatively to the evaluation of correct pixels I have some difficulty on how should I implement this.

- **Week 11** - Week of the 5Th June of 2017 It was at this moment I started doing experiments with Xie-Beni index and gather the results. I also started implementing a inefficient way to calculate the SA, by comparing the position between the original image and the result from our algorithm.

- **Week 12** - Week of the 12Th June of 2017 This week I started to create a method of classifying pixels based on their human segmented image, only later in the week that was done the mapping explained in the results chapter in this report. I also gathered some tests using FCM using different $\epsilon$.

- **Week 13** - Week of the 17Th June of 2017 In this week I coded and verify the output of FCMFP algorithm, since I already implemented this earlier only a few bugs had to be corrected. More precisely the algorithm to decrement cluster number. The key here was to compare point to point the cluster that had more degree of membership towards that point, this is a relevant cluster. We excluded after we see all points the clusters that become irrelevant. Also we focused on getting experiments from our algorithms and improve the report. We had problems testing FCMFP since it consumed too much time and also we didn't know what value we should increment $\zeta$ to obtain good results. Regarding FCM we implemented this week SA since it had some bugs.

- **Week 14** - Week of the 24Th June of 2017 At this time we've been given a data set containing thermodynamic images of woman's breast with the purpose of estimating milk quantity, this was done running FCM or FCMFP with 2-4 clusters and divide the hottest zone to the other area of the woman breast.

  Later on the week I also gathered the best result of FCMFP yet, although it was bad overall, it was a start. I had help from João Neves to confirm SA using Matlab on my experiments.

  Tested FCM with normal images, images with segmented examples and breast images and gather the results.

  Finalization of the report.

# Bibliography

[1] Maria Manuela Areias da Costa Pereira de Sousa. Sebenta tecnologias multimédia. October 2014.

[2] P. Fazendeiro and J. V. de Oliveira. A fuzzy clustering algorithm with a variable focal point. In *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, pages 1049–1056, June 2008.

[3] Carlos S. Pereira, Luís A. Alexandre, Ana Maria Mendonça, and Aurélio Campilho. *A Multiclassifier Approach for Lung Nodule Classification*, pages 612–623. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[4] Yong Yang and Shuying Huang. Image segmentation by fuzzy c-means clustering algorithm with a novel penalty term. *Computers and Artificial Intelligence*, 26:17–31, 2007.