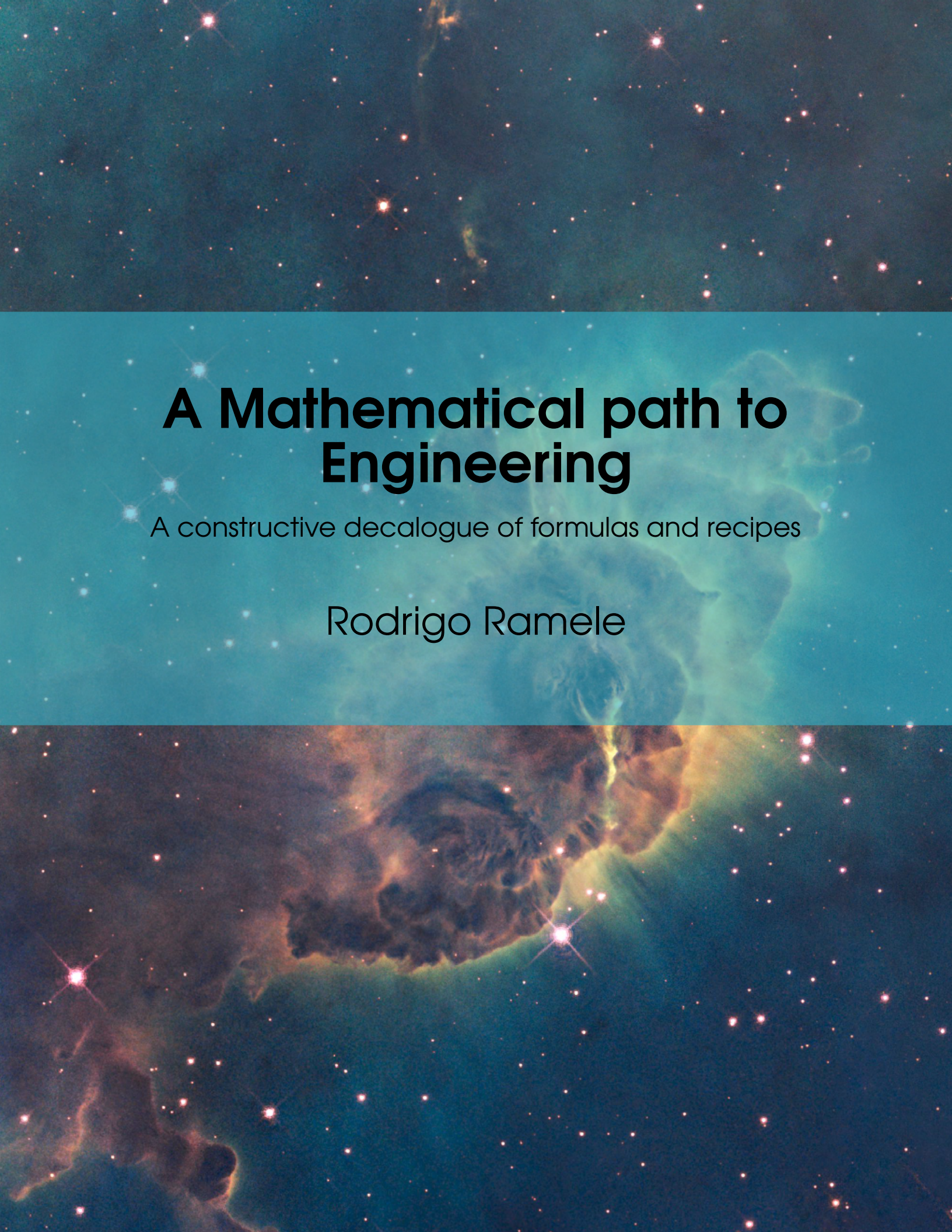


A Mathematical path to Engineering

A constructive decalogue of formulas and recipes

Rodrigo Ramele

SUMMER RESEARCH INTERNSHIP, UNIVERSITY OF WESTERN ONTARIO

GITHUB.COM/LAURETHTEX/CLUSTERING

This research was done under the supervision of Dr. Pauline Barmby with the financial support of the MITACS Globalink Research Internship Award within a total of 12 weeks, from June 16th to September 5th of 2014.

First release, August 2014

Contents

1. Introduction

1.1 Motivation

Sometimes we get lost on the true origin of Engineering. Which is the practical implementation of mathematics.

1.2 Objective

The purpose of this work is to establish in a step by step and constructive fashion a set of mathematical concepts from a pragmatic point of view, of mathematical ideas that are useful to engineering.

Everything that is written here is exactly what I understand. If anything is besides this notes, I just put it in the category of "Not-understanding".

What I found is that I came to this notes, many times, adding more information and connections to other things, and I will always understand the same things from a different and completely new perspective.

After a while I found that I will be very dissapointed of loosing those notes so I decided to write them down digitally. And have a backup with that.

How can you translate something that you created with your own pen ? from your own handwriting ? into something digital that is feel so cold to the naked eye.

Someone may found these notes useful. I believe that will be for two reasons: I wanted to create them in a constructive way: if I cannot find some rule to justify what I am doing then I am not doing anything, it doesn't exist. The second reason is that I always found mathematics to be hard. I love to understand it but it takes time to do it, to sit down. At the same time, "undestanding" is not binary. The best analogy I found is like a complex and evolutive graph or a tree-like structure, with many branches. Let's say you understand something. Then you have only "one path" to go from point A to point B of the concept. But you can only walk across that path, and you, somehow, can follow the train of thinking from one place to the other. However you cannot connect it to other things. When you teach, somebody may point out the existence of point C closer to both. But you were completely unaware of it until some student ask you something about it. Then you discover that connection (and manage to give a response to the poor student, which it will be satisfactory or just mumble jamble).

Then you start to discover many more ramificiations and your level of understanding increases. So understanding, is the level of ramification that you know of something, how much can you connect the concept to others.

for Control engineerings for instance, everything can be a control problem. The same can be applied to the "Rule of 3" or proportions or fractions. Or algebra. These very important tools allows you to tackle problems that is the reason they are so ubiquous. In such way the Consciousness theory share some intuition with these concepts, measuring the level of integration and connectivity (Rhythms of the brain).

1.2.1 References

Since I found so much good information about pretty much everything I wanted to know about, I will just create a remark and let you know where you can find more specific information about, just like below.

For more information about the cosmological principle, review Chapter 1: Why Learn Astronomy?, page 10, from **21st Century Astronomy**, *Hester* | *Smith* | *Blumenthal* | *Kay* | *Voss*, Third Edition, 2010.

2. Set Theory: preliminaries

2.1 First ideas

So, now here you have your first astronomy picture, [1] what do you see?, it is a monochrome image, with different levels of brightness, slightly big (8500 x 5000), it looks like a lot of stars making a spiral.

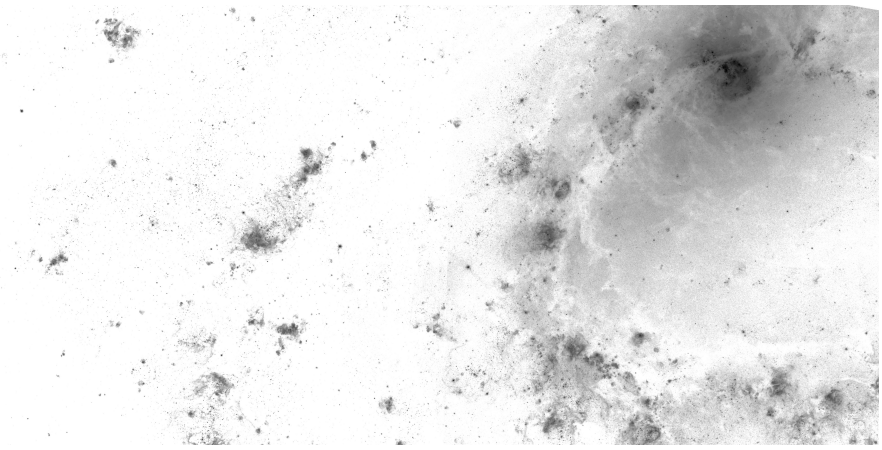


Figure 2.1: Picture of the M83 galaxy, image taken from the WFC3 ERS M83 Data Products, http://archive.stsci.edu/prepds/wfc3ers/m83datalist.html

How can we learn something about this image, quantize, get useful information? In the next subsections I will explain the first ideas.

[1] For example purposes the image selected is a picture of M83 through a Wide H-alpha and [N II] filter.

2.1.1 Superpixel segmentation

The main concept of this is to cut an image into bigger neighbourhood sections, so from an image that has $425x10^5$ pixels we can get maybe less than 500 superpixels, and then analyse separately those little sections and identify what kind of interstellar objects are they, look at image 2.2 it is a self-explanatory example of how a superpixel algorithm works. There are many ways to do this and

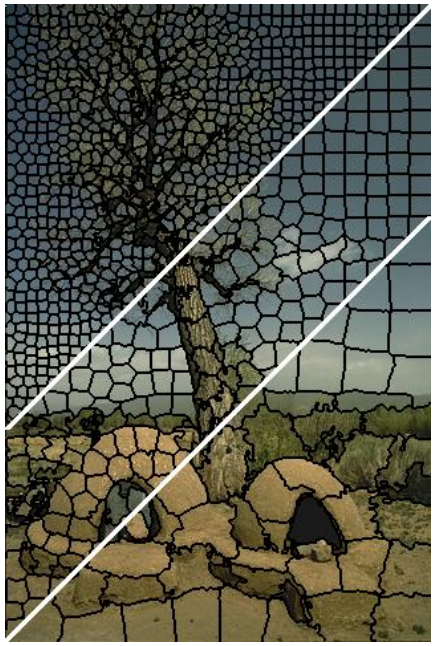


Figure 2.2: Example of a superpixel algorithm

they vary according to color dimensions, methods and number of required superpixels and whether the algorithm is able to find borders and make pixel classifications.

R You can find some example test I tested with Matlab and with Python in this web page: `https://github.com/LaurethTeX/Clustering/blob/master/Methods.md`, also there is a huge amount of information on the internet about this but here are two pages you might find useful:

- Superpixel: Empirical Studies and Applications
  `http://ttic.uchicago.edu/~xren/research/superpixel/`
- Segmentation Algorithms in scikits-image
  `http://peekaboo-vision.blogspot.ca/2012/09/segmentation-algorithms-in-scikits-image.html`

Also there is one article (from IEEE) I found about and might interest you, it's pure computer science,

- Normalized Cuts and Image Segmentation
  `http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf`

2.1.2 PCA

Welcome to Astronomy where you will find more acronyms than words to mention something on articles, lots of fun!, well in this case PCA stands for Principal Component Analysis, the objective of this method is to reduce dimensionality, transform the data to another space where is can be

manipulated and reduced, there are multiple examples of work that has been done in astronomy applying this technique.

Therefore, the idea of applying this method is that if we have multiple-wavelength images of the same target and transform them to PCA space then we will have less dimensionality and it will be easier to process all the data and fins valuable information.[2]

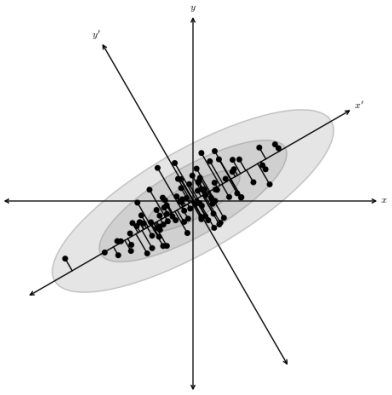


Figure 2.3: A distribution of points drawn from a bivariate Gaussian and centred on the origin of x and y. PCA defines a rotation such that the new axes (x' and y') are aligned along the directions of maximal variance (the principal components) with zero covariance. This is equivalent to minimizing the square of the perpendicular distances between the points and the principal components

R An example article, where they explain how to apply PCA on multi-wavelength images and also mentions the pros and cons of using it.

- Preserving Structure in Multi-wavelength Images of Extended Objects `http://arxiv.org/abs/1101.1679v1`

There's a whole section that talks about this subject with a machine learning approach as a preprocessing step in this nice book,

- Ivezić, Ž. and Connolly, A.J. and Vanderplas, J.T. and Gray, A., *Statistics, Data Mining and Machine Learning in Astronomy*, Princeton University Press, Princeton, NJ, 2014.

2.2 Hypothesis

Our data looks like the images on Fig.2.4, and it contains data from let's say a determined galaxy at different wavelengths, if we assume that the galaxy contains various regions that relate to interstellar objects that can tell, how stars are formed, where, how stars die, where was a star, and other mysteries, I guess we can assume that those certain regions can be identified because they share similar characteristics, the ideas is to find how a galaxy is made from, its contents, apply the concept of the superpixel idea in 3D superpixels.

Take the time to think about this, how the data looks like in 3D, how a star looks like in the data cube, imagine it, this is where ideas of how to tackle this problem come from.

[2]Before I forget to mention, later I discovered that PCA is not commonly used for data mining preprocessing because it is hard to interpret the information in the output result. Imagine clusters of data on PCA space, how do you make sense to that?

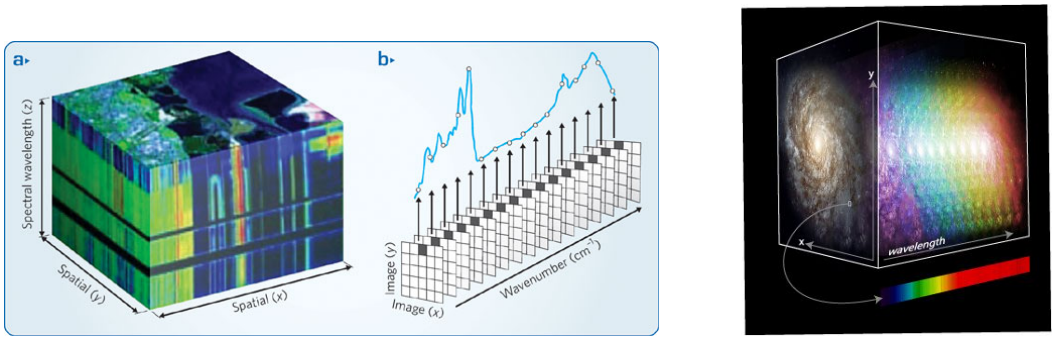


Figure 2.4: Illustrations of how a data cube looks like.

2.2.1 Topics you should review

This will require a lot of work, but hey it will be worthy and fun!

- Astroinformatics and computer science
  - Data mining
  - Machine Learning
  - Big Data Analysis
  - Neural Networks
  - Visualization Resources
- Statistics and Image Processing
  - Probability Density Function
  - Point Spread Function
  - Full width at half maximum
  - Convolution
- Interstellar medium and star formation
  - HII regions
  - Planetary Nebulae
  - Supernova Remnants
  - Molecular Gas
  - All kinds of Nebulae (e.g. dark, reflection)
  - AGN's (Active Galactic Nucleus)
- Astrophysics
  - Units (light-years, parsecs)
  - World coordinate system
  - Light
  - Telescopes
  - Stars and Stellar Evolution
  - Distance, Brightness, Luminosity
  - Galaxies

The GitHub page will certainly help you to understand why you need to learn about that, and where to find articles, web pages and books.

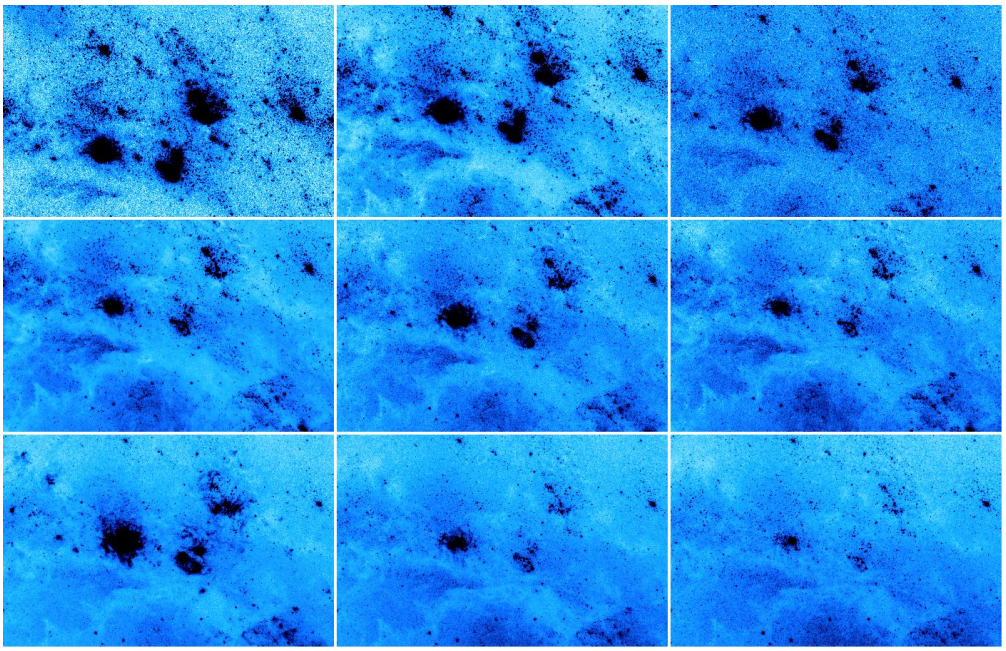


Figure 2.5: Example of how an object can look in 9 wavelengths

2.2.2 Downloading

First, let's equip ourselves with the basic software you will need in order to start then you may probably find other cool programs and later you will install them. There is also the possibility that your assigned computer will have them installed already but here is a brief description of what you can do with them, most of them are easy to use.

DS9: It is a program that visualizes astronomy images in FITS format (don't worry if you recognize this format, it will be explained later), where you can easily manipulate them, read their headers, compare, look at regions, see their characteristics, make graphs, even videos. Well, depending on what you need to use later you will be finding all the functions, the best way is to click everywhere and find out what happens, also you can ask to your astronomy colleagues they will tell you all the perks, or if you like learning by yourself or you need something specific check the documentation web page. It is fairly easy to install, just follow the instructions.

Download: `http://ds9.si.edu/site/Download.html`

Documentation: `http://ds9.si.edu/site/Documentation.html`

The picture below shows (Fig.2.6something cool you can do in DS9.

Python and a user interface: The most *limitless* and user friendly way to develop programs in Astronomy is using Python, there are many packages, modules, functions now available to help you in almost anything. Me, as an undergrad engineer I'm used to program on an user interface and not directly in a terminal. So, here I will explain you my own way of doing things.

I make my programs on the Canopy editor, it shows when and where you have programming error and warnings, and the interface is easy to learn, now to run, I open a terminal, go to the directory where my program is, type `ipython` wait and then type `run myProgram.py`, and

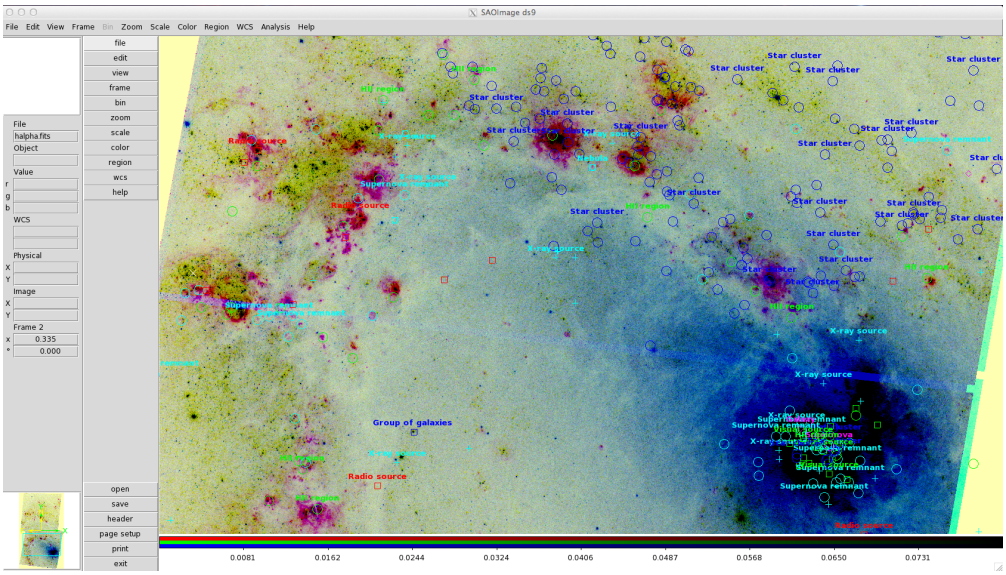


Figure 2.6: This is an RGB picture made from 3 independent FITS files, with a z scale and a region file overlaid from NED database, if you would like to learn more about this, or reproduce it, it is all explained in this web page: `https://github.com/LaurethTeX/Clustering/blob/master/NEDtoREGION-FILE/KnownRegions.md`

wait for the result.

Now there are a lot of fancier ways to work with *Python*, you can program and test directly using *IPython Notebook* on a web browser or you can just go for the terminal, use *nano* or *vi* or the text editor you like and then run it by typing `python myProgram.py`. At this point is up to you, but hey here are some links to start and the packages/modules you should install.

Interfaces or Development environments

- PyCharm, it a development environment, just like CodeBlocks or NetBeans `http://www.jetbrains.com/pycharm/`
- Spyder, actually this is the interface that comes with the Python distribution Anaconda, you will get the Python distribution and the interface. `https://store.continuum.io/cshop/anaconda/`
- Canopy, this is the one I mentioned before, it super easy to use and you can install packages with one click. `https://www.enthought.com/products/canopy/`

Modules

In Python, modules are like the libraries in C, therefore, to use math, astronomy and computer science tools you need to install them. To learn whether you already have a module installed or not, type on *iPython* `import andreaModule`, if the output result is something like `ImportError: No module named andreaModule`, you definitely don't have it installed.

The strategy here to install packages it fairly easy, find their website, go to the download section and follow the instructions, almost all the packages are available on the Python Packaging Index and may be installed by running:

```
pip install pyfits
```

To learn how to use them check the documentation page, user manuals or their API's, if you have experience on object oriented programming it will be like running a new bike and if you don't, don't worry too much, Python was designed to be easy to program, just learn the rules of the game.

- Astropy, this package is the *must have* of every astronomer, contains tools to handle coordinate systems, units, convolution.. well is better if you take a look at the web page. `http://www.astropy.org/`
- Numpy, this package contains the math magic functions, linear algebra tools and the array management variables, make sure you learn all about *Numpy arrays* you will work with them all the time. `http://www.numpy.org/`
- SciPy, well this package is the base of all scikit modules which contain the functions you will use in image processing and machine learning. `http://www.scipy.org/`
  - Scikit Image, contains image processing tools, it is the *OpenCV* for *Python* `http://scikit-image.org/`
  - Scikit Learn, contains data mining algorithms, pretty much contains everything that you will ever need. `http://scikit-learn.org/`
- Matplotlib, this package is probably one of the most powerful tools visualize data, you can draw almost anything you want and exactly how you want it. An example of that are the images of the AstroML book, you can access to the image library code and learn how they are made, this is the website `http://www.astroml.org/book_figures/index.html`.[3]. You can download the package here `http://matplotlib.org/`.
- PyFITS, in this package you will find tools to manipulate FITS files, create new ones, create image cubes, tables, and do all kinds of things with their headers. Certainly this package is more than useful. `http://www.stsci.edu/institute/software_hardware/pyfits`

In the path of researching I'm certain you will find more and new packages and by them you will be prepared to install anything.

Montage: This is a toolkit for assembling astronomical images into mosaics, but it has more functions that you may need in the future to prepare your data before processing it. There are two ways of installing and I would say that is better to have them both. One is to install the toolkit and any time you need it, you run the commands on the terminal, the other one is to install a *Python* module and use it just like any other module. To install montage for terminal, download the latest version in this website `http://montage.ipac.caltech.edu/docs/download.html`, **read the README file** or go to this website `http://montage.ipac.caltech.edu/docs/build.html` and follow the steps, now if you don't have any problem installing it, you can try testing it with an example program found on this website `http://montage.ipac.caltech.edu/docs/pleiades_tutorial.html`, in case you are having trouble and your computer is a MAC, instead of doing step five (*If you want to be able to run the Montage executables from any directory*), try this:

1. Open a file called `.profile` located in your user folder. (e.g. `/Users/Laureth`)

```
$ vi .profile
```

[3] Statistics, Data Mining, and Machine Learning in Astronomy book, it was mentioned before

2. Include in the file the following

```
export PATH=/Applications/Montage_v3.3/bin:$PATH
```

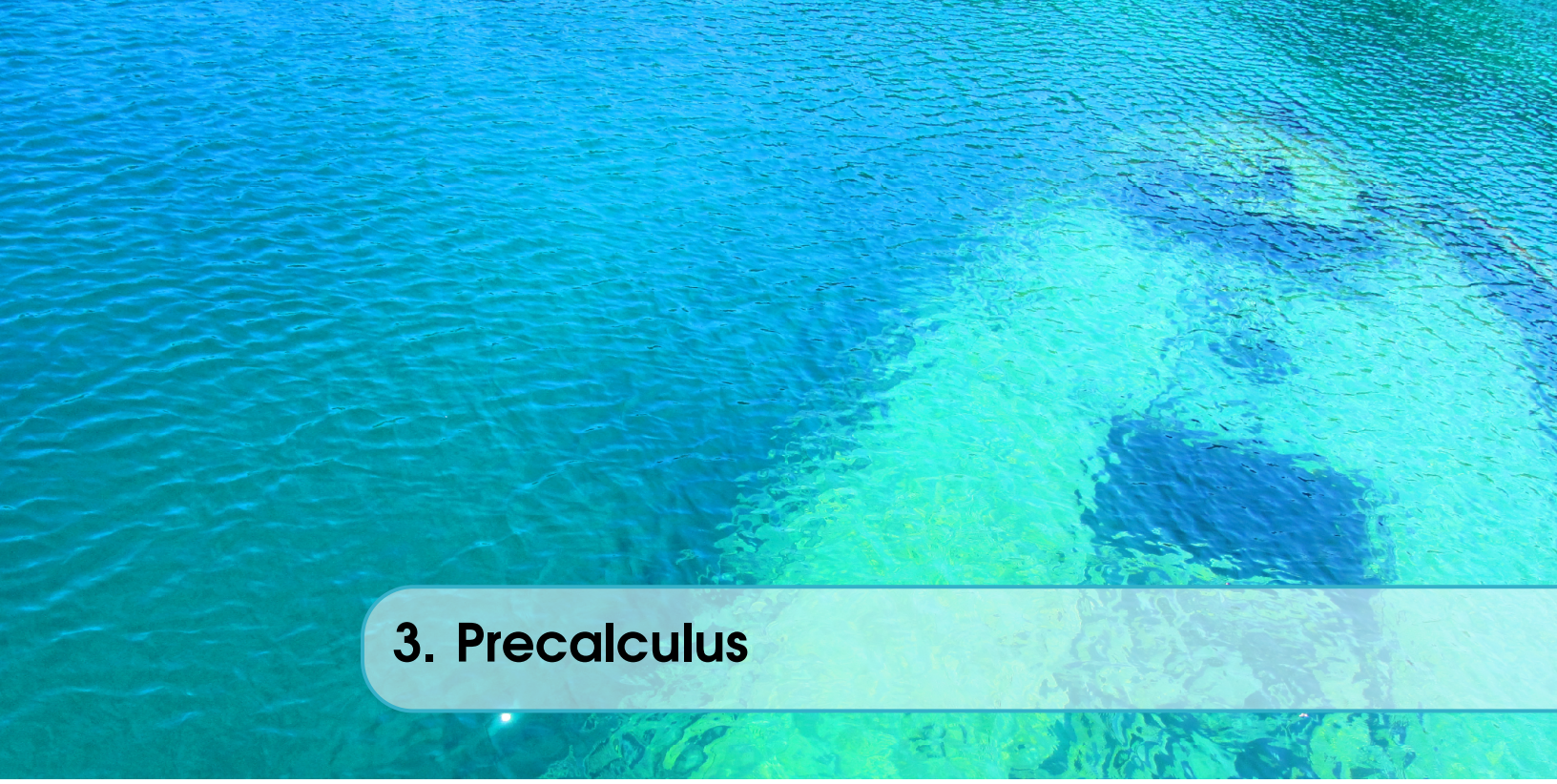
In this link (`https://github.com/LaurethTeX/Clustering/blob/master/Tools.md#the-profile-file`) you will find an example of how this file should look. After you modify it, make sure that you save it and type in `/Users/Laureth`,

```
$ source .profile
```

Then try testing the *Montage* commands, and I'm sure that it will magically work, just remember that any time you use any command, type `source .profile`.

Now the other way to install, implies only to install a *Python* module but this module contains less functions that the terminal application, in any case check the website `http://www.astropy.org/montage-wrapper/`, there you will find all the documentation you may need and the instructions to install it (*Spoilers* `pip install montage-wrapper`).

Any questions you may have and how to install, here is my GitHub page for software tools `https://github.com/LaurethTeX/Clustering/blob/master/Tools.md`



3. Precalculus

Before continuing, first and most importantly you must select the *raw* data you are going to process and later after you acquire experience with an specific dataset the idea is to expand the algorithms to any kind of dataset. The important things are to learn how to input the data correctly, establish the right *learning parameters* in the selected algorithm and find the best way to visualize your results and interpret them correctly.

Now let's start with basic concepts that vary from an engineering to an astronomer point of view.

3.1 What is an image?

As you may know, an image is a matrix of numbers that contains the specific brightness level that corresponds to a given pixel. And from there the concepts evolves and adds channels of colour and depth. But for now, let's just think about monochromatic images (only one channel). In Astronomy, images are usually considered sets of scientific data, observations, that contain information about an specific target in the sky seen through an specific filter and the levels of brightness correspond to the behaviour of the optical sensor (CCD camera) in relation with the number of electrons that hit a particular pixel through an specific waveband. Something else to consider is that the sky is not flat with this I mean that the celestial vault is like a sphere surrounding us therefore Cartesian coordinates are not the parameters used to identify points in space, there is another system called WCS (World Coordinate System) hence a conversion between pixels and WCS coordinates exists. As you are realizing now just one image can contain tons of information related to it, now imagine that multiplied for terabytes and terabytes of stars, galaxies, planets, nebulae or any object in space. Fortunately in astronomy this is solved using an image format that contains the image and its own information.

3.1.1 FITS files

This format is the standard data format used in astronomy, can contain one image, multiple images, tables and header keywords providing descriptive information about the data. The way it works is that this format can contain a text file with keywords that comprise the information about the

observation and a multidimensional array that could be a table, or an image, or an array of images (data cube). This files can be managed in different ways, with an image preview use DS9, for handing the data in a program use the *Python* package *PyFITS*.

3.1.2 WFC3 ERS M83 Data Products

The selected dataset to test the data mining libraries I found is a series of observations of M83 at 9 different wavelengths, the original images can be found in this web page, `http://archive.stsci.edu/prepds/wfc3ers/m83datalist.html`, the specific information about them can be found in Table 3.1. This particular images were observed through HST with the WFC3/UVIS camera.

Filter / Config.	Waveband / Central λ/ Line	Obs. Date	Comment
F225W	UV filter / 235.9 nm	26 Aug 2009	UV wide
F336W	UV filter / 335.5 nm	26 Aug 2009	Strömgren *u*
F373N	Narrow-Band Filter / 373.0 nm	19 Aug 2009	Includes [OII]
F438W	Wide-Band Filter / 432.5 nm	26 Aug 2009	*B*, Johnson-Cousins set
F487N	Narrow-Band Filter / 487.1 nm	25 Aug 2009	Includes Hβ
F502N	Narrow-Band Filter / 501.0 nm	26 Aug 2009	Includes [O III]
F657N	Narrow-Band Filter / 656.7 nm	25 Aug 2009	Includes Hα+[NII]
F673N	Narrow-Band Filter / 676.6 nm	20 Aug 2009	Includes [SII]
F814W	Wide-Band Filter / 802.4 nm	26 Aug 2009	*I*, Johnson-Cousins set

Table 3.1: Summary of Observations

3.2 Preprocessing your data

This section is where you prepare your data to be processed, you have to make sure that all your images have the same grid size, same spatial resolution, less possible quantity of outliers and noise and same coordinate system. Now, what are those things? Same grid size means that your images must have the same pixel size, in the dataset we are processing we don't have to worry about this, the pixel size is 0.0396 arc sec/pixel. Now, spatial resolution, each image has it's own spatial resolution depending on the filter that was used to get the observation, the number that you will be looking for is the FWHM that describes the PSF of every image. When you have all the FWHM for all the images you should choose the largest which corresponds to the poorest spatial resolution and create a convolution kernel with *Tiny Tim* or use a Gaussian kernel calculated with *Astropy* and convolve all the images with that kernel. This exactly what I did, if you look at image **??**, you will see the before and after convolution. In table 3.2 you can see how I chose the number for the FWHM.

After convolving all the pictures, I started to do some tests, but I realized that maybe around 30% of the images was missing information and/or noise and the results I was getting were mislead by the outliers. In clustering algorithms we must help the algorithm, make sure that what we are inputting is something that can be clustered, although some of them are *shielded* against outliers, making our data more accessible and easy for the neural networks to interpret will help you to get better results, as you can see in image 3.2 (open one and explore it in DS9) there is missing information and noise. In order to correct this I decided to go with the easiest way I could think of, just cut the image. And I did selected a processable area excluding all the missing information and noisy areas.

The next step was to build the data cube, at this point you can decide if you want to process your images independently or all of them. The ideal here is to input all of them in a data cube, so the

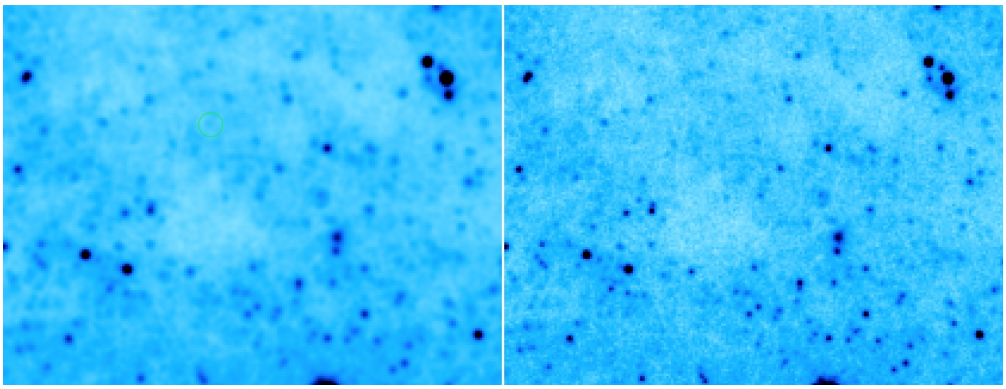


Figure 3.1: In this image you can observe how an observation looks, before and after convolution, this particular image corresponds to the B band filter and was convolved to a 0.083 arc sec FWHM

Filter / Config.	Central λ	FWHM (arc sec)
F225W	235.9 nm	∼0.083
F336W	335.5 nm	∼0.075
F373N	373.0 nm	∼0.070
F438W	432.5 nm	∼0.070
F487N	487.1 nm	∼0.067
F502N	501.0 nm	∼0.067
F657N	656.7 nm	∼0.070
F673N	676.6 nm	∼0.070
F814W	802.4 nm	∼0.074

Table 3.2: WFC3/UVIS PSF FWHM informations for the selected dataset, as you can see the largest number here is 0.083 which means the poorest spatial resolution, this is the number used to calculate the convolution kernel, in order to precess them all images must have the same spatial resolution.

output clusters relate information from all the wavelengths and the regions covered by them can be interpreted more easily. Now if you choose to create an image cube (just append the image arrays in one FITS file) it is possible that your images have a different conversion between their world coordinate system to pixel, so have to make sure all of your images are projected with only one conversion, this mean that you have to re-project them to a common WCS.

Well, what I wrote before it is a brief summary of what I did, but I'm sure that you can find a better way to do your own data pre-processing but here are some things that you should consider:

- Create a method as general as possible, with input parameter that can be adapted to any kind of data, this will save you a lot of work in the future
- Understand first your algorithm, how the data is going to be processed and design the best way to input your data
- Accommodate your data according to the type of attributes that the algorithm can handle
- Consider the size of your dataset, if it's huge your program may never end
- Find out of your algorithm can work with high dimensional data (multi-wavelength), because if not, you won't be able to input data cubes
- Find out if your selected clustering algorithms is able to find clusters of irregular shapes, this

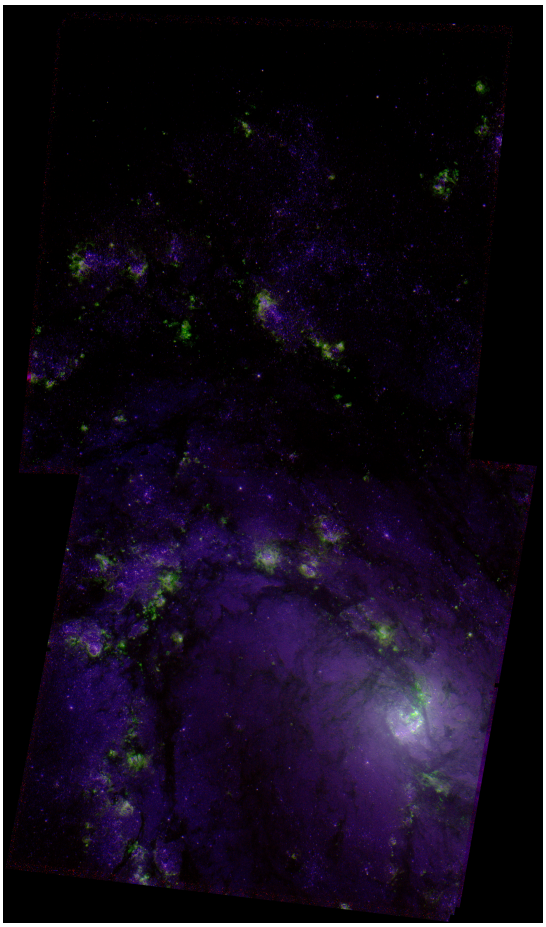


Figure 3.2: Look at the image, it is composed of two mosaics, therefore, there are some regions with missing data, now look at the borders of each mosaic there is noise near the edges, this is data that we don't want messing with our clustering algorithm and can be classified as outliers, it is very important to reduce them as much as possible so the output clusters can be correctly classified and correspond to the information that we are looking for

will help you to device the best way to accommodate your patterns

- Handle outliers, if you identify them, know where they are, try to eliminate them as much as possible, we don't want them messing with our clusters
- In case that you come up with an artful mathematical method like PCA to reduce dimensionality, make sure that what you input can later make sense when is clustered, because you will be working in another space
- Remember that the most important goal is to find hidden knowledge therefore, you must know you to visualize and interpret your results
- For the let's call it *astronomy image processing*, make sure that your data is scientifically approved ask people around you.

This section is explained at length in the GitHub page, there you will find my codes and some helpful links, `https://github.com/LaurethTeX/Clustering/blob/master/Preprocessing.md`

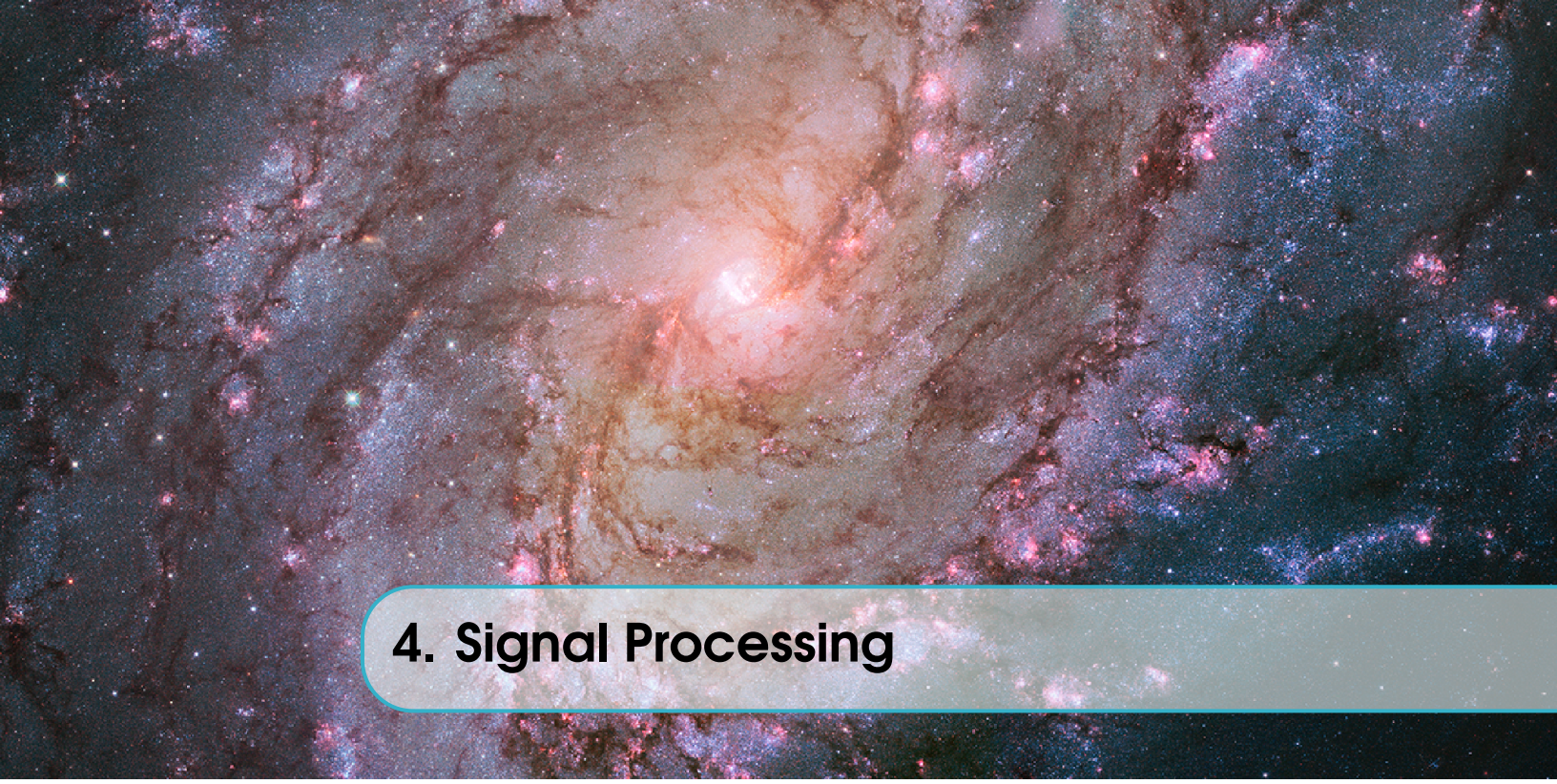
3.3 Software available

For doing data preprocessing there are a bunch of software available, even there is one being developed by Sophia Lianou called *imagecube* which, when it is finished, will be one of the best, has everything you need in one package. I'll say that this part is yours to discover, everyday there are more and more being released or new versions of the existent ones but in the meanwhile it will depend entirely on you, which software you want to use. For *Python* all the functions you will need can be found in the *Astropy* module, **check the API!!!.**

This specific part is all explained in GitHub in this link. `https://github.com/LaurethTeX/Clustering/blob/master/Preprocessing.md#first-step-data-pre-processing`

Some links to start,

- Astropy, Convolution and filtering, `http://docs.astropy.org/en/stable/convolution/index.html`
- AstroDrizzle: New Software for Aligning and Combining HST Images, With Improved Handling of Astrometric Data, `http://drizzlepac.stsci.edu/`
- Tiny Tim HST PSF Modelling, `http://www.stsci.edu/hst/observatory/focus/TinyTim`
- IRAF, Image Reduction and Analysis Facility, `http://iraf.noao.edu/`



4. Signal Processing

I discovered surfing on the internet a cloud computing software that is free, has data mining algorithms embedded, is specifically developed for Astronomy and is programmed by Caltech, University Federico II and the Astronomical Observatory of Capodimonte. The homepage website, `http://dame.dsf.unina.it/index.html`. Well, the platform for testing is ready!, now what? I requested and account and the next day they sent me an acceptance with my user name and my password approved. I introduced myself to the documentation, the available clustering functions, the manuals for every method, the blogs and discovered that the was one method available that could work with data cubes and do its clustering on every pattern (number in the multidimensional matrix) which was exactly what I needed. The name of this method is ESOM (Evolving Self Organizing Maps) and I read its manual, did some foolish test with all my image and ... never got a result ... the experiment ran forever (more than two weeks), when I realised that this wasn't the best way to tackle this problem I started considering only clustering on the independent images and not in the data cube due to the fact that the dimensionality was immense. So, in the end my selected methods have some results but not all, here is where all the work has to be done, analysed and tested again.

4.1 Methods Selected

4.1.1 ESOM, Evolving Self Organizing Maps

The *official* manual for this method can de found here, `http://dame.dsf.unina.it/documents/ESOM_UserManual_DAME-MAN-NA-0021-Rel1.2.pdf`, there you will find a full explanation of the method, the meaning of every variable and the supported file types.

Here is my own explanation of how this particular method works, first of all, can be used as an unsupervised machine learning technique or you can help the algorithm to identify regions an make it a supervised machine learning technique, this type of clustering finds groups of patterns with similarities and preserves its topology, starts with a null network without any nodes and those are created incrementally when a new input pattern is presented, the prototype nodes in the network compete with each other and the connections of the winner node are updated.

The method is divided in three stages, *Train*, *Test* and *Run*. The first step to experiment with this

method is Train. Here, the important variables to understand an look at are, the learning rate, epsilon and the pruning frequency. It is highly recommendable that you check the DAMEWARE manual for this function, there they will explain in detail the meaning of each on the mentioned variables.

Expected Results

This particular method as I mentioned before supports data cubes and considers as an independent pattern all the numbers in the multi-dimensional array this means that our clusters are groups of patterns with similar characteristics, that correspond to volumes of similar fluxes of electrons inside the data cube.

The output files from the experiment that will show us our results are,

- *E_SOM_TrainTestRun_Results.txt*: File that, for each pattern, reports ID, features, BMU, cluster and activation of winner node
- *E_SOM_TrainTestRun_Histogram.png*: Histogram of clusters found
- *E_SOM_TrainTestRun_U_matrix.png*: U-Matrix image
- *E_SOM_TrainTestRun_Clusters.txt*: File that, for each clusters, reports label, number of pattern assigned, percentage of association respect total number of pattern and its centroids.
- *E_SOM_Train_Datacube_image.zip*: Archive that includes the clustered images of each slice of a data cube.[1]

The file that you will be looking forward to see is the last one, the zip where you will be able to see the slices of the volume, and how the final configuration of the clusters was arranged.

Failed and still running tests: What no to do and what is still running

The first tests I did included all the complete data cube, including the areas where data was missing, the images were only re-projected and convolved. That was before realising that outliers might affect the ability of the algorithm to identify the clusters and distract them with noise and missing data. So, the first thing you must NOT do, is to get rid of the outliers when you are training your network, if you ever get to have a well trained network then it might be interesting to learn how the network interacts with noise an outliers, but for now we will help her a bit.

In table 4.1 are the input parameters I used to the failed tests applied in the *raw* data cube, and in table 4.2 are the input parameters used on experiments that are still running since August 7th, 2014. (I wonder if they will ever end)

Some of the failed experiments had histogram like the one you can see on figure 4.1 where the clusters were created but reached a point where the neural network could not define how to differentiate a cluster from another cluster and failed.

Hey, if you were wondering why I always choose to normalize, and one as the input node, well the normalization is due to the fact that I know that the data has, according to its filter, all kinds of ranges of fluxes on every layer which means that the distances between patterns might not be correct, this is a topic you should look into. And for the input node I choose 1 because if I start with any other number the experiment automatically fails, and of course we do not want that.

As I progressed and saw the results and the *log files* in all the failed experiments I decide to try the algorithm on independent layers and see if I could get something. Therefore I selected the Hα convolved observation (halpha_conv.fits) and did some tests on it, table 4.3 shows the parameters I used for the failed experiments and table 4.4 shows the parameters of the still running experiments.

[1] I have my doubts whether this file is produced or not, in none of my test was produced, you might need to contact the developers and ask about this.

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
Train2	1	1	0.3	0.001	5
Train3	1	1	0.7	10	100
Train4	1	1	0.95	1	10
Train5	1	1	0.99	0.1	10
Train6	1	1	0.01	0.01	1
Train7	1	1	0.5	0.7	5
Train8	1	1	0.5	0.5	7
Train11	1	1	0.25	0.00001	10

Table 4.1: This table describes all the failed experiments done in the workspace WFC3 with the *raw* data cube as an input, using the ESOM method in the DAME platform selecting the number 3 as the dataset type and without using a previous configuration file.

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
Train9	1	1	0.3	0.0001	5
Train10	1	1	0.99	0.0001	10
Train12	1	1	0.5	0.0001	5

Table 4.2: This table describes all the experiments done in the workspace WFC3 that are still running since August 7th, 2014 with the *raw* data cube as an input, using the ESOM method in the DAME platform selecting the number 3 as the dataset type and without using a previous configuration file.

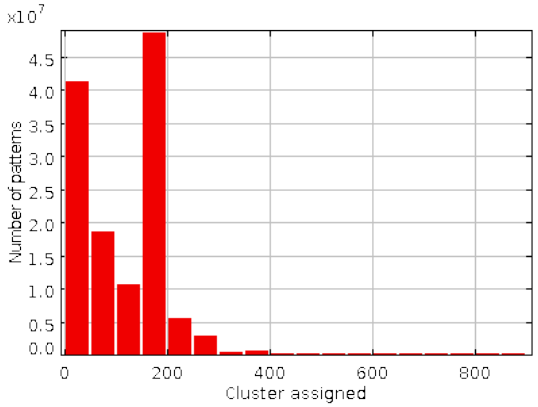


Figure 4.1: In this particular experiment, the neural network failed due to a very low pruning frequency, high number of patterns and all the outliers inclusions.

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
TrainHa1	1	1	0.5	0.01	5
TrainHa2	1	1	0.5	0.001	5

Table 4.3: This table describes the failed experiments done in the workspace WFC3 for the *halpha_conv.fits* file, using the ESOM method for one layer in the DAME platform selecting the number 3 as the dataset type and without using a previous configuration file.

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
TrainHa3	1	1	0.5	0.0001	5

Table 4.4: This table describes the still running experiments since August 10th, 2014 in the workspace WFC3 for the *halpha_conv.fits* file, using the ESOM method for one layer in the DAME platform selecting the number 3 as the dataset type and without using a previous configuration file.

My next mental step was to repeat the tests eliminating as many outliers I could reduce, my hypothesis here is that, if I eliminate all the areas where there is missing data and noise, the neural networks will be concentrated only in the patterns I'm interested in clustering and maybe identifying interesting regions that correspond to some known interstellar object. So, what I did was to try the ESOM algorithm with, again, independent images, this time I decided to apply the same experiment to three different layers, Hα, UV wide and *i*-band. In table 4.5 you can see the parameters of the failed experiments and on figure 4.2 there are some of the output histograms. Also, in table 4.6 you can see the input parameters of the still running experiments.

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
Train1	1	1	0.5	0.001	50
Train2	1	1	0.5	0.01	50
Train3	1	1	0.5	0.1	100
Train4	1	1	0.5	0.001	100

Table 4.5: This parameters where used in three different workspaces (*halphaCrop, uvwidecrop, ibandcrop*), with their own input file that corresponded to the convolved and cropped observation of each filter (halpha_conv_crp.fits, uvwide_conv_crp.fits, iband_conv_crp.fits), all of the experiments had no previous configuration file and the dataset type was 3 and all failed.

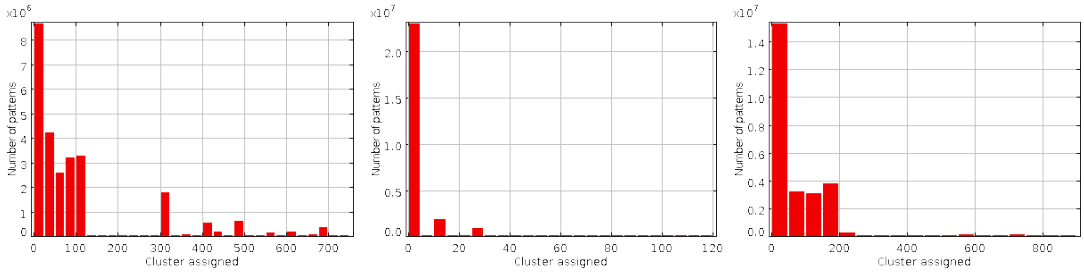


Figure 4.2: The histogram on the left corresponds to the halpha workspace in Train1, the one on the center to the iband workspace in Train3 and the one on the right to the uvwide workspace in Train2, all of them were failed experiments.

As you can see, I discovered that if I choose an epsilon of 0.0001 the experiments will be still running, and all of the other variables can be variated like the learning rate and the pruning frequency.

The big and small re-projected data cube

After a few days of waiting anxiously for the experiments to end and not getting any new results I decided to test the convolved, cropped and re-projected data cube including all the layers with a

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
Train5	1	1	0.5	0.0001	100
Train6	1	1	0.99	0.0001	75

Table 4.6: This parameters where used in three different workspaces (*halphaCrop, uvwidecrop, ibandcrop*), with their own input file that corresponded to the convolved and cropped observation of each filter (halpha_conv_crp.fits, uvwide_conv_crp.fits, iband_conv_crp.fits), all of the experiments had no previous configuration file and the dataset type was 3. The experiments mentioned are still running since August 11th, 2014.

fixated pruning frequency of 0.0001, hopping that this time I could get some interesting results. The input parameters for the two experiments I tested can be seen in table 4.7.

Name	Input nodes	Normalized data	Learning rate	Epsilon	Pruning Frequency
ESOMtrain1	1	1	0.5/0.75	0.0001	100
ESOMtrain2	9	1	0.75	0.001	100

Table 4.7: This parameters where used in two different workspaces (*Data Cube, RPDataCube*), the first experiment is still running since August 12th, 2014 and the second failed. The input for the Data Cube workspace corresponds to a 9 layer data cube with no re-projection and the RPDataCube input is the same data cube but re-projected.

As you can see, in the experiment *ESOMtrain2* I tried to start the neural network with 9 nodes (thinking logically as having 9 layers in the data cube) and immediately the experiment failed, so **do not try to input a number different than one.**

I waited 17 days for the experiments to finish (I did some other stuff in the meanwhile, most of the time learning new things) but I did not get any results so I came up with a different strategy, selecting small data cubes with already identified regions by the NED database. I selected randomly a particular HII region located in RA 204.26971, DEC -29.84933 (See figure 4.3) and centred it in a 605x605 pixels sample.

This time, most of the experiments gave me immediate results failing or finishing. On table 4.8, you can see the input parameters and the status of the experiments I tested with the small data cube.

Name	Normalized	Learning rate	Epsilon	Pruning Frequency	Status
ESOMtrain1	0	0.5	0.001	50	Running
Train2	1	0.5	0.0001	50	Ended
Train3	1	0.5	0.1	50	Ended
Train4	0	0.5	0.0001	50	Running
Train5	0	0.95	0.0001	100	Running
Train6	1	0.99	0.001	50	Ended

Table 4.8: All the mentioned experiment belong to the SmallDataCube workspace, have 3 as data type and one input node, no previous configuration file and the input file is *rp_small_datacube.fits*.

In this case three of the experiments ended and none of them failed (yet), here I detected that the output file that contains the distributions of the clusters on every layer is missing, but we got some interesting results, in the next figures (4.4,4.5) you can appreciate better what I'm taking about.

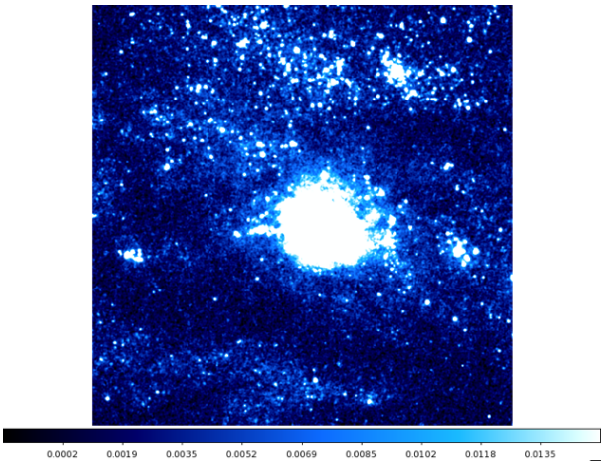


Figure 4.3: Illustration of the randomly chosen HII region for the small sample from the M83 re-projected data cube.

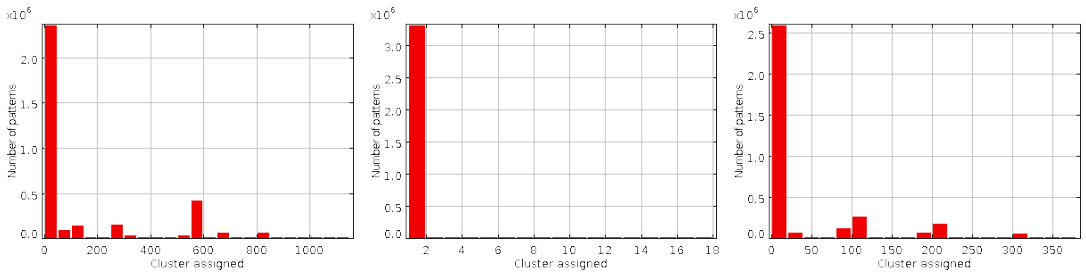


Figure 4.4: All of the images correspond to histograms of the ended experiments mentioned above in order (Train2, Train3, Train6), as you can see there is a predominance on one of the clusters that can mean that is detecting the HII region or the experiment never started, to understand further the results a visualization of the clusters is needed.

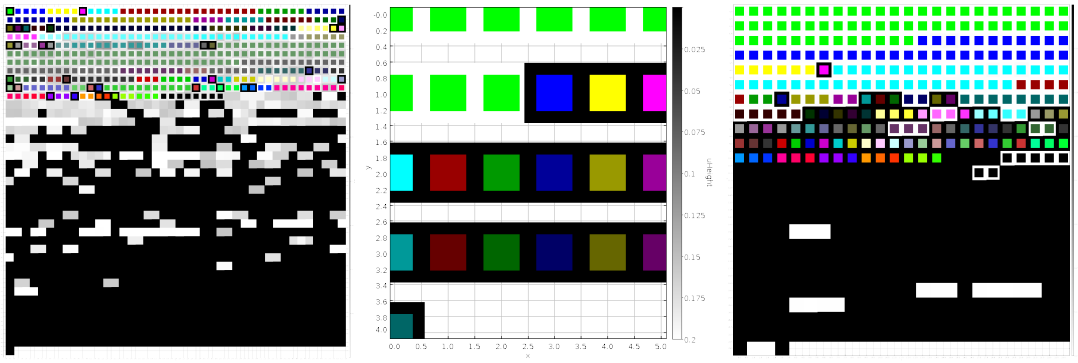


Figure 4.5: All of the images correspond to U-matrices of the ended experiments mentioned above in order (Train2, Train3, Train6)

There is work to be done for this cases, understand what is going on and interpret correctly the results, but last we got some.

4.1.2 CSOM

Well, as I mentioned before I did some tests using the ESOM method but since I wasn't getting any results I thought of testing this method, as always I strongly recommend to read carefully its manual, `http://dame.dsf.unina.it/documents/SOFM_UserManual_DAME-MAN-NA-0014-Rel1.1.pdf` and fully understand what is going on behind the curtains. In the meanwhile, this is my own explanation. This method uses FITS files, does not support data cubes, specifically uses a neighbourhood function in order to preserve the topological properties of the input space, it is a type of artificial network and is mainly unsupervised learning and produces a low dimensional discretized representation of the input space of the training samples. I in this case you can choose the number of clusters/neurons in the first layer (neural network), the diameter, number of layers (in the neural network), learning rate and variance on each layer. Here you have more input parameters to control.

Expected Results

Well in this case, since only FITS images are allowed, what we expect to find are areas identifying the different objects in the interstellar medium.

The important results in this case, are got in the *Run* and *Test* steps, in the *Train* step only the network configuration is outputted. What we are interested on seeing are the plotted clusters.

Tests

In this case I did some tests on the CSOM workspace, but none of the, where successful, too many input variables to control and test. So, in this case I will leave this parameters free for you to try. I do believe that this method could be very useful and if you find a way to input the data cube in a different configuration you will get some interesting results, due to the fact that in this method the preservation of the topology is one of the main principles.

4.2 Further work

Well, finally we reached the point where I my time in Canada finished and I this research is still on its first stages. I have so many ideas of how to explore the clustering techniques in the DAME platform, MatLab, Python and everything else that can be tested.

4.2.1 Some interesting ideas

For now, I would say that your best chance here, is to device an efficient way to input the information contained in a data cube as a list of points with values and reduce its dimensionality by randomly choosing them on every layer. If you are ever stuck, or no new ideas come to your mind, do not hesitate to contact me I might have a new interesting idea you can test.

4.2.2 Links you should check out

Most of them are listed in the useful resources section of The Caltech-JPL Summer School on Big Data Analytics, the web page `https://class.coursera.org/bigdataschool-001/wiki/Useful_resources`, you may need to create an account in Coursera and enrol in the course. And the rest of them are located in the References section on my GitHub page, `https://github.com/LaurethTeX/Clustering/blob/master/References.md`.

Wish you all the best, Andrea Hidalgo

5. Number Theory

6. Pragmatic Control Theory

That thing you are doing is proportional control.