POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree in Telecommunications Engineering

Master of Science Thesis

Optimizing Unimap, a practical astronomy image map maker

Supervisors
Prof. Roberto Garello
Prof. Michele Elia

Candidate
Navid Barati Moghadam

Student ID: s186767

December 2017

Abstract

Astronomy has always been one of the most interesting fields of study for human. From the era which stars were the only guide u to now that we launch satellites to observe the space more accurately, finding the secrets of the sky was an interesting task for people. The science of observing the sky is not limited just to unlocking the secrets of the other planets and ways, it's a key for finding the origin of the world and how the earth was created. Studying the stars which are far from us is actually studying the history. When we are observing a planet which is a million light year far from us, actually means we are studying what happened to that star 1 million year ago and now we are receiving the information. This fascinating fact that we can study behaviors of space of millions of years ago makes it more possible to find the secrets of planet earth and milky way millions of years ago. Since the human started paying attention to science, observing the sky and documenting the observations was among the highest priorities. They first started using the naked eyes and then started using instruments which help them see things which they could not see before.In 1609, an Italian astronomer, Galileo Galilei, pointed a telescope toward the sky. It is said that he was the first to do so. With it, he saw mountains and craters on the moon, and the Milky Way. Scientists kept working on this science and the ground telescopes are the result of this procedure. Among the most advanced versions we can name Keck Observatory in Hawaii , Large Binocular Telescope in Arizona. They have been observing the sky since 1993 and 2005 and won't stop doing it any time soon. Ground telescopes are useful and popular in many aspects. Among the most important ones we can name No limitation in size and weight and cost, but they were not enough for the scientists. They needed to see more and better, so they started to build something which can fly. The first ones were started in 1060s. They were simply some telescopes attached to giant balloons for one single goal: "observing the sky sharper", but that was just the beginning of the story. The story of space, the story of observing what we have started thousands of years ago when looking at the stars and tried to find the path, the story of unlocking the secret of World creation. They started something which is now a combination of several engineering fields for several goals, asking about the most important one? "Observing what we can't from the earth". The space telescopes are here to change the game, they have

brought the space observatory into the next level. Today we have got several space telescope functioning in the space. They have different instruments to observe and each of them have got it's own technology to record the data, and then there is the need of the softwares to convert these raw data to information. One of the space observatory launched in the space is called Herschell. It had been launched into space in 2009 and was functioning for 4 years and priceless data have been captured by this telescope. These data are in fits type which should be processed. The first need is a software called map maker which convert the fits raw data into images which can be studied by scientists. One of these map makers is called Unimap, which is developed by DIET dept. of the University of Rome 'La Sapienza', in cooperation with the ASDC center of the Italian Space Agency (ASI). The project has started in 2011 and 7 stable versions of the product is released successfully. Since this project is using huge data as an input, the test period of the new version is taking so much time and effort from developers. For this purpose, there is the need of a simulator for this product. Using this simulator we can accurately control the noise level of input data and check the new version behaviors based in different conditions. We should also mention that based on the behavior of the telescope, there are different types of noises presented in the input file. Using this simulator, not only we can tune the noise types available in the input, but also the amount of each of them. This will result of accurately studying the changes in all different possible situations. It worth to mention the fact that in this thesis, the goal is not modifying the Unimap software itself, but to develop a software which test all aspects of the new versions of Unimap accurately and help the development of new version.

To my family
who never stopped supporting me

Contents

Chapter 1

Introduction

1.1 Image processing

Computers and digital processors have been a game changer in many fields. The imaging industry is one of those which have been totally revolutionized after introducing the computers and digital sensors. Both in detecting and analyzing fields, it has achieved what was next to impossible in the analog era. Usually an image is a sense of a picture which is a planar representation of brightness or the amount of light reflected or transmitted by an object. In digital representation, an image is usually a function of spatial variables. For example $f(x, y)$, where x and y are the Cartesian location and f is the brightness of the image in this point. This is the most basic and fundamental representation of an image. Any digital image processing system can be divided in three main component, the input system, the image processing part and the output system:

(1) The input system: the input system or digitizer convert the input (in most cases light) into array of numbers. The input of this part is usually a continues wave and the output in form of $f(x, y)$, where f, x and y are integer numbers. This section can be a combination of different hardwares for detecting different range of light which can not be detected by one single sensor. This part is usually divided in two subsection of sampling and quantization which are in charge of indicating the location and the amount of detected light.

(2) The image processor: The image processor will have the digital input and make the data ready for the output. The outputs can be very different and so the image processors. The goal of the system is not always displaying a picture and can be simply detecting an object in the image or not, in this case the processor should lead into a single logical bit output, true or false. There are several types of processors and usually the limitations of the hardware is

limiting the software and the capabilities of processing and accuracy of this section.

(3) The output: As we mentioned above, the output is not always a picture or a display, it can be simply a single sign to show if an object is detected or not, or a text display which writes the result of processing, like a color detector which writes one of the words Red/Green/Blue in case of detection of each of these colors. But in most cases, the output should transform the digital processed data into a continuous tone and spatially continuous wave.

We will focus more on image processing operations in this chapter which are going to be used to explain the different part of the main software of this thesis.

1.2 Image processing operations

In the classic image processing definition, this section starts after saving the data, but nowadays part of the compression and enhancement is done using a specific processor on the sensor itself when the image/video and a temporary storage, then the main process starts after saving the data on the actual storage. This is mainly a solution to the disk space problem and the dimension of the new devices and sensors. But we are going to stick to the classic definitions and categories of the digital image processing. The most famous categorising this section is listed below:

(1) Geometrical operations: In these kind of operations the value of the pixel remains the same, but the pixel is mapped into a new location. This can be useful for some effects like magnify, minify or rotate the image,but pretty essential for some situations which have distortions due to recording geometry, scale changes, rotations,perspective, or due to curved or irregular object surfaces.

(2) Point operations on single image: The value of a pixel at location (x, y) depends only on the same pixel (x, y) in the input file. Changing the contrast or brightening the image are examples of this kind of process.

(3) Point operations on multiple images: The value of a pixel at location (x, y) depends on the same pixel (x, y) in several pictures. In this method we are using data from several pictures to enhance the image quality. As an example we can name averaging the pixel value for noise reduction. Another examples is HDR (high dynamic range) photography. In this method the system takes several images with different brightness levels and after detecting the bright and dark parts of the image, it takes the dark parts from brighter photos and bright parts from darker shadows. This will end to a popular effect in which there is no saturated pixels, rich details and high contrast.

(4) Neighborhood operations on one image: The value of the pixel with location (x, y) depends on the same pixel of another image at location (x, y) and the neighbor pixels of it. This method is used in convolution and spatial feature detections like edge, line and edge detection. Depends on the application and process, the neighborhood can be limited to one single pixel around the location or an area of n pixels.

(5) Neighborhood operations on multiple images: This is the same operation of neighborhood operations on one single image but the sources are more than one.in this case we are sampling from different images and not just one single pixel.

(6) Operation based on object shape: The value of the pixel is determined by the object class to which a pixel belongs; examples include classification, segmentation, data compression, character recognition.

(7) Global operations:The value of a specific pixel depends on the value values of all of the pixels of the image, these include image transformations, e.g., Fourier, Hartley, Hough, Haar, Radon transforms

We will discuss more image processing. We will focus more on those which are needed in satellite image processing.

Geometrical operations:

The geometrical operations are fundamental for our use case. We will have a short review of the most important ones and will check them more in depth in the next chapter while discussing the practical problem which we face. As we have mentioned before, there are different geometrical processes. In the general definition, a geometrical process is mapping a pixel with (x', y') coordinates to (x, y) in the output image. As you can see, we are just mapping and no pixel value change is done in this operation.The most common geometrical operations are:

(1) Transformation by $(x_0, y_0) : x = x' + x_0,\ y = y' + y_0$

(2) Spatial scaling: $x = ax',\ y = by'$

(3) Rotation by 180 degree: $x' = -x,\ y' = -y$

(4) Skew: $x' = x + ay,\ y' = y$

(5) Perspective distortion: $x' = x + axy,\ y' = y$

(6) Rotation through $a : x' = x\cos(a) - y\sin(a),\ y' = x\cos(a) + y\sin(a)$

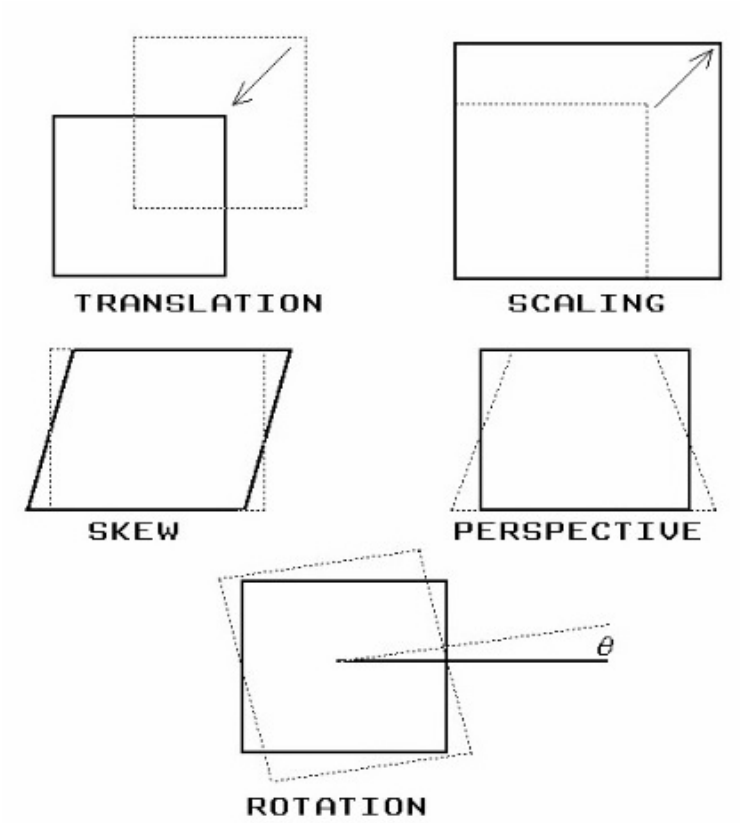

Figure 1.1: Examples of useful geometrical operations

Point operations on multiple images:

Another interesting processing type is the point operations on multiple images. In this process, the output (x, y) depends on the same pixel, but in several images. Images can be different in case of the same image but different time, might have different wavelength,depth resolution and even sensor sensitivity to the light. The output i usually calculated using an arithmetic operation of the input images, but in the binary images the logical operations are also possible. The most common application of this process type is:

- Image segmentation using multispectral information
- Averaging multiple images of the same picture for noise reduction
- Change detection by subtracting
- Windowing by a template mask image
- Correct for detector nonuniformity by division

Neighborhood operations on one image:

Neighborhood operations or better to say, local operations are those in which the value of the pixel depends on the same pixel and also pixels around the same one in the input image. To be more precise there are two definition of these operators. The first one is more mathematical and uses the integral operator, but since we are processing a two dimensional digital signal we can use a method called masking the input. In this method we introduce a mask which have different values for each cell. The values of each cell is called its weight and by changing the mask the operation is totally changed.

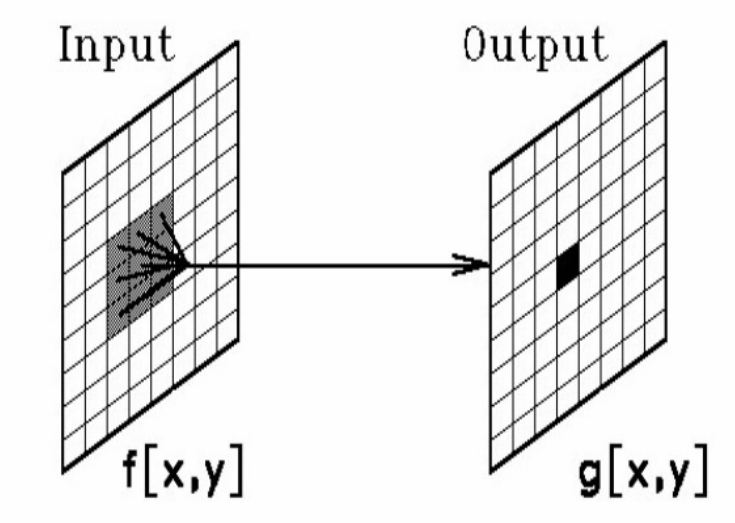

Figure 1.2: Neighborhood operations or better to say, local operations image processing

As an example, the mask for identity operation (output = input) is:

0	0	0
0	+1	0
0	0	0

Figure 1.3: Identity operation mask

Obviously, it is giving a weight of 1 to the same pixel of input image and the rest 0, so the output will be the same image

Vertical blur mask: in this mask the result will be the average of the pixel, and the upper and lower pixel in the input image, so the mask will be:

As you can see, we have divided the mask to 3, so the brightness of the image will remain the same.

Horizontal blur: The same mask here, just it's averaging over different direction
Low pass filter: In this filter, we are removing the sharp changes and smooth the image. No wonder why the mask is like a two dimensional blur

$$\frac{1}{3}\begin{bmatrix} +0.25 & +0.25 & +0.25 \\ +0.25 & +1 & +0.25 \\ +0.25 & +0.25 & +0.25 \end{bmatrix}$$

Figure 1.4: Horizontal blur mask

Sharpening mask:
The sharpening or high pass filter mask can be calculated easily. As we can consider the image is the sum of a uniform averaged image plus a high passed filtered image. In this case the high pass filter mask is:

$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 1.5: Sharpening mask

As we expected, the mean value of the mask is zero which reminds us the frequency zero of a signal.

As you have considered, we did not discuss this process in depth. We just mentioned the main process types and the categories and a step further for those we are going to use. Since we are going to look at them again in the next chapters and more practical use of them in space images and more specifically in Unimap, the software which is processing these images, we will close the image processing here.

Chapter 2

Astronomy image processing

In this chapter we are going to go more in deep with the astronomical imaging. We will focus on a conceptual framework for discussing the basic physics and dynamics of astronomical objects like stars, galaxies and then the quantity and quality of information captured by or instruments. Obviously there are limits about them which we will mention further. This concepts are pretty important as many imaging techniques and astronomical equipments are the results of understanding them. In previous chapter, since we wanted to talk about the concepts and a summary of processing we had a less theoretical approach, but in this chapter we are going to give a more complete description of astronomy imaging and in some cases going more in deep in to details and introducing the equations and some examples. This chapter is going to discuss more the visible light while we are going to start talking about infrared astronomy imaging from next chapter and then we are ready to discuss the Unimap, a maker of astronomical images.

2.1 The nature of light and noise

When we are talking about astronomical images the most important factor is that we are observing the distant object, but why is that important? The further the object is, the less light will be captured by our receivers. Since we are talking about millions of light-year the received information is countable photons. Photos are electromagnetic waves with a particle (quantum) nature. They are emitted and detected on an individual quantum basis. This has a strong effect on information carried by the light. Since we are talking about the quantum dependency of the light we know that the uncertainty of quantum information will affect the quality of information of an image of an object. This uncertainly is called Poisson noise.

Poisson noise can be explained using an example. Measuring it is like measuring the rainfall over a limited area in many buckets and during a specific period of time.

The amount of water in each bucket is slightly different from the one close to it due to the randomness of the falling drops. So for having a more accurate estimate of the rainfall we can average the amount of water in the buckets. Poisson noise is due to the quantum nature of the light. It can be modeled as a mathematical predictability and the uncertainty of an accumulation of random discrete event is described by: $Poissonnoise = squareroot(number_of_events)$

Talking about the poisson noise is actually about the probability. The noise itself denotes the range of input values in which there is 68% likelihood that the output will be true. In other words, The range which is probable enough also called as "margin of error". The inherent Poisson noise of light signals means that it is not possible to increase information by inde fi nitely amplifying or distilling a signal because the noise/uncertainty is intrinsic to the signal itself. This results in strict physical limits to what our receivers/cameras can do, so it is critical to understand this limits and know how far we can get since the light source is totally out of our control. As we have mentioned, this is totally because of quantum nature of the light and it does not exists for non-quantum signals. In this types of signals, the noise is only due to extra sources which interfere with the pure signal. A signal which is strong in energy/power can overpowers the noise easily and S/N becomes nearly linear. Obviously to double the S/N in non-quantum systems you just need to double the signal but in the quantum system, as we have seen before, for doubling the S/N we have to quadruple the signal. This image is a good example for demonstrating the limitations of quantum nature of light. This is an image of M57, the Ring Nebula.It has been captured using a photon counting image. Each dot is representing a photon. Every pixel represent i in a 30 seconds period of time a photon has arrived at the sensor or not. This image was taken by a photon counting ICCD which have zero read noise.

2.2 The Virtual Image

The virtual image consists of information conveyed to the focal plane by photons from astronomical objects, regardless of camera ("before" the camera). In other words, the Virtual image is the maximum amount of data we can have at the camera, so the information we can get on the camera is also limited by the characteristics of the virtual image. So it is also important to understand the dynamics and physics of the virtual image and the study our sensors and cameras. As mentioned before, light is consisted of photons and a telescope aperture is collecting them for a certain amount of time. Since we are far from the object in astronomy images, this time period is usually way longer than normal photography. The optics will convey the photons to the focal plane. This focal plane is where the virtual image will be formed and then will be captured using a sensor or camera. This virtual image contains

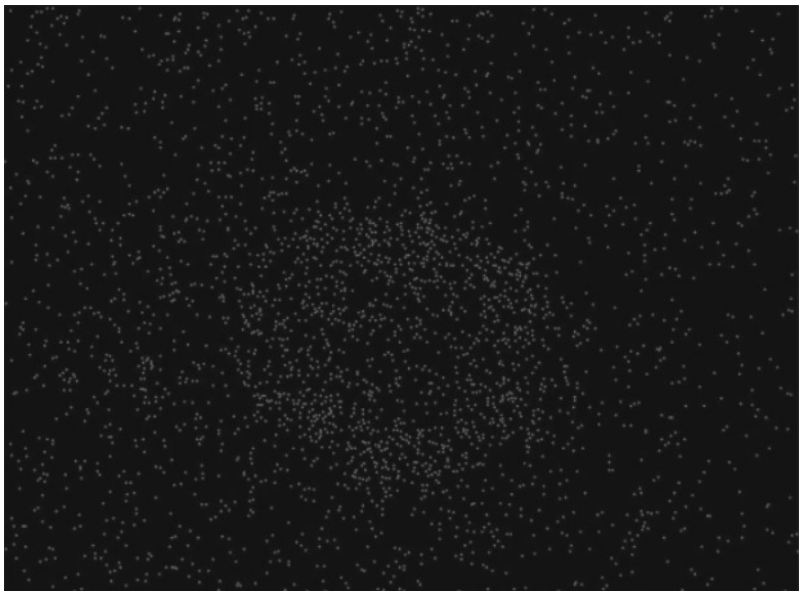

Figure 2.1: M57 (Ring Nebula) 100 ms exposure from a photon counting camera

information about each object. We can also say at the root, we have information of the location of every photon which has been collected by the telescope. Information about an object is based on the number of photons collected from the object, which depends on the area of the telescope's aperture and collection time. Focal length also affects the image. While the Focal length does not affect the number of collected photons, it determines linear scale, which is irrelevant for angular properties.

2.3 Signal to Noise (S/N) Ratio

As mentioned before, the information about an object is captured by number of photons available at the virtual image or simply called our "signal" here. Noise is the uncertainty of this number, so the proportion of signal to noise is called "Signal to Noise Ratio" which can be written as S/N or SNR. SNR is practically the most used measure of information quality. Here, the SNR determines the ability of higher contrast in the virtual image. Here, an object can be any detectable image in the sky. From stars and planet to an arbitrary path of sky. But usually we are talking about a planet, star or a galaxy which occupies an angular area in the sky. A star object is the simplest. All stars have the same angular extent for a given system/exposure. Stellar S/N is a straightforward measure with many applications; for example S/N = 3 is the lowest S/N for star astrometric detection. Star S/N determines astrometric and photometric accuracy. The simple object S/N of large objects is affected by object size, so for large objects it may be preferable to use an angular measure such as square arcseconds. An angular measure is similarly

used for galaxies magnitude per arcsecond squared. For a large object, higher S/N reveals The Theory of Astronomical Imaging finer contrasts, so a fuller description of S/N would include MTF terms, but that is beyond the scope of this chapter. It is worth mentioning that this approach is not common and the usual definition of SNR is totally restricted to a single pixel dynamics, independent of object and information SNR. The pixel based SNR is pretty easy to measure and implement. This method is very useful in normal photography and combatting the in camera noises or additive noises, but not so helpful in astronomy imaging.

2.4 Object S/N of Pure Signal

As we have mentioned before, there is poisson noise from the nature of the light so for the moment let's consider that we don't have any additive noise and we receive a pure signal. The SNR will be: S/N = (n photons)/sqrroot(nphoton) = sqrroot(N photons) As we can see the SNR is nonlinear and will be seen repeatedly in astronomy images.

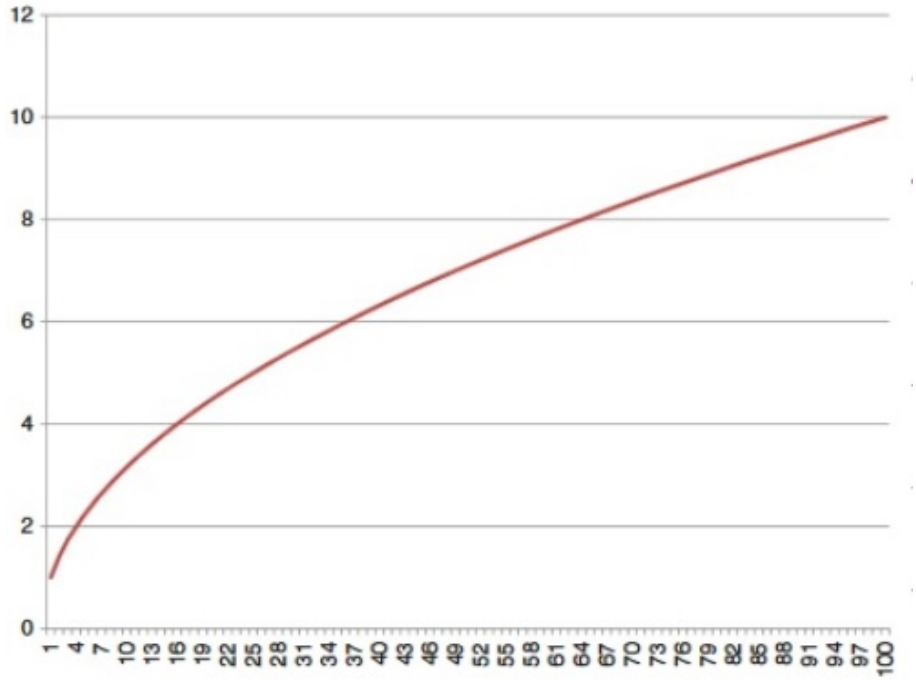

Figure 2.2: Signal to Noise as a function of signal in the absence of other noises

Now let's see how can we double the SNR. According to the equation to double the SNR we need 4 times more signal. As we are talking about astronomy images it is impossible to change the light source, but still we can modify the telescope and camera. One solution is to increase the exposure time 4 times more. This is

actually done since astronomy images are taken using long exposures but can't go any further. Consider an image with exposure time of 4 hours. It is impossible to double the SNR by taking a 12 hours image. The other solution is using a large aperture which has twice diameter of the one before. That's why the astronomy telescopes are built pretty large.

2.5 Object S/N with Sky

As an engineer we know that the noiseless channel never exists. It's also true for astronomy imaging. In the virtual image, moreover than the photons which are emitted from object in the sky, there will be other photons from atmospheric sky (high-altitude emissions and light pollution). Sky photons make the image blurry all over the image and affects the SNR: $SNR = S/(S + Sky)^0.5$ where S =number of photons from object Sky = number of photons from sky glow that occupy the same area of the object This equation clearly shows how the sky noise affects the virtual image. To the bright object, the sky noise is negligible but the case is different for dark objects. If we consider a sky 3 times brighter than the object we will need to extend the exposure time 4 times more to compensate the sky noise. In the following graph the SNR is shown based on the signal strength. As you can see in low signal levels the sky noise affects the SNR heavily.

2.6 Point-Spread Function (PSF) and FWHM

In astronomy imaging the object distance is so high that we are actually counting the arrival photons. The whole information is about the number of arrival photons and the arrival location. Every photon arrives at a different location (even if the distance is subatomic) which then results in image resolution. There is a distance at which or less, the location precision has almost no effect on information quality. The reason is constrained by physical limits of diffraction, atmospheric stability, optical quality, tracking etc. We can characterize this limit as a Point-Spread Function or aka PSF. A point-spread function (PSF) is the result of blurring a point-source. We consider stars as a point source since their angular diameter is really small. But the atmosphere and scope diffuse this point source into a ball shaped object which is fuzzy and blured. If we graph this source we will have a bell-shaped curve which is our PSF. Although the blurring procedures are complicated, but over time intervals the blur goes closely to the Gaussian function. The most important factor in the Gaussian function is the one which defines the width of the PSF. This parameter is a constant measure of resolving the power regardless of intensity. Function width is of the most important factors in the Nyquist sampling theorem. The commonly used

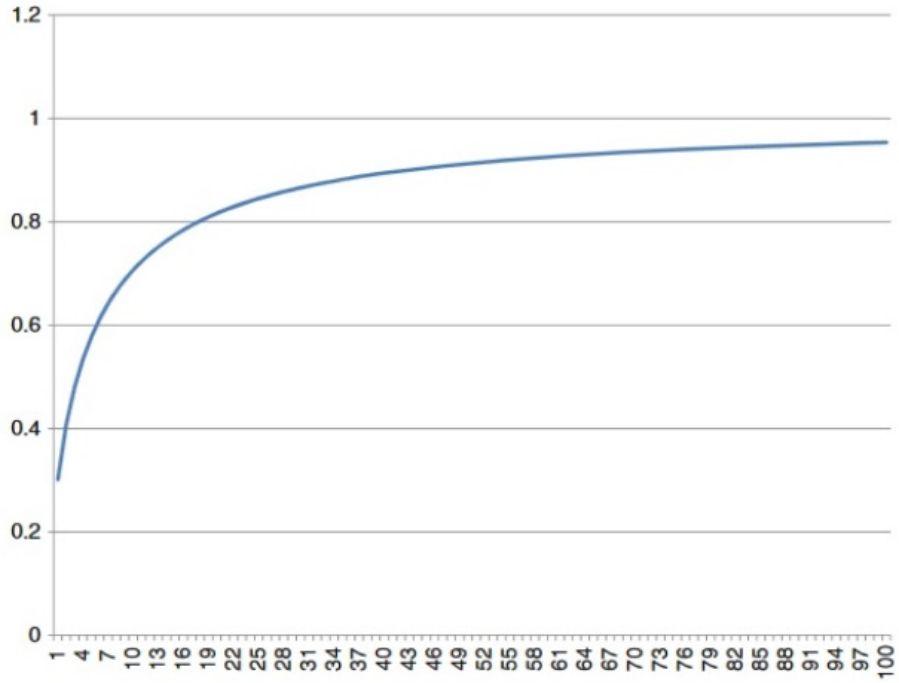

Figure 2.3: Differential effect of sky glow (10 photons) on S/N by signal intensities (1 = same S/N as if no sky glow)

FWHM (Full Width at Half Max) characterizes the PSF resolution by measuring the width (diameter) of the star at half of peak intensity. FWHM is a direct result of Gaussian function width by: $FWHM = 2.355 * width$

In the virtual image, Star images will have different sizes, while brighter stars will seem larger than than dimmer ones. This is the result of the PSF where the appearance of the star is due to a constant intensity level but the FWHM is still constant for both dim and bright stars. Both will have the same characteristics but with different signal powers.

2.7 Sampling

Since now, we have considered the process up to the virtual image, but we need to capture the virtual image and digitise it for any further representation or process. In this step we will divide the location into squares and for every square we are going to save 2 different data. The location of the square and the number of photons encompassed within. Each of these is squares is called a pixel.

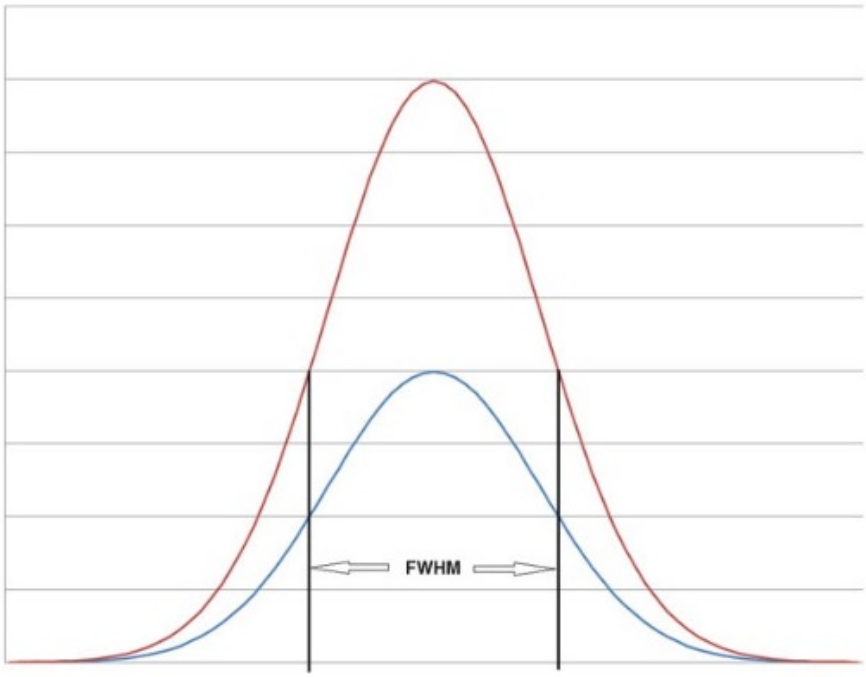

Figure 2.4: Full Width at Half Maximum (FWHM) is constant regardless of star brightness

Nyquist Sampling Theorem

The virtual image is the analog unsampled image which will be mapped to the digital image. One of the factors is the sampling rate and it affects the information quality. If we use a low sampling rate we will lose information. The Nyquist theorem makes it clear that if we use a sample size bigger than the function width of the signal, we will lose information. The Nyquist sample size for a Gaussian function is measured as $FWHM = 2.3555$ pixels. But because of the fact that pixels are squares and the diagonal of the square is longer, we have to update the sampling size to $FWHM = 3 - 3.4$ pixels. Astronomy images with $FWHM < 2.3$ pixels are called "under-sampled". There will be further problems in undersampled images like close double stars may merge into one and some details might be lost. On contrary, the images with $FWHM > 3.5$ are called "oversampled". In these images all the information is recorded. Over sampled images with $FWHM > 5$ might introduce practical problems in detectors and cameras.

Although under sampled and over sampled images have their own specific usage in astronomy imaging, but for most of the times we stick to a number close to Nyquist rate. The oversampling rate make practical problems in our camera and the under sampled images have the lack of information. Image Scale and Pixel Size

There are three ways to measure the distances in the virtual image:

- Angular distance: Angular distance is conceptually identical to an angle, it is usually measured in arcsec and useful in astronomy imaging.

- Linear distance: The distance between two points, in Astronomy images we talk about the distance in virtual image.

- Pixel: considering the pixel size is fixed, then it is easy to culate the distance in pixel number

As an example, The angular diameter of the Jupiter is: Angular: 40 ar-second, This is independent of the focal length of telescope and pixel size

Linear: 200 microns for focal length of 1m

Pixel: 40 pixels for focal length of 1 meter and pixel size of 5 microns

As you can see, linear size depends on focal length and distance in pixel depends on both focal length and pixel size while the angular distance is independent of both of them. That's why in astronomy images the distances are usually shown in angular.

2.8 Object S/N Versus Pixel S/N

Measuring the SNR of the virtual image, we can characterize it by "object SNR". Since the virtual image is not sampled yet this is unaffected by sampling rate. On the other hand, in the sampled image, the signal of the object is divided into pixels and if we want to calculate the object's signal we just need to sum all the pixels which have information about the object.

Instead of talking about the object SNR we can also talk about the pixel SNR. Every single might have some information about the object but not all of them.As we mentioned the object's signal is distributed over several pixels. Obviously it is easier to calculate the signal of a pixel and calculate the pixel noise and the SNR but we should consider that the concept of pixel SNR and the object SNR are totally different.

As an example, we can consider a virtual image of a square object 2*2 arcsec and total intensity of 100 photons. We assume the sky glow is zero and sampling the image with 0.5 and 1 inch pixel size.We will have: $ObjectSNR = sqrt(100) = 10$ 1 inch pixel size: object occupies 4 pixels so every pixel will have 25 photons, then pixel $SNR = 5$ 0.5 inch pixel size: object occupies 16 pixels so every pixel will have 6.25 photons, then pixel $SNR = 2.5$ As you can see the pixel SNR depends on pixel size and one way to battle it is to normalize the pixel size

2.9 Normalizing Pixel Size

To compare any pixel based characteristics of different sized pixels, we have to normalize the pixel size mathematically. This becomes even more important when we have to use different camera on a particular telescope. We can normalize the pixel sizes into each other or to a standard reference. This can be done using an angular distance reference, like 1 arcsec or a linear distance such as 1 mm. As we mentioned, the first one is independent of telescope and can be used to compare images from different telescope but the second one depends on telescope characteristics so can be used to compare different images of the same telescope.

Chapter 3

Infrared image processing

In this chapter we are going to have a short review on infrared light and imaging. Obviously we will have astronomy approach and check the possibilities and problems of working with infrared light. Infrared history starts from the year 1800 and William Herschel. He was working on sunlight and tried to pass the light through a prism which splits it into a rainbow of colors. He was experimenting the differences of the light spectrum from violet to red. He found out that the heating power of the light is different among different colors. The red light was able to arise the temperature much faster than blue or violet, but he wanted to experiment something even more interesting, he wanted to explore beyond what was visible with eye and find out if there is anything from sunlight which produces radiant heating effects but still can not be seen by eyes. For this purpose he used three different thermometers on his table while the middle one was in the light spectrum and the other two were out of the light spectrum. So he could use the two side thermometers for the room temperature and change the one in the middle to experiment the different light's heat power. He started by testing his previous experiment and confirming the results which say the heating power of the red light was more than the violet light, but this time he did not stop and went further and further.He moved the thermometer out of the spectrum which he could see and surprisingly, he was able to detect heat power even out of the spectrum. He kept recording the results out of the spectrum and he realized that the maximum heating effect occurs about 1 inch after the las visible light and the heating effect can be seen up to 1.5 inch after the last visible part of the light spectrum which was 8 inch. This was a historical moment because William Herschel has discovered the infrared light

3.1 The electromagnetic spectrum

Astronomy science relies on astronomy imaging which is detecting the signals which are carried by electromagnetic waves. These can be visible light, infrared light,gamma rays or radio waves. Among all of them the infrared plays a key role and is very important for astronomy, we will discuss why but before that we need to learn more about the nature of electromagnetic radiation and the sources which produce the infrared light. Herschel's discovery was the basic most important step of the infrared but as any other discoveries in the world, brought so many questions about light itself. The theories and discoveries continued up to 20th century when it finally led to the quantum mechanics. At that time scientist could not agree on the nature of the light. Some believed in it as a wave phenomenon or some others thought it is made of particles. As we know now it has both wave and particle properties but in that era, the wave theory was more popular so that they believe there was a wavelength assigned to it. Herschel used a prism to separate the light. The prism bent or refracted the sunlight rays with different angles depending on the wavelength. So that the blue light was bent more than the red. So Herschel discovered that the prism is dividing the sunlight into different sections with a range from blue to red which could be seen by eye, and a part after red which could not be seen but he could measure using his thermometer. We know that the electromagnetic spectrum covers much more wider range of wavelengths than visible light or even what Herschel has measured. From radio emission to the X-ray and gamma-ray radiations are part of the electromagnetic spectrum. Depending on the wavelength they have their specific characteristics and their interaction with matters. As an example the blue part of the visible light refract stronger than the red part. So the prism can divide these lights but for the ultraviolet and X-rays, the prism is opaque so it can divide these lights. The atmosphere is also opaque to very short length waves which provides a kind of protection for us,since short wavelengths are harmful for human. The wavelength of infrared light is between the edge of the visible light which is almost one micron and 1 millimeter. Since this range is wide, infrared itself is divided in three different sub-ranges:

- Near infrared: It has wavelength of 1-5 microns and is the closest part of the infrared to the visible light.
- Mid-infrared: The wavelength of 5-25 microns
- Far-infrared: The wavelength of 25-500 microns

Nowadays the infrared which was detected by Herschel is not of our interest in astronomy. It is now called a extended part of the visible spectrum which is called optical. That's because the same techniques can be applied to it as the visible

light. This is just about astronomy and for normal photography this wavelength (0.8 microns) is still regarded infrared light.

3.2 Thermal radiation

There are different ways which light is produced and every electromagnetic wave has some parts. One of the factor which is common in all of them is heat. All bodies produce electromagnetic radiation but the moment we think that we have felt the radiations, in fact we have perceived the radiant heat. As an example thinking about a tungsten light bulb, we can see the light produced by it but what we actually feel from it is the radiated heat. The light we receive from the Sun is also radiant heat, which is traveling the 150 million kilometers of space between Earth and Sun. These examples are just a few of radiant heat and are obvious to us because we can see and actually feel the heat by our skins. But not just these radiating electromagnetic waves. Even cold objects are radiating, from cold surfaces in earth, to the temperature of the universe which is - 270.27C. As we know there are several measuring units for temperature.The most famous ones are Celsius and Fahrenheit, but physicist prefer another one. Kelvin is the one which is similar to Celsius but instead of using the freezing temperature of water as the reference, it uses the absolute zero. The -273C temperature is the lowest one which any object can have. The heat itself is a state of movements and interactions of the atoms of an object with each other and at the 0 Kelvin, it is believed that there is no interactions between atoms. That's why it is called absolute zero and the lowest possible temperature for any object. Practically this temperature is also impossible to reach and is a good reference since all the other temperatures are above it. As we mentioned, every object radiate electromagnetic radiations and the characteristics of it depends on the matter and energy level of the object. In a medium the object interact with radiations until they evolve to a state which the emitted and received radiations are equal but in their own frequencies. This radiation is called Thermal or black body radiation and obviously is higher for the hotter body. The actual spectrum of black body radiation, the power emitted at each wavelength, was a puzzle for physicists at the end of the 19th century. The light coming out of a fixed cavity was considered to arise from all possible wavelengths of light that could fit inside it. These can be thought of as a set of standing waves whose wavelengths are a fixed multiple of the size of the cavity, so they always have a value of zero at the walls. A standard result of the study of heat and energy, making up the theory of thermodynamics, states that each of these standing waves, or modes, should have the same energy. As you can tell from Figure, there are many many more high frequency modes, waves that go up and down many times from one side of the box to the other, than low frequency modes. These correspond to high frequency

electromagnetic radiation, so they would represent ultraviolet light or beyond. This means that, under the assumptions of 19th century, classical physics, there should be a huge amount of ultraviolet light in all Black body emission at different wavelengths for different temperatures thermal radiation. But we can easily see when we look at the spectrum of thermal radiation that this ultraviolet catastrophe, as it was known, is not the case. The solution to this problem was devised by Max Planck in 1901, and provided the first hints of the quantum physics revolution that was to follow. Planck suggested that the amount of energy that could be carried by electromagnetic radiation was not continuous. Instead, this energy came in discrete lumps, or quanta, like tiny bullets.

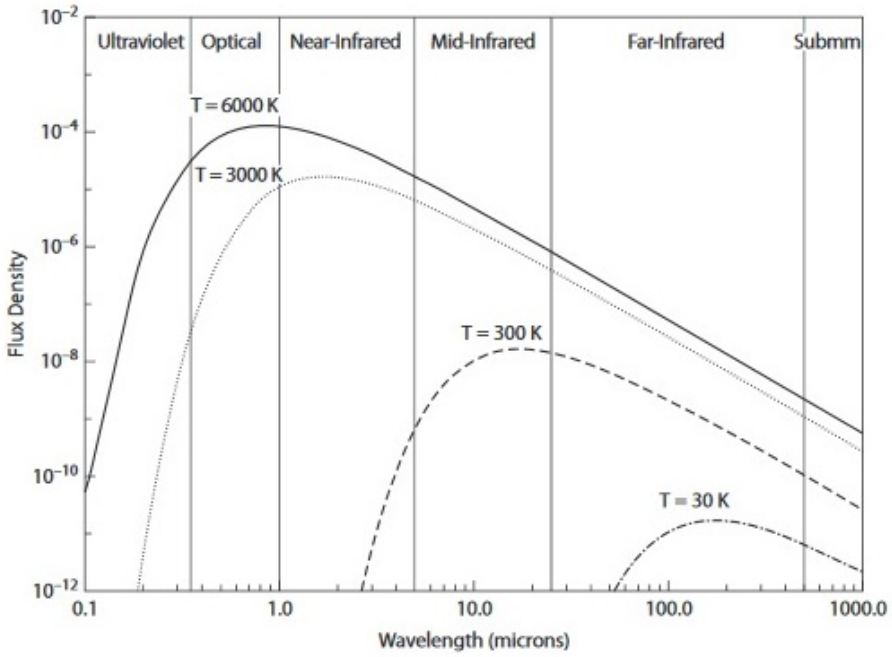

Figure 3.1: The spectrum of black body radiation at different temperatures.

The black body spectrum for a star like our Sun, at a temperature of - 6000K, peaks in the optical. Cooler objects have black body spectra that peak at longer wavelengths. Something at room temperature, 300 K, emits most strongly in the mid infrared, around 10 microns in wavelength, while interstellar dust, at around 30 K, emits most strongly in the far infrared, around 100 microns.

Overall, the black body spectrum has fundamental properties. The most important one is that it's peak wavelength is not fix for every material and depends on the temperature of the emitting object. The hotter is the object, the shorter the wavelength becomes and the colder objects emit longer wavelengths. As an example, the Sun is about 6000K, with a peak of black body spectrum to the color yellow. IT is interesting to know that this wavelength is the peak of human eye sensitivity

too. As another example the human body skin also emit heat by electromagnetic wavelength. The wavelength is about 10 microns at the 300K (human body temperature). This wavelength is in mid-infrared thats why infrared cameras can be helpful in finding human in emergency situations. Because other object will emit wavelengths different from this one and using a filter we can just detect the live humans. The second point is still interesting. As the emitting wavelength of the object in specific temperature is fixed and so the radiated energy, we can use the amount of received energy to estimate the emitting material. There are some exceptions here, mostly about opaque object.

3.3 Spectral lines and quantum mechanics

Planck started a great way in solving the light and electromagnetic waves problems by his solution of quantising the amount of energy in electromagnetic radiations. Although this was just the first step, but also a fundamental and essential step to solve lots of classical physics problem. One of these which is important in astronomy, is the origin of spectral lines.

During the 19th century, scientists discovered that, in certain circumstances, a given material would emit light at specific wavelengths. This could come from a tube of gas through which an electric current was passed, or from a chemical compound being heated in the flame of a bunsen burner. The spectral lines seen in emission when a compound was heated could also be seen in absorption when light passed through a similar material. A tube of gas that gives out an emission line at a specific wavelength when excited by an electric current would, for example, absorb light passing through the tube at that same wavelength.

These spectral lines provided reliable fingerprints for identifying the presence of a given element, allowing chemical flame tests to determine what an unknown compound might contain. Copper, for example, produces a lovely green colored flame in a bunsen burner. These techniques were applied to observations of stars and nebulae by William and Margaret Huggins.They found that astronomical objects contained the same chemical elements as we find on Earth.

The origin of spectral lines remained a mystery until the quantum physics revolution of the early 20th century. Planck's discovery that light is quantised, that it comes only in discrete particles whose energy depends on their wavelength, was only the start. Physicists soon realised that everything in the world of atoms and molecules is quantised. An atom or molecule can be in only a limited number of states. If an atom is hit by a photon of light whose wavelength matches the energy of one of those states, then the photon can be absorbed. Photons of other energies cannot be absorbed so easily. This process gives rise to spectral absorption lines at specific wavelengths, while the opposite process, whereby an atom decays from a

high energy, excited state to a lower energy state, produces an emission line.

The exact wavelengths of the lines depend on the detailed properties of the atoms, since the process of absorption and emission are shifting the electrons within the atoms from one allowed quantum state, also known as an energy level, to another. The atom doesn't pass through any intermediate states in such a transition - it just goes from one state to another in what has been called a quantum shift. It isn't just atoms that have quantised transitions giving rise to spectral absorption and emission lines. The same physics applies to oscillations in molecules, where some of the atoms in the molecule bounce around like weights on a spring. Many of these molecular transitions are at lower energies, in the infrared rather than the optical, since the absorption of higher energy photons may break the weaker bonds that keep molecules in one piece.

Observations of spectral lines, both emission and absorption, can be used in astronomy for more than just working out what elements and molecules make up an astronomical object. Emission lines from atoms in higher energy, excited states can reveal something about the nature of the rad iation that is exciting them. Ionised material, atoms and molecules that have had electrons stripped away from them, have different spectra than the un-ionised versions. High levels of ionisation, where an atom has had many of its electrons stripped away, take a lot of energy, possibly implying the presence of hard, X-ray radiation or some other highly energetic processes which might not be directly detectable by an observer. The study of spectral lines, across the electromagnetic spectrum, is thus a very powerful tool for the observational astronomer.

3.4 Telescopes

Now let's get into more details of the telescopes and how they work. Astronomy is all about observation and that shows the fundamental tools of this science are telescopes which can observe the universe clearly.

Basically we have two main types of telescopes: First we have telescopes using lenses to collect the light and then capture it, these are called refracting telescopes We have also telescopes which uses mirrors to collect the light. These are called reflecting telescopes The refracting telescopes were the first made telescope which were used by first generation of astronomers like Galileo. But nowadays almost all professional and even some amateur telescopes are reflective. There are several reasons for building the reflector telescopes instead of refracting, but probably the most important is the practical problems of making a huge refractor telescope is much more than the reflectors. Don't get this wrong, the reflectors have been very useful during the last eras and actually the fundamentals of the astronomy have been discovered by them, but astronomers need bigger telescopes which can collect more

light and is more accurate. The refractors were very useful and still are, but the new reflectors are even better. Also the glass of the refractor telescopes is opaque to a large section of infrared wavelengths. A typical reflecting telescope is shown in the figure below.

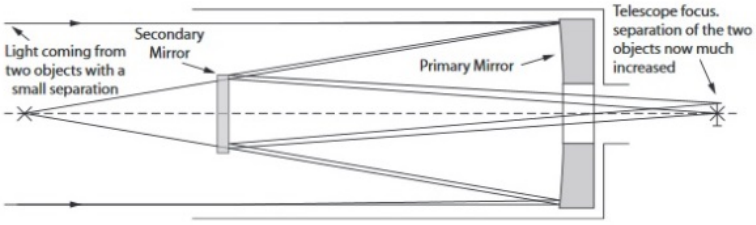

Figure 3.2: Diagram of a typical reflecting telescope

It is clear in the image that how this telescope is able to differentiate two object even if the arrival photons are close to each other.In this kind of telescope, the larger mirrors results in better separation in close objects. Another consideration here is the effect of the wavelength to the separation. Long wavelengths are separated less than the shorter ones, so for separating the long wavelength we need even bigger mirrors. On the other hand we have some practical limitations for big mirrors. As an example for Space based telescopes we need the mirrors and all the rest instruments to be launched to the orbit and that makes us limited to a maximum weight and size. Also on the ground telescopes we have same limitation. For observing the sky the telescope needs to turn around and this movements makes it even more difficult to manage the heavy mirrors. For battling this problem some active support systems are used and the mirrors are made out of several hexagonal segments instead of one big mirror. Another big problem for these kind of telescopes is the need of a clean view of the sky. The telescope shouldn't have any kind of obstacles in it's sight. Any air pollution or clouds is problematic for the telescope. This is actually one of the main reasons which make astronomers interested in space telescopes but for ground based telescopes they usually use an isolated place far from cities. As an example the Thirty Meter Telescope (TMT) is on one of the Hawaii islands is a perfect ground based telescope in case of environmental situation.

3.5 Most known telescopes

After have a short review of the engineering and physical specifications of the telescopes, now it's time to have a look at different telescopes which astronomers have been using several years for studying the sky. Many of telescopes which we are going to mention are not operational anymore but not only they have influenced the astronomy science, also the data captured by them years ago are still in use

by data processors For two main reasons. First by using new processing methods we can extract more information from the same data, and second is the fact those data are unique. The images captured of a star far from here is valuable because those starts might not exist anymore and even using more advanced observatories, we can't have any other images from them anymore. Now we will have a look at the most advanced and well known observatories.

3.5.1 Ground-based telescopes

UKIRT Telescope

Figure 3.3: The UK Infrared Telescope (credit: UKIRT/JAC)

This is not one of the first large telescopes which has ever built, but one simple fact made it very special. It was built and dedicated for infrared observatory. This telescope has a large 3.9 meter diameter primary mirror which make it the second largest infrared dedicated observatory in the world. UKIRT stands for United Kingdom Infra-Red Telescope. Unlike its name it is located on Mauna Kea, Hawai'i as part of Mauna Kea Observatory. It is owned by the Science and Technology Facility Council of United Kingdom and is currently being funded by NASA.

For more information please visit the official website of UKIRT observatory at http://www.ukirt.hawaii.edu/

The Keck telescopes

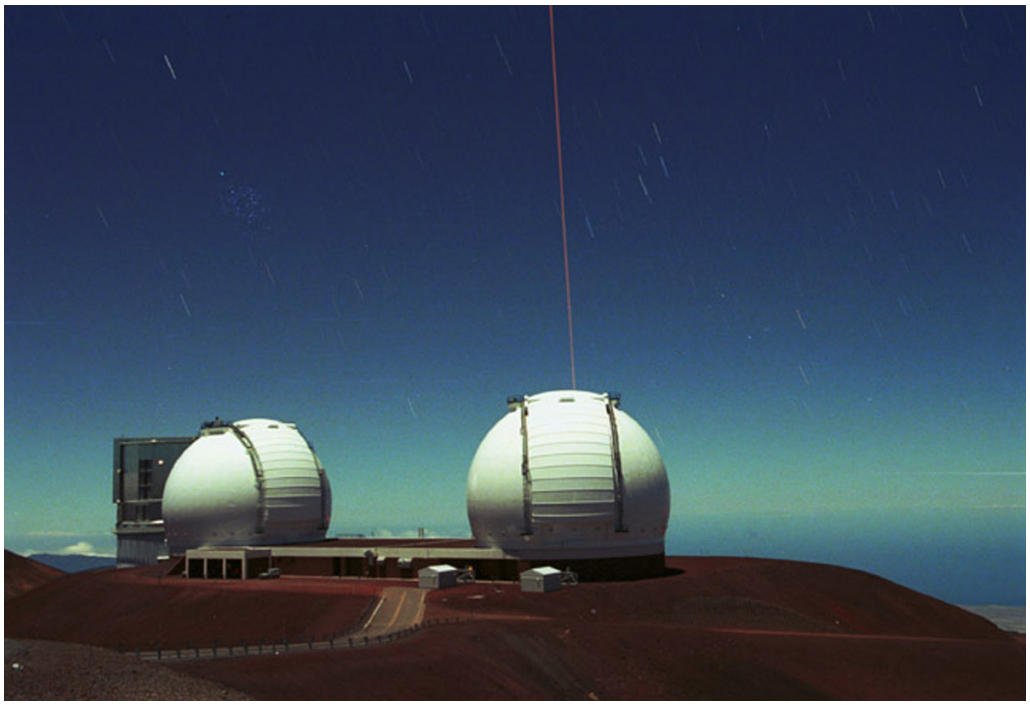

Figure 3.4: The Keck telescopes, credits:www.keckobservatory.org

These are also based in Mauna Kea, and also among the largest telescopes in the world. The main difference is the fact that these are optical/near-infrared telescopes. These Telescopes have primary mirrors which are 10 meters in diameters and such huge mirrors are possible thanks to the fact they are made of several segments. Any of them are made of 36 hexagonal mirror segments. This telescope has actually started working in 1993 and was the first telescope which has used segmented mirrors. This technology was actually a revolution in building large telescopes. As mentioned, these telescopes are not dedicated to infrared images but can observe near field infrared. Ech of them weigh about 300 tons and can smoothly track objects for hours. There is a new technology used in the Keck telescopes called Adaptive Optics which removes the effects of atmospheric blurring. AO measures and corrects this effect using a deformable mirror that changes shapes up to 2000 times per second. This method is using a special-purpose laser to make a bright artificial star by exciting the atoms in sodium layers. For the first time this method was used in 2004 for producing more detailed and sharper images. For more information please visit the official website of Kek observatory at http://www.keckobservatory.org

The European Southern Observatory (ESO) Very Large Telescope

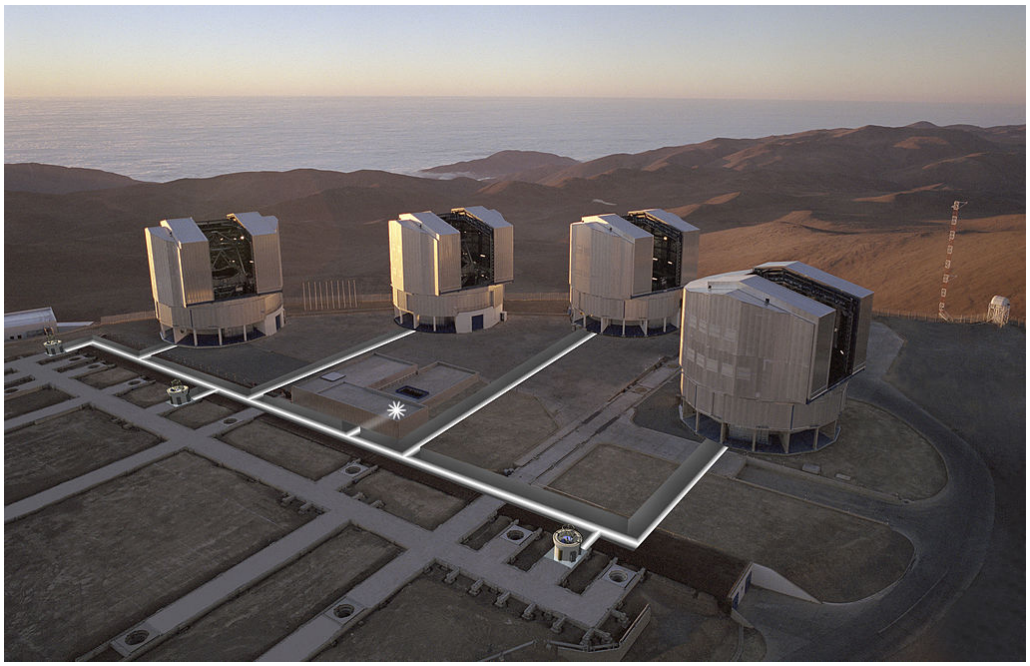

Figure 3.5: The European Southern Observatory (ESO) Very Large Telescope By ESO (ESO) [CC BY 4.0 (http://creativecommons.org/licenses/by/4.0)], via Wikimedia Commons

This complex is located on a top Cerro Pachon in atacama desert in Chile. This observatory complex is actually made of 4 completely separate telescopes, each of them has a 8.2 diameter mirror which are actively supported. These telescopes are not the largest in the world and usually are working separately, but it is possible to combine the light from all of them to form a much larger telescope. The name is actually because of this combination. Each year almost 2000 proposals are made for this telescope. These are requesting 6 times more than available nights in the year!That has made the ESO the most productive observatory of the world and a great number of over 840 refereed papers are published based on ESO data, just in 2013.The huge archive have almost 1.5 million images which are almost 65 terabytes of data. The VLT is also functioning on both visible light and mid and near field infrared. This combination of the visible light and infrared observatory has been really useful for the astronomers. In the old days, they actually had to go tu UKIRT to have infrared data but nowadays almost all the visible spectrum telescopes also works with at least nearfield infrared. For more information please visit the official website of Eso observatory at http://www.eso.org

3.5.2 Space based telescopes

As we have mentioned in previous chapters, the infrared photography has so many limitations on the earth. For some wavelengths it is actually impossible and

for the other the problems of earth atmosphere still remains. For these reasons the telescopes have been sent to space. Obviously this is an expensive and difficult mission but the results were so satisfying that a new mission has been started before the previous one ends. Here we will have a short look of few space based telescopes.

IRAS

Figure 3.6: An artist's impression of the IRAS satellite while operating in Earth orb it.(Courtesy of NASA.)

IRAS is practically the first space based telescope ever. The Infrared Astronomy Satellite has done more than a mission for infrared astronomy. It can be called the foundation of the far infrared astronomy since its images were the first far infrared images which have arrived to the astronomers. It will be even more interesting to know that all these valuable information were achieved by a mirror of 57 cm and operation time of 10 months. It was launched in 1983, but changed the infrared astronomy forever.

Hubble Space Telescope

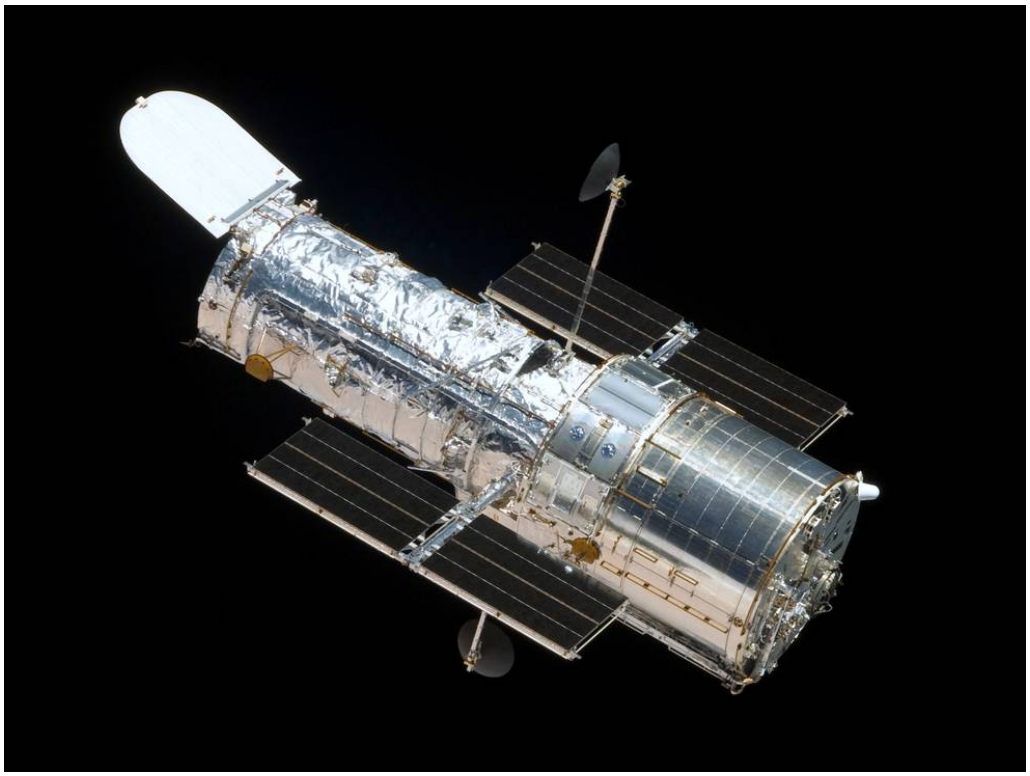

Figure 3.7: This photograph of NASA's Hubble Space Telescope was taken on the fourth servicing mission to the observatory in 2009. Credits: NASA@*https* : *//www.nasa.gov/mission*
pages/hubble/story/index.html

After IRAS, Hubble was the next infrared telescope launched into the space. Hubble was primary an optical observatory, it has been equipped with different near infrared instruments during its lifetime. The Hubble mirror was 2.5 meter which was a great upgrade over IRAS. The Hubble has been aunched in 1990 and still operating. The successor of the Hubble is the James Webb Space Telescope (JWST) and is scheduled for launch in 2019. For more information please visit the official website of Hubble observatory at www.hubblesite.org

Herschel Space Observatory

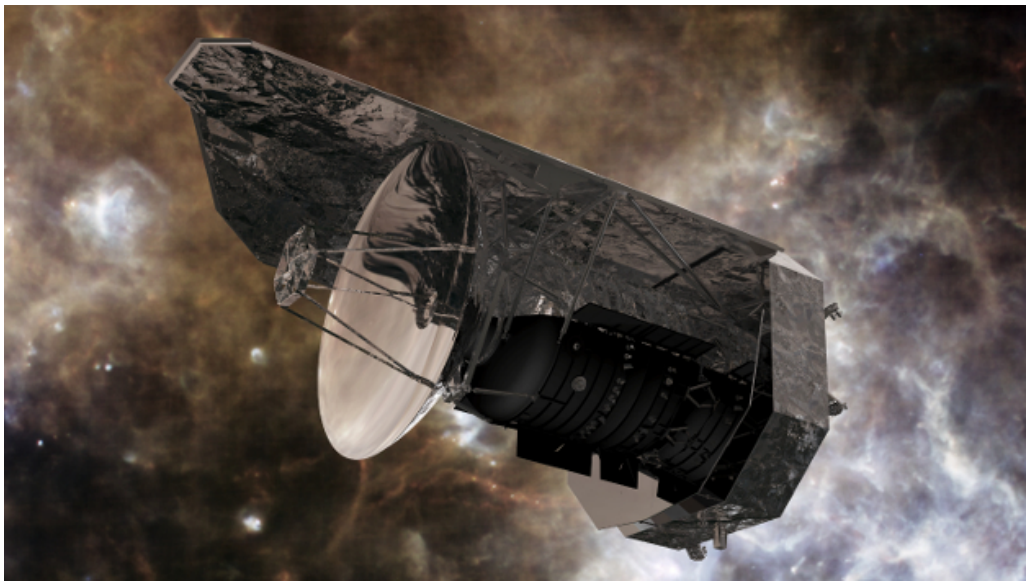

Figure 3.8: Satellite: Herschel Space Observatory Copyright: ESA/Herschel/NASA/JPL-Caltech; acknowledgement T. Pyle and R. Hurt (JPL-Caltech)

One of the most ambitious missions to date, is the Herschel Space Observatory. Wit a 2.5 meter mirror, Herschel has the largest mirror which has ever launched into the space for infrared astronomy. It has been launched in May 2009 and was active for 4 years. Its instruments were capable of seeing the coldest and dustiest objects in space. We will discuss more this satellite in next chapters since the main input data and aso the results of simulation are data captured by Herschel sattelite For more information please visit: https://www.herschel.caltech.edu/

Chapter 4

Unimap software and Unimap Simulator

As we mentioned, The Unimap software is using the Herschel satellite data as input. So we are going to have a short look at the Herschel satellite specifications and instruments and then have a deep review of Unimap.

In this chapter we are going to review to main article about Unimap and check the specifications in details. Then we will start talking about the need and development of the simulator for this software. Herschel satellite is a space telescope launched by the European Space Agency (ESA) in 2009. It hosts two infrared instruments, the Photodetector Array Camera and Spectrometer (PACS) and the Spectral and Photometric Imaging Receiver (SPIRE).

The instruments operate with an unprecedented resolution, sensitivity and dynamic range, making Herschel a key tool for astrophysical research, the data of which will be exploited in next years. Several map-makers exist for Herschel data, including Madmap, Scanamorphos, Sanepic, Tamasis and Romagal. Among these, Unimap stands out as one of the most cost-effective. Indeed it can produce quality images with a comparatively modest hardware and is efficient memory-wise, which is a noticeable advantage, given the huge volume of data produced by Herschel. This is achieved thanks to several original algorithms and implementation choices that will be described in next sections. Unimap has been adopted by several research groups and has been used to process the data of some Herschel key programs, like Hi-GAL and Hermes. Moreover, Unimap was evaluated and compared with other mapmakers at a workshop organized by the National Aeronautics and Space Administration (NASA) and the ESA, showing that it is a robust and performing software.Unimap implements a full pipeline, starting from the level 1 data of the standard pipeline and delivering high quality final maps. Unimap is the successor of and was obtained from the Hi-Gal/RomaGal pipeline. With respect to the Hi-Gal pipeline Unimap offers some improvements: it is automatic and it is a standalone, portable package.

Furthermore several novel processing approaches are introduced.

4.1 Herschel Satellite

The Herschel Space Observatory was a space observatory built and operated by the European Space Agency (ESA). It was active from 2009 to 2013, and was the largest infrared telescope ever launched carrying a single 3.5-metre mirror and instruments sensitive to the far infrared and submillimeter wavebands (55–672 µm). Herschel was the fourth cornerstone mission in the ESA science program, along with Rosetta, Planck, and Gaia. NASA is a partner in the Herschel mission, with US participants contributing to the mission; providing mission-enabling instrument technology and sponsoring the NASA Herschel Science Center (NHSC) at the Infrared Processing and Analysis Center and the Herschel Data Search at the Infrared Science Archive. The observatory was carried into orbit in May 2009, reaching the second Lagrangian point (L2) of the Earth–Sun system, 1,500,000 kilometers (930,000 mi) from Earth, about two months later. Herschel is named after Sir William Herschel, the discoverer of the infrared spectrum and planet Uranus, and his sister and collaborator Caroline Herschel. The observatory was capable of seeing the coldest and dustiest objects in space; for example, cool cocoons where stars form and dusty galaxies just starting to bulk up with new stars. The observatory sifted through star-forming clouds—the "slow cookers" of star ingredients—to trace the path by which potentially life-forming molecules, such as water, form. The telescope's lifespan was governed by the amount of coolant available for its instruments; when that coolant ran out, the instruments would stop functioning correctly. At the time of its launch, operations were estimated to last 3.5 years (to around the end of 2012). It continued to operate until 29 April 2013 15:20 UTC, when Herschel ran out of coolant. The mission involved the first space observatory to cover the full far infrared and submillimeter waveband. At 3.5 meters wide, its telescope incorporated the largest mirror ever deployed in space. It was made not from glass but from sintered silicon carbide. The mirror's blank was manufactured by Boostec in Tarbes, France; ground and polished by Opteon Ltd. in Tuorla Observatory, Finland; and coated by vacuum deposition at the Calar Alto Observatory in Spain. The light reflected by the mirror was focused onto three instruments, whose detectors were kept at temperatures below 2 K 271 ℃. The instruments were cooled with over 2,300 liters liquid helium, boiling away in a near vacuum at a temperature of approximately 1.4 K 272 ℃. The 2,300-liter supply of helium on board the spacecraft was a fundamental limit to the operational lifetime of the space observatory; it was originally expected to be operational for at least three years.

Herschel carried three detectors: PACS (Photo detecting Array Camera and Spectrometer) An imaging camera and low-resolution spectrometer covering wavelengths

from 55 to 210 micrometers. The spectrometer had a spectral resolution between $R = 1000$ and R=5000 and was able to detect signals as weak as -63 dB. It operated as an integral field spectrograph, combining spatial and spectral resolution. The imaging camera was able to image simultaneously in two bands (either 60-85/85-130 micrometers and 130-210 micrometers) with a detection limit of a few millijanskys. SPIRE (Spectral and Photometric Imaging Receiver) An imaging camera and low-resolution spectrometer covering 194 to 672 micrometer wavelength. The spectrometer had a resolution between$R = 40$ and $R = 1000$ at a wavelength of 250 micrometers and was able to image point sources with brightness around 100 millijanskys (mJy) and extended sources with brightness of around 500 mJy. The imaging camera had three bands, centered at 250, 350 and 500 micrometers, each with 139, 88 and 43 pixels respectively. It was able to detect point sources with brightness above 2 mJy and between 4 and 9 mJy for extended sources. A prototype of the SPIRE imaging camera flew on the BLAST high-altitude balloon. NASA's Jet Propulsion Laboratory in Pasadena, Calif., developed and built the "spider web" bolometers for this instrument, which is 40 times more sensitive than previous versions. The Herschel-SPIRE instrument was built by an international consortium comprising more than 18 institutes from eight countries, of which Cardiff University was the lead institute.

HIFI (Heterodyne Instrument for the Far Infrared) A heterodyne detector able to electronically separate radiation of different wavelengths, giving a spectral resolution as high as $R = 107$.[28] The spectrometer was operated within two wavelength bands, from 157 to 212 micrometers and from 240 to 625 micrometers. SRON Netherlands Institute for Space Research led the entire process of designing, constructing and testing HIFI. The HIFI Instrument Control Center, also under the leadership of SRON, was responsible for obtaining and analyzing the data. NASA developed and built the mixers, local oscillator chains and power amplifiers for this instrument. The NASA Herschel Science Center, part of the Infrared Processing and Analysis Center at the California Institute of Technology, also in Pasadena, has contributed science planning and data analysis software. Although this method can be applied to other image types too, but this thesis is optimized for PACs instruments and all the examples and graphs are produced using PACs images. Within the complement of three instruments selected to form the science payload, the shortest wavelength band, 60-210µm, will be covered by the Photo detector Array Camera and Spectrometer (PACS), which will provide both photometric and spectroscopic observing modes suited to address the key scientific topics of the Herschel mission.

4.2 Unimap Software

4.2.1 Image formation basic

We review some basic image formation concepts and techniques. We start by considering the image synthesis in the presence of noise. A. Image synthesis and noise Consider an image of M pixels, represented by an $M \times 1$ vector m. The image is observed with an instrument producing $D >> M$ readouts, represented by a $D \times 1$ data vector d. Assuming a linear, noisy instrument, the data vector can be written as

$$d = Pm + n = s + n \tag{4.1}$$

where P is a $D \times M$, full-rank matrix, which depends on the instrument and on the observation protocol, n is a zero mean, random noise vector and we introduced the signal vector

$$s = Pm \tag{4.2}$$

representing the ideal, noiseless output. The image formation problem is that of estimating m knowing d, P and the statistics of n. This is a classical problem having several established solutions. A simple and effective one is based on Least Squares (LS) and is summarised below. Since for a generic image x the ideal output is Px, we can minimise $|d - Px|^2$ for varying x and obtain the image producing the output closest to the actual data vector. This image is the LS estimate of m and is given by

$$\bar{m} = (P^T P)^{-1} P^T d. \tag{4.3}$$

The vector

$$\bar{s} = P\bar{m} = P(P^T P)^{-1} P^T d \tag{4.4}$$

is the corresponding LS fit to the signal. The vector $\bar{n} = d - \bar{s}$ gives the fit error and is an estimate of the noise. The Mean Square Error (MSE) is $MSE = |\bar{n}|^2/D$. LS estimation is an effective technique when the noise is white. However, infrared data are typically affected by correlated noise too, a disturbance found in several other imaging systems, e.g. DOI [14] and fMRI [15]. In this case, standard LS performs poorly and it is better to use a Generalised LS (GLS) approach2 [16], where the error is weighted in order to account for the noise correlation. Indeed, when the noise is Gaussian, this approach yields the Maximum Likelihood (ML) image. The GLS estimate has a simple expression, namely

$$m_g = (P^T N^{-1} P)^{-1} P^T N^{-1} d \tag{4.5}$$

where N is the noise covariance matrix, given by$N = Enn^T$. However, the computation of m_g is far from trivial as soon as the data size is appreciable. In fact, the direct use of (5) requires the inversion of a $D \times D$ matrix, which is impossible when D is

large. In section V-A we see how to circumvent this problem. B. Processing pipeline and disturbances Infrared data are affected by a number of other disturbances, in addition to noise, which need to be compensated by the image formation system. The optimal approach is to jointly compensate the disturbances and estimate the image. However, this is difficult to realize due to complexity reasons. Therefore, the image synthesis is normally preceded by one or more preprocessing steps and the software is organized as a pipeline of modules. Each module removes some disturbances and passes an updated data set to the following one, down to the last module which produces the final image. Several disturbances affecting infrared data can be modelled as deterministic, smooth, additive curves, depending on a few parameters. Similar disturbances are found in a number of other systems, e.g. [14], [15], [17], [18]. Then, the data model becomes $d = s + n + y$ where y is a vector depending on $K << M$ unknown parameters. Such disturbances can be removed by estimating the parameters from the data vector and producing a corresponding estimate of y. Then, the estimate is subtracted from d to obtain an updated data vector which is used instead of d in (5) to synthesize the final image. The parameters estimation is clearly a critical step and is not trivial, because the data vector also contains the signal s, which can bias the results. If this happens, the signal leaks into the estimate and is removed when the estimate is subtracted from the data, resulting in information loss. In section IV-D we present an approach to avoid this problem. Infrared data are also affected by impulsive disturbances, corrupting a small fraction of the data vector. In principle, also these disturbances can be estimated and subtracted from the data vector. However, they are not easy to model. Moreover, only a fraction of the data set can be used in the estimation, making the results less reliable. A better approach is that of detecting the readouts affected and excluding these readouts from the synthesis. This can be done with a negligible impact on the performance as long as the percentage of readouts discarded is low. However, this makes the computation of the GLS estimate more involved, as better discussed in section V-C. C. Filtering and other basic operations We introduce some basic operations that we need later. Given n numbers organized into a vector $x = (x_1, \ldots, x_n)$, their (arithmetic) mean is denoted by $mea(x) = (\sum_{k=1}^{n} x_k)/n$ and the median by $med(x_1, \ldots, x_n)$ or by med(x). The median is useful because if the numbers are samples of a random variable, it is a robust estimator of the ensemble mean while the arithmetic mean is more affected by the outliers. With a terminology abuse we also introduce the variance, given by $var(x) = (\sum_{k=1}^{n} x_k^2)/n - [mea(x)]^2$. A well known operation that we will use is the linear filtering of a sequence x[k] with an impulse response $h[k]$, producing an output sequence $y[k] = x[k] * h[k] = \sum_{n=-\infty}^{\infty} x[n]h[k-n]$, where $*$ denotes convolution. Moreover, given a sequence $x[k]$, we can compute the median over a sliding window of $2w+1$ samples and subtract it from $x[k]$ to produce $y[k] = x[k] - med(x[k-w], \ldots, x[k+w])$. This operation is a form of high-pass filtering [18] and is called the median high-pass filtering. Naturally, the filters can

be applied to finite length sequences too, by padding with zeros or mirroring. As we will see, the data vector d is normally obtained by stacking several different sequences of data. While this is handy, since it allows to develop compact expressions like (5), other operations have to be carried out separately on the sequences or on parts of them. In this case we will simply say that a sequence is îextractedî from the vector d and will not introduce a formal description of this operation, in order to keep the notation simple. In particular, we can extract the individual sequences from d, filter each of them and stack the output into a D1 vector h. When this is done using a median high-pass filter, we will denote the operation by $h = high(d)$.

4.2.2 Herschel data

The photometers of PACS and SPIRE consist of two and three arrays of bolometers, respectively. The first PACS array operates in the 70 (blue) µm or 100 (green) µm band and is divided into eight subarrays. The second operates in the 160 (red) μm band and is divided into two subarrays. The SPIRE arrays operate in the 250, 350 and 500 μm bands. The field of view of each array is approximately 45 squared arcsec for PACS ad 2 squared arcmin for SPIRE, but a typical Herschel observation covers a much larger sky area, up to some square degrees wide. To observe the area, the telescope is moved along a set of parallel scan lines covering the area. During the scan the bolometers are sampled at frequency f_s, producing a sequence of readouts for each bolometer which is termed a timeline. Redundancy is guaranteed by observing the area at least twice, along two different scan directions, so that each bolometer normally produces two or more timelines. The timelines, together with the corresponding pointing information, constitute the observation output and are collectively referred as the Time Ordered Data (TOD). Ideally, each readout gives the emission, integrated over the Point Spread Function (PSF), received from the direction into which the bolometer is pointed at the sampling time. In practice there are a number of deviations from this assumption, the most relevant of which are reviewed in the next subsection.

A. Noise and disturbances

Each timeline is affected by a thermal and electronic noise [4], [5], which is modelled as a zero mean, stationary, Gaussian random sequence n[k]. The noise power spectrum, denoted by $P_n(f)$, can be expressed, for —f— = fs/2, as the sum of two terms, namely a white noise with flat spectrum Pw(f) = N0 plus a correlated noise with spectrum $Pc(f) = (f0/f)aN0$ where $f0$ is called the knee frequency and a the frequency exponent. The correlated noise is also referred as the $1/fnoise$ and dominates the spectrum at low frequencies. There are several deterministic disturbances affecting the timelines. The first is a drift, i.e. a deviation from the baseline, an example of which is shown in figure 1A. The drift is slowly varying with respect to the sky signal and the noise but may be much larger than the signal, especially for

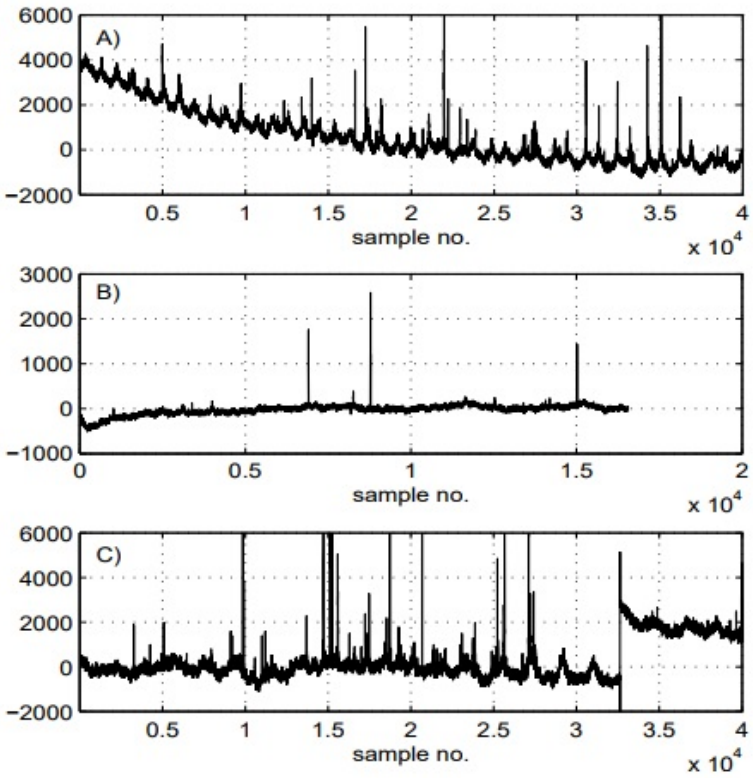

Figure 4.1: Examples of timeline. Note the large drift in plot A, the initial onset and the three glitches in plot B and the jump in plot C.

PACS3. It can be modelled as a low order polynomial curve [19]. Both a specific drift, different for different timelines, and a common drift, equal for all the timelines of an array or subarray, can be observed. A second disturbance is an onset, i.e. a deviation from the baseline affecting the initial part of the timeline, due to the memory of the calibration phase that precedes the scan. The onset can be roughly modelled as a decaying exponential and an example is shown in figure 1B. Finally, each timeline is affected by an offset, i.e. an unknown additive factor. The timelines are also affected by impulsive disturbances, due to cosmic rays. Specifically, when a ray hits the instrument, it may provoke a glitch or a jump, depending on where it hits. A glitch is a high spike in the timeline, lasting one or two samples, examples of which can be seen in figure 1B. A jump is an abrupt and long lasting deviation from the baseline, shown in figure 1C. When the ray hits the readout electronics, the jump may affect a whole row of the array and be seen in all the corresponding timelines. Another problem that may be regarded as an impulsive disturbance is the saturation which occurs when a bolometer is pointed towards a very bright source, which may affect several consecutive samples. The timelines are affected by several other disturbances, like quantization noise, Relative Pointing Error (RPE),

onboard compression, bolometer memory effects and interference from the solar panels. While these disturbances do degrade the image, they have less impact and will not be considered in this paper.

B. Data model

Assuming that the TOD is composed of Nt timelines and that the $k-th$ timeline is composed of Nk readouts, the data vector d is obtained by stacking the timelines and is a D1 vector, where $D = PNtk = 1Nk$ is the total number of readouts. Then, in order to use (1), we need to specify the signal model of (2) and the statistics of the noise. The latter can be derived from the noise spectrum, as we see later. Concerning the signal, we assume that the observed sky is pixellised, i.e. that it is partitioned into a grid of M non-overlapping squares (pixels) where the flux is constant, and is represented by an M 1 vector m. Next, we assume that the signal component of each readout is equal to the value of the sky pixel where the bolometer was pointed at the sampling time. Then the matrix P of (2), which is termed the pointing matrix, is a DM, sparse, binary matrix, constructed from the pointing information and such that Pk,i is 1 if the $k-th$ readout falls into the $i-th$ sky pixel and is 0 otherwise. Clearly, our signal model is an approximation. In fact, the readouts do not yield exactly the emission coming from the pointing direction but the emission integrated over the telescope PSF. This could be taken into account in the pointing matrix, e.g. [6], but at the price of a drastic increase in the complexity, which is undesirable for Herschel data. On the other hand, the approximation is good if the pixel size is comparable to the PSF size. Furthermore, the choice of a sparse and binary pointing matrix greatly simplifies the processing. Indeed the matrix can be efficiently stored. Moreover, the LS estimate of (3), which is also called the simple projection or the naive map, can be computed with D sums and M divisions, since the value of each pixel is just equal to the average of the readouts falling into the pixel. Finally, the production of the signal estimate of (4), namely $\emptyset s = P\emptyset m$, termed the backprojection of $\emptyset m$, only requires D memory access operations. A second approximation implicit in our signal model is that the sky is continuous in nature and not pixelised. Again, the approximation is good when the pixel size is comparable to the PSF size. Naturally the pixel size also has to be large enough that several readouts fall into each pixel, so that D ¿¿ M and the redundancy needed for the image formation is guaranteed. Both conditions can be satisfied for Herschel data.

4.2.3 Unimap overview and preprocessing

Unimap is written in Matlab language and is a pipeline composed by eight modules, identified by mnemonic names. Each of the first five modules compensates one or more disturbances and produces an updated TOD. In the last three modules the TOD is frozen and several sky images are produced. Due to the offset affecting the

timelines, these images are themselves affected by an unknown offset, which has to be estimated using an external reference (calibration). The processing is controlled by a set of parameters, the main of which are summarized in table I. There is a set of default values and the images produced using the defaults are normally good. In this sense Unimap is an automatic mapmaker. However sometime the image quality can be improved by tuning the parameters. To help this task, the program produces a set of evaluation images, summarized in table II, that will be described in the paper. The input to Unimap is a set of Herschel observations, each one stored in a separate file. Typically, but not necessarily, the input is obtained from the Herschel Interactive Processing Environment (HIPE) [20], which is the standard processing pipeline maintained by the ESA4. Specifically, each file contains a set of readouts organized into timelines and the corresponding pointing information. For each readout also an input flag must be in the file, which is a binary value indicating the validity of the readout. The flagged readouts will be excluded from the image synthesis and it is the user's responsibility to flag the readouts affected by disturbances which are not treated by Unimap. Typically, it is sufficient to flag the saturated readouts. Unimap can handle any number of input observations, the only limit being the memory size. Moreover, it is not memory eager, compared to other mappers, and wide images can be processed on a personal computer. To give an idea, an input dataset of 400 Msamples can be processed on a laptop with 8 Gb memory, with a processing time that depends on the data and may range from 2 to 20 hours. The memory requirements are approximately proportional to the size of the input data. A. Data formatting and offset compensation The TOP module loads the inputs and builds the TOD d and the pointing matrix P. Specically, given a user-specified pixellisation of the observed area, the module assigns each readout to a pixel and produces a $D1$ vector p such that pk is the index of the pixel where the k-th readout falls. This vector is called the Time Ordered Pixels (TOP) and is an efficient way to store the pointing matrix. The module discards timelines where the number of flagged readouts exceeds a user-specified percentage. Moreover, the module may discard the initial samples of each timeline, which is a brute force approach to remove the onset. Finally, the module performs a rough offset compensation, by subtracting to each timeline its median, thereby producing an updated TOD that, in order keep the notation simple, is still denoted by d. The module saves the naive map obtained by projecting the updated TOD and a second image, termed the noise map, giving the corresponding standard deviation. That is, for each pixel, if x is the vector of the readouts falling into the pixel, the naive map gives mean(x) and the noise map $\sqrt{var(x)}$. B. Jumps and onset The Pre module detects the jumps using a procedure described in appendix VIII-A. Next, a set of jump flags is produced, by flagging 100 readouts starting from the position of any detected jump. Typically, the percentage of flagged readouts is less than 0.5% for PACS, while the jumps are almost absent in SPIRE data. Finally, the module

merges the input and the jump flags and fixes all the flagged readouts. Specifically, the vector d is updated by replacing the flagged readouts with a linear interpolation between the preceding and succeeding valid readouts. This is a rough approximation, useful to regularise the timelines during the pre-processing. In a later stage the flagged data will be removed. As a second task, the module performs the onset removal which is controlled by a parameter λ, specifying the length of the onset. An exponential curve is fit to the first λ samples of each timeline and the TOD d is updated by subtracting the fit from the original timeline. The onset removal must be used with caution, because the exponential model is not always adequate. The module saves a flag mask, i.e. an image where each pixel gives the number of flagged (input or jump) readouts falling into it. The module also saves the naive map obtained by projecting the updated TOD and the corresponding noise map. C. Glitches The Glitch module detects the glitches by checking the readouts falling into each pixel and by identifying the outliers, as described in appendix VIII-B. A glitch flag is set for every readout affected by a glitch. After the detection, the TOD d is updated by linearly interpolating the flagged readouts. The glitch detection is reliable only if a sufficient number of readouts, some tens, falls into each pixel. However this condition is not always guaranteed for Herschel data. In this case, a coarser pixellisation can be used in the detection, in order to include more samples into each pixel. This is controlled by a user specified parameter η that is an integer specifying a multiplicative factor for the pixel edge. Overall, the procedure removes the glitches with an acceptable level of false alarms. A sane glitch rate is no more than 0.5% for both PACS and SPIRE data. A known problem is that the procedure tends to over-flag pixels containing point sources or strong signal gradients, since in these pixels the assumption of a constant signal is less valid. The module saves a glitch flag mask, which can be used to check over-glitching. It also saves the naive map obtained by projecting the updated TOD and the corresponding noise map.

D. Drift removal The Drift module removes the drift affecting the timelines. Since at this stage all other disturbances have been removed or are negligible, the TOD is modelled as

$$d = s + y + n \tag{4.6}$$

where s is the signal, n is random noise and y is a drift vector, constructed by stacking the drifts of all the timelines, each of which is modelled as a polynomial of order Na, depending on Na + 1 coefficients. The drift is removed using an algorithm based on Alternating Least Squares (ALS) which is a well known optimization method, e.g. [21]. The algorithm is one of the Unimapís original features5 and consist of the following steps Repeat 1-6 until convergence: 1. Make a naive map: mØ = (PTP)-1Pd. 2. Estimate the signal: Øs = PmØ. 3. Remove the signal: w = d-Øs. 4. For each timeline: 4.1 Extract the corresponding subvector from w. 4.2 Fit a polynomial to the subvector by means of LS. 5. Stack the polynomials obtained in step 4.2 into

vector yØ. 6. Remove the drift from the TOD: d = d-yØ. The algorithm performs a sequence of alternating signal and drift LS estimations. Indeed, the vector Øs of step 2 is the LS estimate of the signal while step 4 implements the LS drift estimation. The key point is that the drift is estimated from the vector w, from which the signal has been removed, thereby avoiding the signal leakage discussed in section II-B. Naturally, the signal estimate Øs is biased, due to the presence of the drift. Then, also the drift estimate yØ is biased. But removing it from the TOD, in step 6, we obtain an improved data vector, where the drift is partially removed. The process can be repeated to improve the drift removal at each iteration. Note that w is the LS fit error, from which the MSE can be computed. The algorithm is analysed in [24], showing that it converges and that the updated TOD is given by

$$d = s + c + \tilde{n} \tag{4.7}$$

where c is a constant vector and nò a random vector. Comparing the latter expression with (6) we see that the algorithm preserves the signal and removes the drift except for a constant. However, this is irrelevant, since c can be regarded as an offset affecting the timelines and is removed in the calibration phase. The statistics of the noise nò can in principle be obtained from those of n but this is not necessary since we are going to directly measure them. The convergence is controlled by checking the MSE, stopping the iterations when the MSE improvement gets below a user-specified threshold. Similarly, the polynomial order, which is user-specified, can be selected by checking the MSE. In particular, the MSE decreases for increasing polynomial order and an adequate order is when the improvement becomes negligible. In practice the order should be no more than, say, 5, because at higher orders the polynomial model is not applicable. The algorithm can be extended to other drift models, like sinusoidal or piece-wise linear. Moreover, it can be adapted to a common drift, by assuming that the same drift affects a group of timelines. Indeed in Unimap the user can select between either removing a drift from each timeline or removing a drift from each array (SPIRE) or subarray (PACS). The second option is faster but less performing, because it leaves a residual, specific drift. However this is not a problem since the specific drift can be regarded as correlated noise and is suppressed by the GLS image synthesis. The module saves the naive map obtained by projecting the updated TOD and the corresponding noise map.

4.2.4 GLS image synthesis and post processing

From (7), by neglecting the constant and dropping the tilde, we obtain (1). Then we can use (5) to obtain the GLS estimate. However mg cannot be calculated directly, due to the size of the matrices involved. A way out of this problem is to rewrite (5) as $Amg = m^*$, where A = PTN-1P and m* = PTN-1d, and note

that the latter is a linear system in the unknown vector mg, which can be solved using a standard method. Typically, e.g. [6], [8], [10], the system is solved using the Parallel Conjugate Gradient (PCG) [22] which is an iterative solver that, to be implemented, only requires to compute the vector m* and to perform a sequence of multiplications of the matrix A with a vector. Both these operations can be broken into sub-multiplications with the matrices P, PT and N-1. The multiplication by P or PT is a projection and can be implemented efficiently since P is sparse. On the contrary the multiplication by N-1 is more involved and is discussed in the next subsection.

A. Multiplication by the inverse covariance matrix We start by assuming that there is a single timeline. Then the vector n is a window of D samples of a noise sequence6 n[k] for$k = -\infty, \ldots, \infty$, having an autocorrelation sequence $R[k] = E\{n[i]n[i+k]\}$ which is the Inverse Fourier Transform (IFT) of the noise spectrum Pn(f), denoted as $R[k] = IFT\{P_n(f)\}$. Since $P_n(f) > 0$, we introduce H(f) = 1/Pn(f) and the sequence h[k] = IFTH(f) which is called the noise filter response. Since Pn(f) is real and symmetric, the same is true for H(f) and h[k]. Moreover, in practice, the sequence h[k] tends to zero and there is an index L such that $h[k] \approx 0$ for $|k| > L$. Consider now the multiplication of N-1 by a generic $D1$ vector v, to produce $w = N - 1v$. In appendix VIII-C we show that, when $L << D$, the vector w can be approximated as follows. First, from v we construct a sequence $v[k]$ by padding and mirroring the vector: $v[k] = vk + 1$ for $k = 0, ..., D - 1v[k] = v - k$ for $k = -L, ..., -1v[k] = v2D - k$ for $k = D, ..., D + L - 1v[k] = 0$ elsewhere. (8) Next, we filter the sequence $v[k]$ with the response $h[k]$, to produce $w[k] = v[k] \times h[k]$, which can be done with $O(2DL)$ multiply-add operations. Finally, the vector w is obtained by extracting the rest D samples from the sequence $w[k]$, i.e. $wk = w[k-1]$ for $k = 1, ..., D$. The extension to the case of $Nt > 1$ timelines is straightforward. Indeed, since the noise processes of the timelines are independent, the matrix N is block diagonal with Nt blocks and the reasoning can be repeated for each block. Then, the multiplication $w = N - 1v$ can be approximated by breaking the vector v into timelines, separately filtering the Nt timelines and stacking the results into w. B. Estimation of the noise filter response The Noise module estimates the filter responses. As a first step, an estimate of the noise vector is produced, given by $n\emptyset = d - \emptyset s$ where $\emptyset s = PT(PTP) - 1Pd$ is the signal estimate. Next, the filter responses are computed, separately for each timeline. In particular, given a timeline, we compute only the non zero part of $h[k]$, which is a $T = 2L + 1$ samples long sequence7. This sequence can be obtained as the Inverse Discrete Fourier Transform (IDFT) of $H[i] = 1/Pn[i]$, where $Pn[i] = Pn(ifs/T)$ for $i = 0, ..., T-1$ is a sequence of T samples of the noise spectrum. In turn, the sequence $Pn[i]$ can be measured from the noise estimate $n\emptyset$. This is done by extracting from $n\emptyset$ the subvector corresponding to the timeline, by segmenting the subvector into blocks of T samples with an overlap of L, by computing the Discrete Fourier Transform (DFT) of each

block and by averaging the squared magnitude of all8 the DFTs. Note that, due to the $1/f$ noise, we have $Pn(0) = 8$ and correspondingly we set $H[0] = 0$, implying that the filter response is zero mean. This guarantees that when the timelines are filtered to produce the map m*, any residual offset is eliminated. One problem is that the signal may leak into $n\emptyset$ and bias the spectrum measurement. A first way to mitigate this problem is to separate the measurement of the spectrum shape and amplitude. To this end we normalise to unity the DFT of each block before the averaging and produce a normalised response $\emptyset h[k]$, enforcing $\emptyset h[0] = 1$. Next, we estimate the noise power P as the median of the variances of the blocks. Finally, we compute the response9 as $h[k] = \emptyset h[k]/P$. As a second improvement, we fit the measured spectrum $Pn[i]$ to the $1/f$ plus white noise model and obtain a sequence Ø Pn[i] which follows the theoretical spectrum and is used to produce the filter response instead of the measured spectrum. However, since the model is not always applicable10, the user is left the option to skip the fit. The module saves the measured noise spectra and the filter responses.

C. Flagged data removal As a second task, the Noise module removes the flagged readouts. In principle this is simple: to remove a flagged readout it is sufficient to remove the entry from the vector d, the corresponding row from the matrix P and the corresponding row and column from the matrix N; then, the GLS map is still given by (5). However this is not easy to implement when the multiplication by N-1 is realised as a filtering. Therefore an original, approximate approach is used, which is described in the following. As a first step, the flagged readouts (input, jump and glitch) are removed from the TOD d and the TOP p, shifting the other elements downwards to fill the gaps. In this way an updated, shorter TOD is produced, where the time continuity of the noise is not preserved. However this is not a serious problem as long as the flagged readouts are isolated samples. On the contrary, when a long sequence is removed, the problem is exacerbated. For this reason, when a sequence of five or more samples is removed, the corresponding timeline is broken in two segments, which are treated from now on as independent timelines, albeit with the same noise filter. In particular, this guarantees that the timelines are broken in correspondence to all the detected jumps. This is important because in this way the jump translates into two different offsets for the two segments, which are eliminated by the zero mean noise filter.

D. GLS Image synthesis The GLS module carries out the image synthesis, by running the PCG algorithm, which is an iterative solver and needs an initial guess to start with. The user can select between the naive map or an all zero map. In principle the PCG will produce the same solution, irrespectively of the start image. However, the iterations required to achieve convergence depend on the start image. Moreover, convergence may be diffcult to achieve if the start image is not properly selected. Typically, when the image is signal rich it is better to start from the naive map and convergence requires a few tens of iterations; when the image is dominated

by the background it is better to start from the zero map and convergence may take a few hundreds of iterations. The main outputs of the module are the GLS map mg, the naive map mn and the corresponding noise map ms. The module also saves a coverage image m_c, giving the number of readouts falling into each pixel.GLS approach is effective against the correlated noise. Unfortunately, the approach also has a serious drawback, namely it may introduce distortion11.The distortion takes the form of a cross-like artifact placed on a bright source. However, as we see soon, the distortion can take other forms and is not always so obvious. The distortion is better discussed in [25] and is due to the approximations implicit in the signal model and to the uncompensated disturbances, e.g. the RPE. The distortion is not present in the naive map, which is instead affected by the correlated noise. In order to evaluate the presence of distortion, the GLS module saves a map $eg = mg - mn$ which is the difference of the GLS and the naive maps. Since mg is constituted by the signal plus distortion and mn by the signal plus correlated noise, by taking the difference we remove the signal and we expect eg to be constituted by the correlated noise (with negative polarity) plus the distortion. By inspection of hundreds of images we verified that the distortion is a serious problem, affecting all the GLS mapmakers. The distortion is more pronounced for PACS but not uncommon in SPIRE images. Cross-like distortion is due to bright, narrow sources and is often found in PACS blue and green images. Diffuse distortion, which can reach 15% of the real emission, is due to broad sources or to diffuse emission and is more frequent in PACS red and SPIRE images. Often the two types of distortion are mixed and both present in the same image. E. Distortion removal The PGLS module removes the distortion introduced by the GLS synthesis, by running the Post-processing for GLS (PGLS) algorithm, which is another Unimapís original feature12. The algorithm is described in [25] and will be summarised here. The inputs are the TOD d and the GLS map mg and the output is the PGLS map mp . The algorithm amounts at the following steps Init $mp = mg$ and repeat 1-5 until convergence: 1. Estimate signal plus distortion: $t = Pmp$. 2. Remove signal: $r = t - d$. 3. Remove correlated noise: $w = high(r)$. 4. Remove white noise and estimate distortion: . $md = (PTP) - 1Pw$. 5. Subtract distortion from the map: $mp = mp - md$. Assuming that the GLS synthesis removes the noise but introduces distortion, the map mg contains signal plus distortion. These components are maintained in the back-projected TOD t computed in step 1. Since the original TOD d contains the signal, the correlated noise and the white noise, by subtracting it from t in step 2 we obtain a TOD r where the signal is removed but the noise (with changed polarity) and the distortion are present. By performing the median high-pass filtering of r we eliminate the correlated noise. By projecting w in step 4, we eliminate the white noise and produce a map md which only contains the distortion and is subtracted from the signal map, in step 5. Naturally md is just an estimate and the distortion removal is not perfect. However by repeating steps 1 to 5 more

distortion is removed at each iteration. Convergence is achieved when the map mp does not change significantly across two iterations, which typically happens after three or four iterations. The PGLS effectively removes the distortion, provided that there is enough redundancy in the data. In particular at least two orthogonal scans are required. The only drawback is an increase in the background noise which is due to the fact that the median filtering and the naive mapping do not perfectly remove the noise, so that the distortion estimate md is noisy and this noise is injected into the PGLS map when md is subtracted from mp in step 5. However when the signal is strong the noise increase is negligible. The PGLS has a single parameter, namely the window length of the median high-pass filter, which can be set using a trial and error procedure. In particular, wider filters improve the distortion removal but also inject more noise. The module saves the PGLS map mp and the total distortion estimate $e = mg - mp$. It also saves the difference of the naive map and the PGLS map, $ep = mp - mn$, which can be used to evaluate the results and to set the window of the median filter.

F. Noise minimization The WGLS module minimises the noise injected by the PGLS by running the Weighted post-processing for GLS (WGLS) algorithm, which is an additional original approach. Specifically, the module detects the distortion, using a procedure described in appendix VIII-D, and produces a mask me which is one in the pixels where distortion has been detected and is zero elsewhere. Next, the module computes a map by subtracting the PGLS distortion estimate from the GLS map, but only in the pixels where distortion has been detected, thereby avoiding the noise injection in the pixels with no distortion. Formally, by usingto denote the elementwise multiplication of two vectors, the WGLS map is given by $m_w = m_g - e \odot m_e$. The module saves the WGLS image mw, which is the final Unimap map, and the mask me.

4.2.5 Performance evaluation

To evaluate the performance, we developed a simulator. In order to account for the continuous nature of the sky, we produce a synthetic but realistic sky image, having a higher resolution than the map to be formed, i.e. a lower pixel size (one fourth). Next, using the pointing information of a true Herschel observation, we sample the synthetic sky to produce the signal vector s. Then, a realistic example of the Herschel disturbances is obtained by picking a true TOD, denoted by w, produced by observing a sky area free of emission13. Finally, the simulated TOD is constructed by adding the signal and the disturbances, as $d = s + gw$, where g is a gain factor that is used to set the Signal to Disturbance Ratio (SDR), defined as $SDR = var(s)/[g2var(w)]$. Moreover, following [23], we produce a target map mt as the naive map obtained by projecting the signal vector, i.e. as $mt = (P^T P)^{-1} PT$ s. Given the setup just described, we feed the simulated TOD to Unimap and produce

the output maps. To evaluate the quality of, say, the GLS map mg, we compute the Image to Error Ratio (IER) that is the ratio of the target image variance to the error variance, i.e. $IER = var(mt)/var(mg - mt)$. The IER is computed similarly for the other maps. The PGLS is inferior, indicating that it introduces background noise. However the WGLS is capable of removing the noise and attains the same IER of GLS. The naive map is plagued by the correlated noise and is the worst one. When the SDR is increased, all the maps get closer to the target. However, the GLS performance saturates at some point, due to the distortion. At this point, the PGLS yields a higher IER, indicating that it successfully removes the distortion. Moreover, the WGLS attains a further IER increase and yields the best map. At high SDR the WGLS and PGLS performance is the same, indicating that the noise injected by the PGLS is negligible. At very high SDR, even the naive map becomes a good choice. We conclude that the WGLS map is the best one across the whole range of SDR, but at high or low SDR other maps can have a comparable quality. For a deeper performance analysis the reader is referred to [13]. Obviously, for true data the IER cannot be computed and we need to evaluate the map quality by other means. To this end note that we can write the value of a generic pixel as $v = s + e$ where s is the target value and e is a Random Variable (RV) representing the error. Then, a useful performance figure is the variance of e, denoted by $\theta^2 e$. The error variance can be estimated from the Unimap outputs as discussed below. Consider the naive map and denote by v1,...,vK the readouts falling into a generic pixel. These can be written as $v_i = s + e_i$ where s is the target value and ei is a RV representing the disturbances. Since the pixel value is $v = mea(v_1, \ldots, v_K) = s + e$, by assuming that the RVs ei are uncorrelated and have the same variance, denoted by $s2$, we have $se = s/vK$. Then, se can be approximately computed, since an estimate of s is given by the noise map ms and K is given by the coverage map m_c. For the other maps the error variance is more difficult to estimate. An expression exists for the GLS map, e.g. [8], but it is mainly of theoretical interest, since it does not account for the distortion. However, at least for the WGLS map, we can assume that the error variance of the naive map is an upper bound. In fact the naive map is plagued by the correlated noise which is instead absent in the WGLS map. Indeed we studied the error by means of simulations, confirming the validity of this assumption. The results also indicate that the error is not Gaussian. However, a deeper discussion of the error distribution is out of the scope of this paper.

4.3 The Unimap simulator

The Unimap software is more optimized by every single version. At the moment of writing this thesis the last official version is 5. Thanks to the modular structure of Unimap these optimizations can be applied independently, one by one or

together which is a great help for checking changes and their effects, but still there is a problem. While the real image and the noise characteristics and amount are unknown the effects are unknown based on noise behaviors. The need for studying the changes based on the behaviors of different noises affecting the signals made us to develop a simulator for this software. The goal for this simulator is to make an input data for Unimap which has known noise amount and the same characteristics of the real noise affecting the images. Using this simulator we can study the effects of new changes on different amount of noises.

4.3.1 Simulator specifications

This Simulator is developed with several specifications, we have listed some of them here:

- Input parameter file:This simulator function doesnít have any direct parameter input and read all of them from an external file. This feature makes it possible to produce a compiled version of software which can be run in Linux.
- The results: The simulator produce the results in the same directory of the input data, so , using a batch file, it is possible to run several simulations in serial without any conflicts and no need to move the results files.
- Noise control: The amount of every single noise type can be tuned independent of each other and even turned off. This will help us studying the effect of different types of noises independent, one by one or all together.
- In house sub functions: There are required sub functions in simulator which are available in Matlab itself, but for more control on these functions they have been developed in simulator itself from scratch. Easier bug detection and faster functions are other goals of this approach. As an example we can name upsampling function. This function is done using FFT which will be explained in details in upcoming chapters.

One of the main strategies in studying the features of a software in noise reduction applications is to freeze all other noises and change one single variable and then study the output. While this is already a well known and standard method, we need even a more professional tool to study the behaviors of this software. With respect to all theoretical and scientific methods, this software is a practical tool which is based on signal processing methods.

4.3.2 Simulator inputs

The simulator is reading its parameter input from a file available in the same directory of simulator function itself. These parameters are divided in different

categories: Input data and true sky:

- data_path: The Simulator will read a char variable from this line, this variable refers to the data path where the simulator will read the reference image.
- sim_sky: A char type variable which will show the image name which is going to be used by Simulator.
- sim_pix_size: The pixel size of the image which is going to be used by simulator. This is available in meta data of the image itself but due to some tests we needed a tunable pixel size.
- sim_up_rate: As a part of simulator there is an upsamling function which is increasing the resolution of the data, this will help us studying the effect of noise reductions in high and low resolution images.
- sim_PSF: There is PSF function built in the simulator, the size of PSF filter is decided by this parameter which is in arcsecond.
- sim_frame: Because of the upsampling, there is a border effect around the image. This can be reduced by adding an extra border around the mage which smoothly goes to zero.

4.3.3 Simulator paramters

As we have mentioned before, The Simulator reads the parameters from a TXT file, now we will check these parameters:

- **Data reduction parameters** sim_sample_subs: Due to our needs and the size of data, there might be the need to reduce the processing time, this factor will read the parametrs one in every several, so the data size is reduced and in result the processing time. sim_bolo_subs: This factor has the same functionality of sim_sample_sub on the bolometer data. sim_reset_flag: This will reset the flags to the initial values, otherwise it will use the latest flags available for processing the image
- **RPE parameter** sim_RPE: One of the noises added by Simulator is RPE. This noise can be controlled by this variable, sim_RPE is shift in samples and can have a value between -10 to 10. 10 is actually an exaggerated value since the practical numbers won't be even bigger than 1
- **real noise parameters** sim_noise_path: This is chat variable which will show the path for the real noise. The real noise will be loaded and added to the image based on the sim_rnoi_SNR sim_rnoi_SNR: The SNR related to real noise, There will be no real noise if this variable is assigned to 'inf'

- **simulator drift parameters** sim_drift_coe: Another noise which can be added using this simulator is drift. This parameter is common drift coefficients. sim_drift_deep: deepness of variation for uncorrelate drift, this variable is in percentage. sim_SDR: Signal to drift Ratio, using this variable we can control the amount of drift added to the data.

- **simulated noise** sim_fknee: This is an artificial noise to be added to data, this variable is the simulated noise frequency knee sim_nexp: variable for controlling the noise exponential factor sim_SNR: Signal to Noise Ratio of simulated noise. Using this factor we can control the amount of noise added to data, if this is set to inf, the noise will not be added.

4.3.4 Simulator outputs

Simulator saves a file with the name unimap_obsid_sim which is .mat file and contains the information of a new image. This file can be passed to Unimap for reduction and while all the information about the noises are known, it can be a great test for efficiency of new modules in Unimap. Moreover than the main output file, the Simulator also saves the Matlab log file, and simulator parameters file for later uses.

4.3.5 How does it work

The Simulator starts with reading the parameters from the parameter file. It immediately saves the parameter file in the directory of reduction. As mentioned before, having a copy of the parameter files helps to remember the simulation variables.Thanks to the parameter file we can find out the amount of different noises and all changes which the simulator has done to the image. The procedure of processing starts with widening the image. This act is done to get over the border effects which will be seen in borders of images after upsampling . The image size is increased regards to the input method "sim_frame" which is provided in the parameter file. In our real time tests it has been fixed at 10 but totally depends on the image size. The widening segment adds rows and columns to the beginning and end of image with value on one. This help smoothing the image at the borders so we won't see any artifacts at the borders of the image. Then we start the upsampling procedure which is developed specifically for this simulator. The upsampling is working based on FFT. The inputs of this function are the image itself and the up sampling rate. The output will be an image with the size of the original image multiply by the up sampling factor to the power two.

The up sample function produce a matrix with the same size of output matrix,which is n times bigger than original image, with all elements equal to one,

then it inserts the real pixels in to the new matrix as there will be “n” new pixels between every real pixels. Now comparing the fourier transform of these two images shows us the fact that they are different in frequency range and high frequency. Regards to these result we understand that the high frequencies are the changes from real data and zeros in the up sampled image, so by removing the high frequencies and then transfer the signal back, we will have the original image with smooth transition from any pixel to the next. At the end it increase the power of the result to match the input power. Here are an example of the results for the famous picture ”lena” which is cropped to show the difference better. We have to mention that for using a jpg file we have modified the function a little bit because the original function just accepts the fits file

Figure 4.2: The original image (256*256pixel) on the left vs the 5 times upsampled image on the right

Next we have the “taper” function. This function is removing the NaN elements from image and also normalizing the image over the segment which is real.

After the taper, it apply a PSF. This PSF is removing some artifacts due to up sampling function. Obviously it is applied only if the up sampling function is active.

The next step in Simulator is to load the real noise and save it in a parameter for further steps. The real noise is captured by the satellite itself. This real noise is captured using the real sensors but with closed lenses, so in no light situation and the result is zero data and just noise.

Then the observation process starts. It starts with checking all the files in the loaded directory from parameters. It loads all the files with specific name “unimap_obsid_****.fits”. After some basic calculations it calls the sim_obs function which is the main function of this simulator. We will discuss this function later in this chapter. After using this function the results are ready and the Simulator just save the results in the same directory which it has read the data from.

Sub function sim_obsid After simple calculation like converting the pixel size and checking if variables are in the correct range, This functions for the first step

starts to add the RPE noise. In Unimap, the pointing information is obtained from a true Herschel observation. In order to produce the simulated data vector, it perturb the pointing information by shifting each sampling point along the scan direction, by a fraction of the sampling spacing, which mimics the presence of a RPE in the data. Here we have to shift back the each sampling point in the way that it reproduce the RPE noise based on the parameter of the input file. The number is in arcsec and can be between -10 to +10

After adding RPE, The simulator starts the projection. The raw data is in spherical format and it should be projected to a flat plane surface. This is done thanks to a function available in the library of Unimap named gnomonic_projection. In the next step we have to make the real image. Up to now we have been working on a data which is not the image itself, it contains the information of pixels, and there are two other matrix named u_ra and u_dec which contains the pointing information. In other words the two matrix of u_ra and u_dec show us the real place of the information in the matrix. So in this step, using these information, we shape the real image. Then we calculate the signal power for using in the next steps. While the amount of noise to be added is given in SNR, we need to know the signal power before adding the noise. After this computation we start to add the real noise regards to the SNR read from parameter file. The next step is adding the artificial noise. The noise is produced by the function mk_noise. This function produce the noise regards to it's input, the power of noise and the sim_fknee which is the knee frequency introduced for the noise. Actually it first produce a white noise regards to the size of image and then it reduce the unwanted frequencies from it.

It then make the drift based on the drift parameter in the input file and then There is only one step before developing the wanted parameters for the output and closing function. If in the input file the subsampling is determined it's going to downsample the image. This is due to the huge data size of the images and reducing the size for the tests. This part is actually helpful in the very beginning steps when we need lots of tests and then optimizing the functions. When we are confident with the code we go further for a complete data simulation. Now we are done with the sim_obs function which stands for simulation observation.

After using the observation function of simulator, everything is ready and we form the output file using one of the function available in the Unimap software called 'fitswrite'

In the next chapter we are going to review the results of different noises and the unimap noise reduction of these images. These results are developed using a real data from Herschel satellite which we will go more in depth in the next chapter.

Chapter 5

Simulation results

In this chapter we are going to review the results of one real data source with several variations of noises and check the output of the last stable version of Unimap software. We have to declare that while we are using a stable version of the Unimap, the results are pretty satisfying, so there won't be any surprise when we check the results in even bad situations. Let's start with different variables in this simulations.

We are going to change the RPE noise. We have already discussed what the RPE is. In this simulation we are going to test 7 different variable for the RPE . First the zero amount which means no shift or no RPE noise. Then we will have +0.3 , +0.7 and the extreme value of +1.2. Since the RPE can be a forward shift or backward, we are going to test the exact same values backward as -0.3 , -0.7 and -1.2.

The other variable that we have for this test is the the SNR . We have already mentioned that we can add two different kinds of noises, the real noise and the artificial noise. The real noise is the data captured by the satellite with closed mirrors, so this is the most accurate noise that we can actually have and better than any other noise created by computers. Since the results for positive SNR were too much good, we prefer to continue the tests just for worse situations. We are going to do our tests for $SNR = 0$, -10 , -20 . So in total we have got 3 SNR level and for each of them 4-8 levels of RPE . We have to remind that every single simulation plus reduction of the data takes few hours of time which means we needed a day for every single reduction. For having a complete and close to reality reduction results we turned off the downsampling in the Simulator.

Before checking the results, we have to also consider that for the input we are using the $InputSNR$, but for the output there is a different method to measure the performance. This is due to the fact that the amount of noise where there was no object in the sky is not important for us, so we take into the account just those parts of the image which we have detected an object. This is done using another function developed by us for measuring the performance of Unimap with known noise types

and amounts.

The output SNR function: This function uses the morphology function and flag morpho variable for detecting the "area of interest". Based on input, we can decide to calculate the SNR just in the area where an object was detected, or object and a border around the object or the whole image. Respect to the point that we already have the "true sky" or noiseless image, we can calculate the absolute noise:

$err\_img = wgls - ref\_img$ Where the $err_img$is the absolute noise data, $wgls$ is the output of the last module of the Unimap software and the ref_img is the reference image which is the noiseless image. Then for calculating the SNR, we have:

$SNR\_sig\_wgls = SIG\_wgls\_sig \div ERR\_wgls\_sig$

While the SNR_sig_wgls is the calculated SNR , the SIG_wgls_sig is the power of the signal in area of interest and ERR_wgls_sig is the power of the noise in area of interest. We recall that area of interest is where the software has detected an object and a border around it.

Now we start to check the results: As we have mentioned before, we have got different noises, in this thesis we are focused on two of them, RPE and the instruments noise which we are going to address by SNR. After running a reduction over a data received from Herschel, we will fix the amount of noise and run the simulator, then the data is ready to test the Unimap. We will discuss for every single SNR different simulations with different RPE . The data is called Hope38 red. Below we have a picture of clean reduction results. Since the differences of simulations are not clear in the image, we will keep checking the parameters.

$SNR = 0db$

As we know, the SNR = 0 db means that the power of our noise is equal to the power of our image, now let's see how the Unimap works under this condition. At this SNR , the output SNR is listed below:

$OutputSNR@RPE = 0$	$OutputSNR = 38.651$
$OutputSNR@RPE = +0.3$	$OutputSNR = 29.256$
$OutputSNR@RPE = -0.3$	$OutputSNR = 29.313$
$OutputSNR@RPE = +0.7$	$OutputSNR = 22.236$
$OutputSNR@RPE = -0.7$	$OutputSNR = 22.389$
$OutputSNR@RPE = +1.2$	$OutputSNR = 17.430$
$OutputSNR@RPE = -1.2$	$OutputSNR = 17.635$

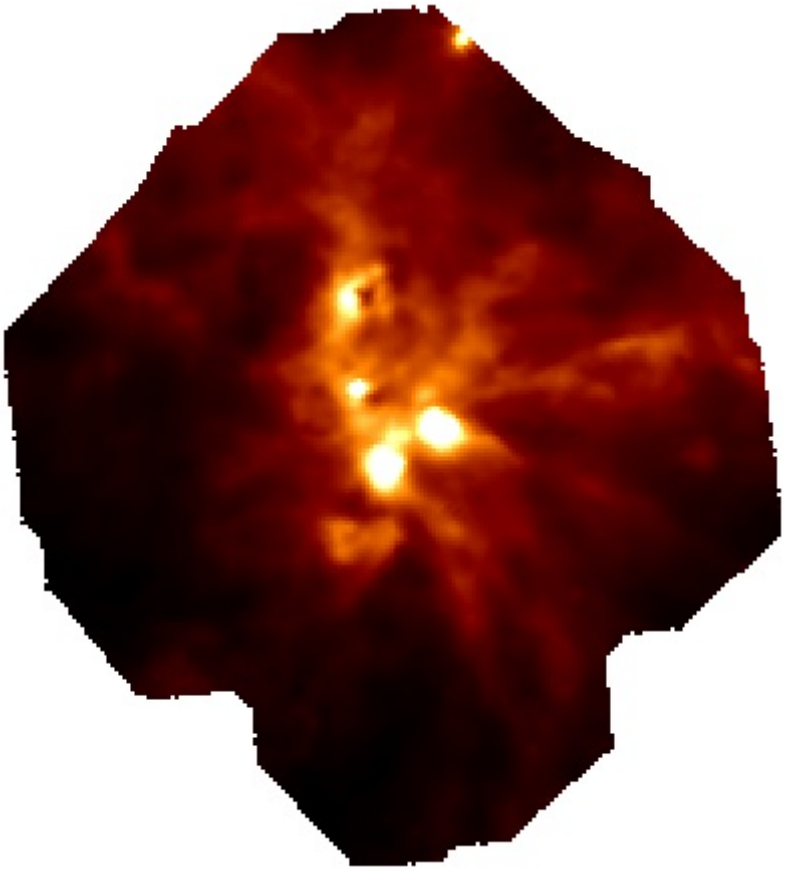

Figure 5.1: Hope-38 real data from Herschel satellite

As we can see from the results, Unimap was clearly able to remove the noise from image. In this case the area of interest was where we have detected an object. Another important point which we can see clearly from these numbers is that for unimap the sign of the RPE is not important and it does the same job for positive and negative values of RPE. For this reason we are going to continue just with the positive values of the RPE as we know the results will be the same for negative values.

$SNR = -10db$

We are going to test a very bad situation when we have got SNR = -10 db. As we have mentioned before we are talking about the real noise and this is the closest possible noisy image which can be produced for this kind of data. Here, we are going to test 4 different RPE values and check how the software is able to remove the noise in this situation.

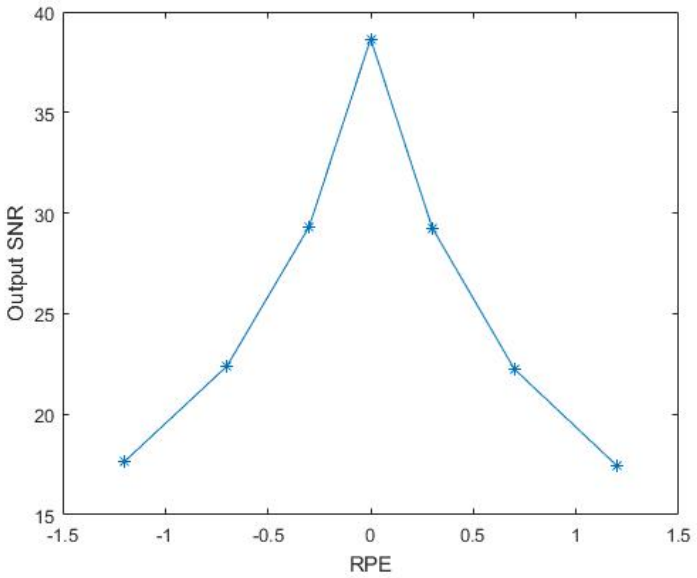

Figure 5.2: The output SNR change by different RPE at fixed Input SNR = 0

$$OutputSNR@RPE = 0 \qquad OutputSNR = 38.651$$
$$OutputSNR@RPE = +0.3 \qquad OutputSNR = 27.368$$
$$OutputSNR@RPE = +0.7 \qquad OutputSNR = 21.724$$
$$OutputSNR@RPE = +1.2 \qquad OutputSNR = 17.117$$

As we can see, again, while the results are pretty satisfying, but the effect of the RPE is obvious. This was actually the first time that we have tested the same signal with only one variable in Unimap software.

$SNR = -20db$

At this point we are going to test an extreme situation which probably we are never going to face in the reality, but it's definitely interesting for us to know how much this software can push the limits and still get good images out of noisy situation. Here are the results for these 4 reductions:

$$OutputSNR@RPE = 0 \qquad OutputSNR = 23.031$$
$$OutputSNR@RPE = +0.3 \qquad OutputSNR = 22.005$$
$$OutputSNR@RPE = +0.7 \qquad OutputSNR = 19.155$$
$$OutputSNR@RPE = +1.2 \qquad OutputSNR = 16.083$$

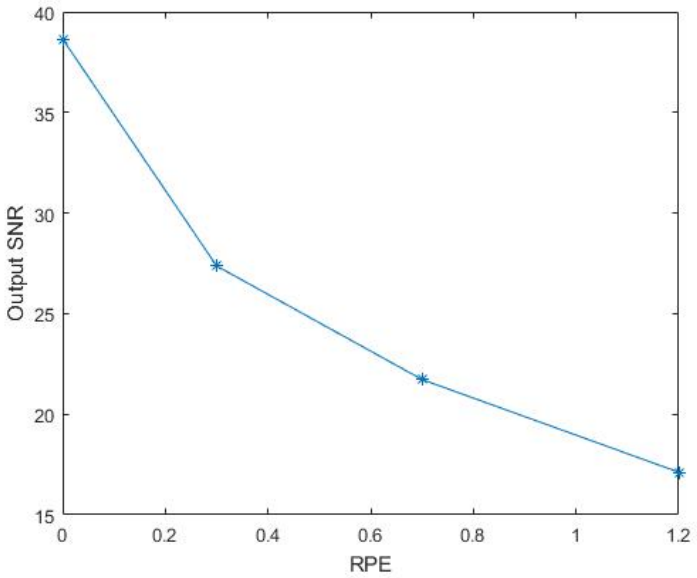

Figure 5.3: The output SNR change by different RPE at fixed Input SNR = -10

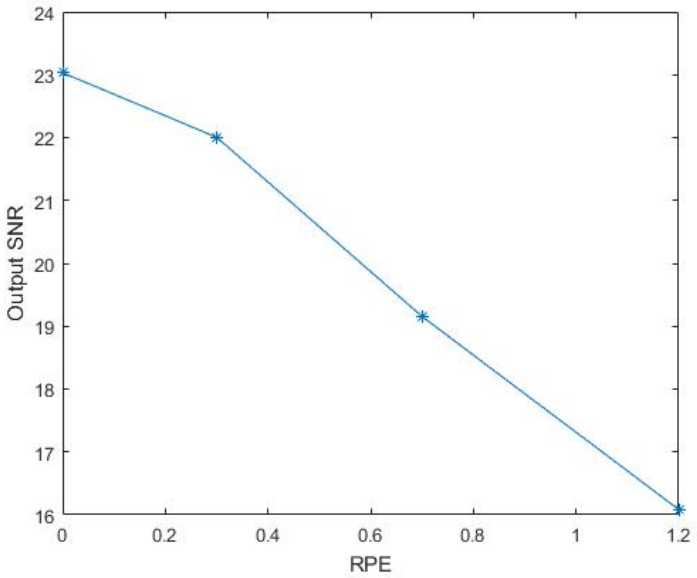

Figure 5.4: The output SNR change by different RPE at fixed Input SNR = -20

The results were really surprising. Considering we have tested the software with $-20db$ noise and still responding well, make us pretty confident that we have got a good margin for practical usages of this software. We can also check the same result in another point of view. Now, instead of checking the results for a fixed SNR and looking at the RPE as the variable noise, we are going to do the exact opposite.

We will fix the RPE and look at the SNR as the variables.

$RPE = 0$ We have got 3 different SNR values, let's check the results:

$$\begin{aligned} OutputSNR@inputSNR = 0db \quad & OutputSNR = 38.651 \\ OutputSNR@inputSNR = -10db \quad & OutputSNR = 38.651 \\ OutputSNR@inputSNR = -20db \quad & OutputSNR = 23.031 \end{aligned}$$

As the results, We can see that at low noises while not having the RPE noise, the software is working pretty well and actually can remove the low to medium noise levels pretty well. Now let's see how it behave at the presence of RPE. $RPE = 0.3$ Now we have added a low amount of RPE to check the software

$$\begin{aligned} OutputSNR@inputSNR = 0db \quad & OutputSNR = 29.256 \\ OutputSNR@inputSNR = -10db \quad & OutputSNR = 27.368 \\ OutputSNR@inputSNR = -20db \quad & OutputSNR = 22.005 \end{aligned}$$

As we can see, now the presence of RPE is affecting the behavior of software, but still negligible. Let's check the results with more RPE

$RPE = 0.7$

The 0.7 RPE is actually a practical amount of noise for Herschel, let's check the behaviors of Unimap.

$$\begin{aligned} OutputSNR@inputSNR = 0db \quad & OutputSNR = 22.236 \\ OutputSNR@inputSNR = -10db \quad & OutputSNR = 21.724 \\ OutputSNR@inputSNR = -20db \quad & OutputSNR = 19.155 \end{aligned}$$

The same situation is repeated here, we see the behavior is changed based on the higher RPE. Now let's check an extreme value for RPE

$RPE = 1.2$

The value 1.2 is a number which we might not see never in the results of the Herschel satellite, but let's have a look at the results.

$$\begin{aligned} OutputSNR@inputSNR = 0db \quad & OutputSNR = 17.430 \\ OutputSNR@inputSNR = -10db \quad & OutputSNR = 17.117 \\ OutputSNR@inputSNR = -20db \quad & OutputSNR = 16.083 \end{aligned}$$

As we can see, at this level the software performs much worse and it's obvious for us the reason. The RPE noise is affecting heavily even for the low amount of

noises. At this point, we have found out that any improvements in RPE module, will probably improve the output SNR, even for noises which have an independent noise source.

Chapter 6

Summary

In this thesis we started with a long introduction about astronomy image processing and then after reviewing the specifications of Unimap software we introduced the Unimap Simulator. This Simulator has become an essential part of this software since 2 years ago. This is both on academic and practical point of view. As the practical point of view now we can have a deeper, faster and more complete test on new modules. Not only we can check of the module is more optimized than before, but also we can track any effects on the other modules. As an example we saw that the RPE noise is affecting the other modules too and any improvements to any module can affect all parts of the software and also on the contrary, a small change in module A can affect the performance of module B. The same story is true about academic point of view. While there have been academic articles about the modules and theories which have been used in this software, they have been always tested with real data or very limited artificial data which were not close to practice, but since two years ago the results which have been prepared for the article were pretty solid. The results could show the behavior of the new improvements not only for the raw data captured by the satellite, but also for a range of different noise levels and noise types. Thanks to the measuring instrument which we have introduced during the development of the Unimap Simulator, there can be a very detailed and precise comparison between the new and old version of the modules and instrument. As an example, in two different IEEE articles. As an example in the article "Least Squares Image Estimation for Large Data in the Presence of Noise and Irregular Sampling" which was published in "IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 26, NO. 11, NOVEMBER 2017", the simulated data is produced by Unimap Simulator. In section B "Simulated data" of this article we have:

"We also realized tests using simulated data. The simulator is based on a synthetic but realistic sky image, which is used as the truth. In order to simulate the continuous nature of the sky, this synthetic image has a higher resolution than the estimate to be produced, i.e. a lower pixel size. The pointing information is obtained from a

true PACS observation. In order to produce the simulated data vector, we sample the synthetic sky according to the pointing information, to obtain the signal vector s. Next, a synthetic noise vector w is produced by filtering pseudo-random white Gaussian noise with a filter having frequency response $\sqrt{S(f)}$. Finally, the simulated data vector is constructed by adding the signal and the noise, as $d = s + \eta w$, where η is a gain factor that is used to set the Signal to Noise Ratio (SNR), which is defined as the ratio of the arithmetic variance of the signal and noise vectors, i.e. $SNR = var(s) \div [\eta 2var(w)]$. Using the simulator, we implemented several tests, varying the sky and the pointing information. In the following we present an example, based on an observation of the Alphaboo star in the red band. The SNR is set to 0 dB. To give an idea of the simulated data, the arithmetic mean image, mA, and the HLS estimate are shown in figure 7. In the test, we produced 50 instances of simulated data, each having a different noise realization, and computed the estimates. . . ."

All these simulated data and features have been developed as a part of this thesis. As another example we have the article "Least Squares Time-Series Synchronization in Image Acquisition Systems" published on "IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 25, NO. 9, SEPTEMBER 2016". In section results of this paper we have:

"The estimation methods introduced in the paper were integrated into the Unimap software , as a pre-processing step, preceding the GLS estimation. Moreover, in order to evaluate the performance, we developed a simple simulator of the PACS instrument. The simulator is based on a synthetic but realistic sky image, which is used as the truth. In order to simulate the continuous nature of the sky, this synthetic image has a higher resolution than the estimate to be produced, i.e. a lower pixel size. The pointing information is obtained from a true Herschel observation. In order to produce the simulated data vector, we perturb the pointing information by shifting each sampling point along the scan direction, by a fraction of the sampling spacing, which mimics the presence of a RPE in the data. Then, using the perturbed pointing information, we sample the synthetic sky to produce the shifted signal vector Rs" As we have seen, not only the reduction results of the paper has been really useful for the developers of this software, but also the Unimap Simulator has remained as a helpful tool for them. This Unimap Simulator software has become a companion for the Unimap for the rest of its journey towards helping astronomers understanding our world better and deeper.

References

[1]. Lorenzo Piazzo, Luca Calzolettib, Fabiana Faustinib, Michele Pestalozzic, Stefano Pezzutoc, Anna di Giorgioc and Sergio Molinari "Astronomical Infrared Imaging Case Study: the Herschel Satellite and the Unimap Software" .

[2]. Lorenzo Piazzo, Luca Calzolettib, Maria Carmela Raguso, "Least Squares Time-Series Synchronization in Image Acquisition Systems" Published in: IEEE Transactions on Image Processing (Volume: 25, Issue: 9, Sept. 2016)

[3]. Lorenzo Piazzo, Maria Carmela Raguso, Javier Gracia Carpio, "Least squares image estimation in the presence of drift and pixel noise" Published in: Signal Processing Conference (EUSIPCO), 2016 24th European

[4]. David L. Clements, "Infrared Astronomy – Seeing the Heat", CRC Press

[5]. Robert Gendler, "Lessons from the Masters", Springer Press

[6]. Chris Solomon, Toby Breckon, "Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab", Wiley Press

[7]. Lorenzo Piazzo , "UNIMAP (Lorenzo Piazzo)" presented in "Herschel PACS and SPIRE Map-Making Workshop"

[8]. Lorenzo Piazzo ,Unimap official webpage at sapienza website "http://infocom.uniroma1.it/unin

[9]. Lorenzo Piazzo ,Unimap official webpage at sapienza website "http://infocom.uniroma1.it/unin

[10]. Wikipedia Herchel webpage "https://en.wikipedia.org/wiki/Herschel_Space_Observatory"

[11]. "https://www.herschel.caltech.edu/"

[12]. "https://www.herschel.caltech.edu/"