Final Report: Image Processing

Chris Turner, 196447
chris@christurner.org.uk

Table of Contents

1. **Aims and Objectives**

1.1 Introduction

When an image is taken of the night sky it will, no matter how still the night, have an element of atmospheric distortion. This is due to the light having to pass through the atmosphere before reaching the eye or in this case a lens of a telescope.

The problem with atmospheric distortion comes from one of the most basic properties of light, refraction. The distortion originates at the boundary between the upper atmosphere and the vacuum of space. At this point the light, say from Alpha Centuri, has already travelled 4.2 light years before reaching the atmosphere and then, as happens with all electro-magnetic (EM) radiation moving through a boundary, the light "bends" slightly.

This alone wouldn't be a problem as, using Snell's Law (Eq. 1.1),

$$\frac{1}{n} = \frac{\sin\theta}{\sin\phi} \quad \text{- (1.1)}$$

the angle of refraction could be calculated and a true position of the star obtained. However, the atmosphere is made up of many layers each having different properties and it is this that causes the problem.

As the light enters each of these layers it is distorted by a very small amount and in some cases the signal is degraded. So when the light reaches the eye, the distortion is seen as "twinkling" and this would need to be removed in order to enable further study of the star.

1.2 Primary Aims

Therefore, the primary aims of this project are:

- To write a program that will take an image stored in either the JPEG[1], TIFF[2] or 24-bit Bitmap formats and allow the user to perform various operations upon them.
- To apply deconvolution filters to such an image so that a much more distinct version of the image is obtained.

1 Joint Picture Executive Group
2 Tagged Image File Format

1.3 Objectives

In order that the aims of the project can be achieved the following will need to be done in the order specified:

1. The pixel data is to be extracted from the image so that a graph can be plotted.
2. The pixel function is to be found from the data points.
3. A Fourier Transform is to be applied to the pixel function so that the point spread function can be found.
4. A function is to be found that, when applied to a pixel, will produce a much more distinct version of the image.

1.3.1 Extracting the Pixel Data

Pixel data, or values, are stored in a range of $0 \leqslant x < 256$ as there are only 8 bits that the pixel value can be stored in for each of the 3 colour channels, hence 24-bit bit map. Therefore, as the pixel value is an integer between 0 and 255 it should be fairly straightforward to extract the data from the image.

1.3.2 Finding the Pixel Function

The pixel data can only be extracted for each individual pixel. This is a problem for performing a Fourier Transform as the values will not produce a continuous function. Moreover, the graph produced will only have results for the function for the integers $(f(x) \forall x \in \mathbb{Z})$. Therefore a preliminary function is to be found that can "predict" a pixel value for the real values $(\forall x \in \mathbb{R})$. This function will be continuous and will be used to find the point spread function.

1.3.3 Fourier Transforms

When an image is taken using a CCD camera, the final image is convolved with the shake from the camera when the shutter is pressed and the point spread function. However, these can both be extracted from the pixel function using a Fourier transform (Eq. 1.2).

$$g(\phi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-it\phi} dt \quad - (1.2)$$

1.3.4 Deconvolution

The point spread function can then be used to deconvolve the image and remove the camera shake and the atmospheric distortion from it by forming a filter that can be applied to the image, thus producing a pure images

2. Background Theory

2.1 Astrophotography

2.1.1 A Brief History of Astronomical Observations

Many cultures have engaged in the discovery and conclusions drawn from astronomy. These include the Chinese, Ancient Greeks, Ancient Egyptians and the Macedonians on the campaign through Persia. All of the major events in the early part of history for these cultures arose from the historical and the mathematical observations made on the days that they occurred. For example, if the death of a major ruler occurred on the same day as a solar eclipse then, to scientists who observed it, the conclusion would be that on the day that a solar eclipse occurred a major ruler would die.

This is especially shown in the Ancient Chinese culture when two astronomers of the royal court failed to predict a solar eclipse, which showed that there was an equation in place many thousand of years ago for predicting a solar eclipse.[3]

From that time onward, man still observed the skies for scientific study as well as for religious purposes. It was because of this that the study of astrology was developed in the way we know it today.

In the middle ages, around 1450 A.D, Copernicus came up with the theory of orbits. This stated, from his sixth book, that the Earth and all the planets moved in an orbit centred at the Sun. He also suggested that the moon was on an elliptical orbit centred at the Earth.[4]

This theory was partially accepted by the church. However, they still believed that the Earth was the centre of the universe and all the planets, stars and the Sun orbited the Earth, and that as a man falls to Earth, the Earth was falling to the centre of the Universe and as it was the closest to the centre the other bodies orbited as they were trying to get closer.

The theory that Copernicus put forward was eventually proved right, although still not accepted by the religious authorities, by Galileo when he invented his telescope and made the historic observations of the moons of Jupiter and the phases of Venus he stated that the theory put forward by Copernicus was in fact correct and like the moons orbited Jupiter, the Earth and all the other planets orbited the Sun. This was not accepted by the church and Galileo was imprisoned as a heretic.

3 [1] Page 10
4 [1] Page 28

The observations of Galileo were used by Kepler when he published his first two laws in "Astronomia Nova, seu Physica Coelestis tradita commentaris de Mobibus Stellae Martis" in 1609 which state:[5]

- The Planets describe ellipses with the Sun at the focus of each ellipse.
- A line drawn from a planet to the sun sweeps over equal areas in equal times.

However, it took Kepler another nine years, due to the lack of logarithmic graph paper, to discover his third law that states:

- The square of the time period is proportional to the cube of the mean distance to the Sun (Eq. 2.1)

$$T^2 \alpha \bar{x}^3 \quad \text{- (2.1)}$$

These laws were used by Newton when he proved his theory of gravitation which stated that the size of the gravitational force between two bodies is proportional to the product of their masses and inversely proportional to the square of the distance between them (Eq. 2.2).[6]

$$F \alpha \frac{M_1 M_2}{d^2} \quad \text{-(2.2)}$$

Since then there have been many other developments in the field of astronomy. More specifically the development of photographic film and the camera in the late 19th early 20th Century. This allowed astronomers to photograph the skies, and this helped Einstein prove that gravity bends light.

5 [1] Page 42
6 [6] Page 330 - 333

2.1.2 CCD Astrophotography[7]

When the Charged Coupled Devices (CCDs) were first introduced it wasn't that clear on how much an effect they would have on society. The idea behind them is simple: A chip with a light sensitive area is fitted behind a lens, when light is detected it changes the charge on the chip and a digital representation of the picture is produced (Figure 2.1).

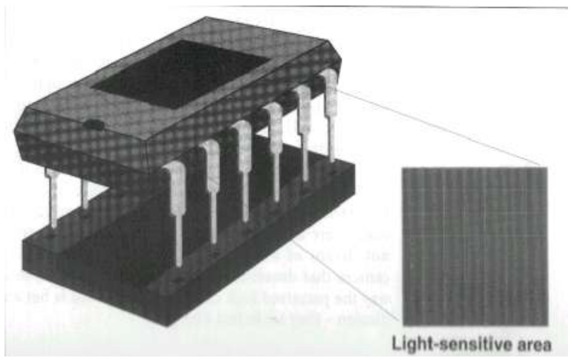


Figure 2.1: A CCD chip with the grid shown on the light sensitive area[8]

The camera has no view finder as the CCD chip is enclosed within a protective casing which is also thermal protective as a CCD camera works at an optimum level with temperatures below 0ºC. The light sensitive area of the chip is divided up into a grid and each of these light sensitive areas is what will eventually produce a picture element or pixel. It works in the following way:

The chip effectively counts the number of photons, or rather 30% of them, that fall upon the area and this is then translated into a number that is stored in the chip. When transferred to a computer the numbers are converted into a value that can be interpreted into an intensity. Table 2.1 shows the values for different bit rates.

7 [4] Chapter 1, [2] Chapter 1
8 [4] Page 2

Number of Bits	Intensity per colour channel
8	256
10	1 024
12	4 096
14	16 384
16	65 536

Table 2.1: Maximum Intensity per colour channel for various bit rates[9]

The CCD counts the number of photons that fall on a particular element of the light sensitive area of the CCD chip. This alone is not sufficient for an image to be produced. The numbers that the CCD counts must be converted from, say 10-bit integers, to an 8 bit integer as this is considered the standard for an image that is displayed on a VDU. To achieve this the following algorithm is required:

```
Algorithm Convert(Received_Value):New_Value;
        result:=MOD(Received_Value,256);
    End.
```

Algorithm 2.1: Conversion to 8 bit numbers

Algorithm 2.1 will take a value from the image, say a pixel value stored as a 10 bit integer and apply the modulo function to it to produce a value in the range $0<x<255$. The image, now in the 8-bit standard format for any graphics program can now be displayed on a computer screen and can have various functions applied to it.

Due to the necessity of astrophotgraphy needing to be performed at night has the problem of light pollution as urban areas have street lamps that will effect the final image in that there would be background noise within the image. To avoid this, with similarity to radioactivity, a reading of the background noise at the lens of the camera has to be taken. This reading must include values on all three colour channels and then this can be subtracted from the final image using algorithm 2.2.

9 [4] Page 9

```
Algorithm Remove_LP(Pixel:Array[1..3], L_P:Array[1..3]);
               for i:=1 to 3 do
                    Pixel[i]:=Pixel[i] - L_P[i];
                        end;
          End.
```

Algorithm 2.2: Removing Light Pollution

The image can now be taken from the camera and transferred to the computer and have the light pollution removed from it. There are many more image processing functions and techniques that can be applied to the image. These will be discussed in Section 2.2.

2.2 Image Processing

2.2.1 Image Processing Algorithms

Images taken using a CCD camera can now be downloaded from the camera to the computer and have various functions and techniques applied to them. One of these, as discussed in the previous section, is how to remove the light pollution from the image and this algorithm is shown above. There are many other image processing algorithms that can be applied to digital images and these will be discussed here. All algorithms discussed will be written in the Python programming language. Before various algorithms are applied to the image the image must be loaded into the memory of the computer. This is done using the code below:

```
import Image

im = Image.open('Image_File_Name')
```

When using a monochrome CCD camera, as most are, there can be noise added to the image when the image is transferred across to a computer. This can sometimes appear as a shade of yellow or red. To remove this the easiest thing to do is to re-grayscale the image before the image is deconvolved. To grayscale the image and average is taken of the intensity values of all three colour channels and then each colour channel is set to this average value in accordance with the maximum intensity rule. The code for this is shown below.

```
Def Grayscale():
     def Gray_Image(i,j):
          Pixel = im.getpixel((i,j))
          Gray = (Pixel[0]+Pixel[1]+Pixel[2])//3
          im.putpixel((i,j),(Gray,Gray,Gray))

     for i in range(0,im.size[0],1):
          for j in range(0,im.size[1],1):
               Gray_Image(i,j)
```

This function goes through the image pixel by pixel according to the coordinate i and j extracting the pixel data and performing integer division, i.e. ignoring the decimal points and rounding to the nearest integer, upon the pixel before setting the new pixel values to that result.

Another useful function that can be applied applied to images of this sort is the threshold function. This looks at each individual pixel and depending on the number of threshold values, in this case there will be two threshold values, will set the new pixel value to the new pixel value. The code for this is shown below.

```
Def Threshold(i,j):
     Pixel = im.getpixel((i,j))
     If Pixel[0] < 128 then im.putpixel((i,j),(0,0,0))
     else im.putpixel((i,j),(256,256,256))
```

With two threshold values the intensities on the colour channels can take either 0 or 256. This makes the threshold value 128, therefore, any pixel that is within 5 of this value will either appear as 0 or 255. The more threshold values that are added to the function will make it even more useful.

There are many other functions that can be applied to images, another one of particular mention is the gamma correction function that will make the displayed image look the same as the sky did when the original picture was taken. It is also useful for the printing of images. This function uses the following equation on each of the colour channels, $P' \propto P^{\gamma}$ where P' is the new pixel value and P is the old pixel value. There is a constant of proportionality but this is usually set to 1 by convention.

The main purpose of this function is to increase or decrease the brightness of the image depending on the gamma values of the hardware devices that the image will be displayed upon.

2.2.2 Image Processing Filters

Along with algorithms, there are a numerous number of filters that can be used to alter an image. Those that will be discussed are the edge detect, the box filter and the Gaussian blur.

The operations that can be performed on an image can be considered in the following way, an image processing algorithm can be compared with a mathematical function and all the operations that can be performed on a function can be performed on an algorithm whereas the filters can be considered a matrix and therefore the matrix operations can be applied to it.

-1	-1	-1
-1	8	-1
-1	-1	-1

Table 2.2: The Edge Detect Filter

The edge detect filter, as shown in table 2.2, is applied to every pixel in the image. As it moves through the image it looks for pixels that are different from its neighbours by taking negative contributions from all those surrounding it and enhancing the central one. When it finds a large difference between two pixel intensities, in this case between a star and the black void of space, this will be enhanced by the filter. This is an example of a high pass convolution filter.

1	1	1
1	1	1
1	1	1

Table 2.3: The Box Filter

The box filter, as shown in table 2.3, is an example of a low pass convolution filter. The main objective of a low pass filter is to average out all the differences between the pixels and remove any large differences between the pixel values. This is useful for removing noise from an image and the erroneous values within the image can be removed by averaging out over the surrounding pixels.

1	1	1	1	1
1	3	3	3	1
1	3	9	3	1
1	3	3	3	1
1	1	1	1	1

Table 2.4: The Gaussian Blur

Another useful blurring filter is the Gaussian blur, see table 2.4, this is called the Gaussian blur because, if the matrix were plotted as a 3D-Graph, it would have the shape of a Gaussian (Equation 2.3).

$$f(x, y)=e^{-(x^2+y^2)} \quad \text{-(2.3)}$$

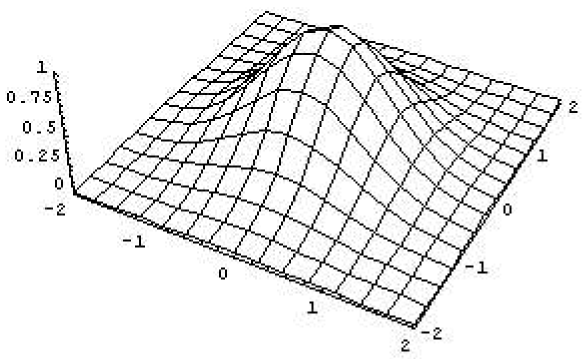


Figure 2.2: A Plot of Equation 2.3

This filter has the result that the dim pixels next to really bright pixels will take some of the brightness of the really bright pixel and adjacent pixels of the same brightness will remain the same. In this case the reverse of the Gaussian blur should be applied to the image to retrieve the intended result.

These filters cannot be applied directly to the image as each value in the matrix must be multiplied by the intensity of the pixel and then every pixel in the image must be returned to a value in the range $0 \le x \le 255$ this can be achieved using algorithm 2.1.

2.3 Mathematical Proofs

2.3.1 Fourier Transform and Fourier Series

The Fourier series can be applied to any suitably well behaved function as long as it obeys:

$$f(t+T)=f(t)$$

i.e. the function is periodic.

A suitably well behaved function is described as being single valued, continuous except for a finite number of points in each period and have a finite number of maxima and minima. The Fourier transform is given by equation 2.4.

$$f(t)=\frac{a_0}{2}+\sum_{n=1}^{\infty}[a_n\cos(n\omega t)+b_n\sin(n\omega t)] \qquad \text{- (2.4)}$$

where:

$$a_n=\frac{2}{T}\int_0^T f(t)\cos(n\omega t)dt \qquad \text{- (2.5)}$$

$$b_n=\frac{2}{T}\int_0^T f(t)\sin(n\omega t)dt \qquad \text{- (2.6)}$$

As the functions that will be used within the Fourier Series will not be well behaved it is essential that the above equations be found in a form that would work for all functions. Therefore, using complex numbers, we can rewrite 2.4 as:

$$f(t)=\sum_{m=-\infty}^{\infty}c_m\mathrm{e}^{im\omega t} \qquad \text{- (2.7)}$$

where:

$$c_m=\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\mathrm{e}^{-im\omega t}dt \qquad \text{- (2.8)}$$

We can now get a non-periodic function by taking the limit of $T\rightarrow\infty$ we can define a new variable $\phi=m\omega$ when we take that limit.

Therefore we can now define the Fourier transform for any function:

$$g(\phi)=\frac{1}{\sqrt{2\pi}}=\int_{-\infty}^{\infty} f(t)\mathrm{e}^{-it\phi}dt \qquad \text{-(2.9)}$$

Where the inverse of this is:

$$f(t)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} g(\phi)\mathrm{e}^{it\phi}d\phi \qquad \text{-(2.10)}$$

3. Resources used in the Project

3.1 Programming Languages

3.1.1 Python

When considering the best programming language to take an image file and open it to perform operations upon it. The most obvious choice would therefore be C or VC++. However, this language is a low level language and therefore would take time to perfect. The next choice would be a functional language, for example Haskell.

A functional language has the benefit over non-functional languages by not needing the variables to be predefined. For example, in C the variables used in a procedure will need to be defined at the start as memory within the computer will have to set aside for the information contained within these variables. In a functional language, however, the memory is set aside when the function is called. This has the advantage that the memory will be released back to the system immediately after the function has produced a result whereas C has a command which will release this memory back to the system.

Python, being an interpreted language, uses the most useful parts from both these types of languages in that it doesn't need the variables to be pre-defined in the way that C does and that it uses the procedure and function definitions that Haskell doesn't use. The other advantage that Python has over C is that, as a high level language, it does not require the memory allocations to be specified within the user code as these are added by the interpreter when the program is run.

Another advantage of python is that the language has libraries that have the required code for opening say a JPEG image already encoded within the interpreter. This is useful in that the extraction of pixel data and the application of pixel functions and matrices can be encoded very quickly. The down side to this is that the computational time is almost doubled as the calls to the functions that perform the operations have to be interpreted and then the code looked up and finally the operation applied. However, this is the only downside to this language that I can find.

3.1.2 Borland Delphi

Borland Delphi is a form of object pascal that has a nice user interface, GUI, and this can used to design a program that will display images in the formats that we need have on the screen using the inbuilt TGraphic type. However, like C, this language requires the variables to be pre-defined and operations cannot be performed on the Tgraphic format as easily as they can in Python.

Therefore, the program language that will be used to write the software will be Python.

3.1.3 Wolfram Mathematica

Mathematica has only one true disadvantage within it in that it is very resource dependant in that it will "eat" all the resources that are available when performing a calculation. However, with the power that the software has this isn't really a problem compared with the operations it can perform.

Mathematica is a form of functional language in that the main operations it can perform are list based and it will only operate on lists and lists of lists. This has the advantage of making it very easy to use as well as very powerful. The graphs used and shown within this report are all made using Mathematica and thus shows the potential of this language.

Finally, again due to the fact that the language is so powerful it can interpolate a function of an image of a star for example and then perform a Fourier Transform on that function. This will be used and shown later in this report in more detail.

3.2 Books

All books used as references in this project can be found in the Bibliography at the end of this report. The main books used in the writing of the code for this project were:

- "Python in a Nutshell" - A. MARTELLI
- "Delphi 5 Developer's Guide" - S. TEIXEIRA and X. PACHECO

3.3 Other Resources

3.3.1 Lecture Notes

In this project I have made use of the lecture notes from the following courses;

Department of Computer Science

Computer Graphics, CS217, Dr. M. JONES

Scientific Modelling and Simulation, CS329, Prof. M. WEBSTER

Department of Physics

Advanced Techniques of Theoretical Physics, PH304, Dr. N. DOREY

3.3.2 Internet Resources

As Python is a language that is freely available, i.e. Open Source, there are many freely available web resources. These include:

- Python, http://www.python.org
- Pythonware – for the Python Image Library, http://www.pythonware.com
- The Python Cookbook – for examples of code for image processing, http://aspn.activestate.com/ASPN/Python/Cookbook/

4. Description of Project Work

As discussed, the main aim of this project is to extract the point spread function from an image and then to write either a deconvolution filter or function that will be able to restore each star in the image back to a point source of light. In order that the ability for this to be done the following was done.

4.1 Programs

In the second year computer graphics course, the coursework for this course is to write a program that will perform simple operations upon an image in the 24 bit bitmap format. Therefore, when starting to set up a program that would perform the necessary operations upon an image the easiest thing to do in the outset would be to take this program and use this as the basis of this project.

The following program was written to test various theories and then the program was re-written to open files in JPEG format finally the program was re-written in Python.

4.1.1 Initial Program: CCD_Process

The initial program was written in Delphi as, at that particular time, the images available were in the bitmap format. Therefore as the basics of the program were already written in Delphi the easiest thing to do at that time would be to continue with that program, primarily to get the main functions and graphic operations written and tested on an image before outputting them into python and applying them on a larger image.

The functions included with this program will be discussed below, with the pseudo code for the relevant function with the explanation.

Main Program

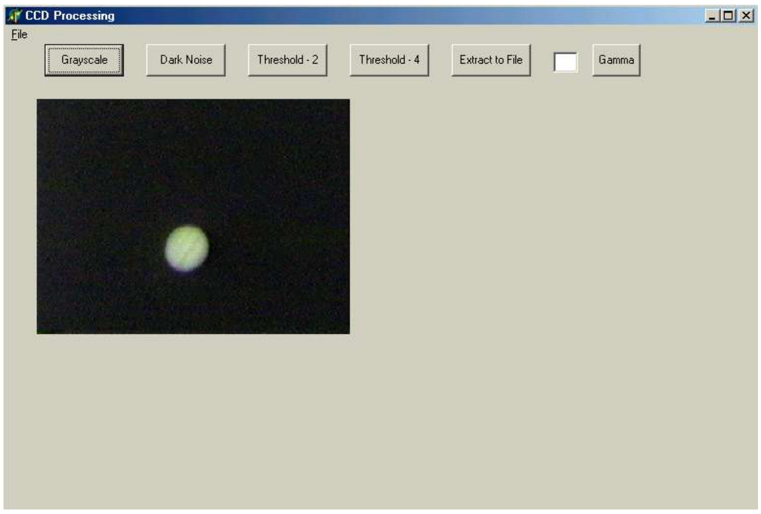


Figure 4.1: Screen Shot of CCD Process

This program was written in an object orientated way that allowed the graphical user interface (GUI) to be designed in such a way that made it firstly easier that specifying the coordinates of each item in the window and secondly, made it simple to add new buttons to activate more functions.

When deciding on the functions that should be included within the program it was clear that as the bitmap images were taken in monochrome that there was noise in the line between the camera and the computer or video recorder that accounts for the fact that the image has a yellow tint to it. In order to remove this the image must be taken back to the grayscale image and then the algorithm for removing light pollution, described above, will be used to remove the light pollution for each pixel.

The next functions that were added were the threshold functions, this provided the capability to go on and write the dark-noise function. These functions went through the image, pixel by pixel, and for the threshold functions, if the value of the pixel was in the range $a \le x < b$ then the pixel value would be set to a. The dark noise function works on a more liberal basis in that it will only reduce those pixels that are below a particular value and leave all the rest alone.

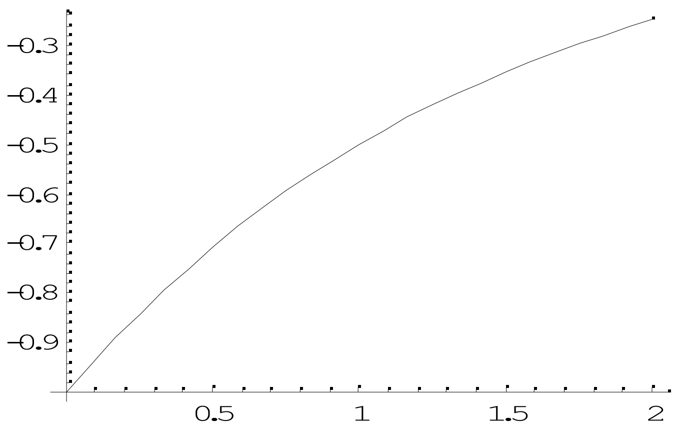


Figure 4.2: The Gamma Curve

The final function to be added was the gamma correction function, this allows the intensity of the whole image to be adjusted according to the gamma curve above.

Gray scale

The code for this function is as shown:

```
procedure grayscale(Pixel:Array[R,G,B])

begin
     gray := trunc((Pixel[R] + Pixel[G] + Pixel[B]) / 3)
     Pixel[R] := gray
     Pixel[G] := gray
     Pixel[B] := gray
End.
```

For an image to be grey-scaled, so called as this colour map only uses 256 shades of grey with black as 0 and white as 255. A grey image has the value of each colour channel of a pixel set to the same value. This value is calculated from the "average" of the initial values using integer division as each colour channel can only hold the integer values for the intensity.

Threshold - 2, Threshold – 3

```
Procedure Threshold_2(Pixel:Array[R,G,B])

begin
     if Pixel[R] < 128 then do
          Pixel[R] := 0
          Pixel[G] := 0
          Pixel[B] := 0
          end;
     else do
          Pixel[R] := 255
          Pixel[G] := 255
          Pixel[B] := 255
          end;
End.
```

The threshold – 2 function moves through the image pixel by pixel and turns each pixel either completely black or completely white. For this example it has the benefit of showing where the major features are within the image and where possible noise has eliminated some of the not so prominent features.

```
Procedure Threshold_3(Pixel:Array[R,G,B])

begin
     if Pixel[R] < 64 then do
          Pixel[R] := 0
          Pixel[G] := 0
          Pixel[B] := 0
          end;

     else if Pixel[R] < 192 then do
               Pixel[R] := 128
               Pixel[G] := 128
               Pixel[B] := 128
               end;

          else do
               Pixel[R] := 255
               Pixel[G] := 255
               Pixel[B] := 255
               end;
End.
```

Threshold – 3 has the added benefit of showing the prominent features, e.g. planets, with more detail than the threshold – 2, it works in the same way but with three "frequency" bands instead of 2. This means that the pixels with intensity values around the threshold values will loose less of their intensity than would otherwise. However, those with intensity values of more than 128 will still be given the value of 255 when the function is applied.

Dark Noise

To compensate for the fact that those pixels with intensities greater than 128 would be given the maximum value when the threshold function was run gave rise to the dark noise filter. This filter worked on the same basis as the threshold values but works with a pixel value that can be changes and will leave the important features of the image virtually untouched.

```
Procedure Dark_Noise(Pixel:Array[R,G,B])

begin
     if Pixel[R] <= 84 then do
          Pixel[R] := 0
          Pixel[G] := 0
          Pixel[B] := 0
          end;
End.
```

This filters out all pixels that have an intensity of less than 84 and can show other objects in the image when this is run. For example, if there was a star located in the following section of an image it would not be visible due to the noise surrounding it.

84	78	83	81	83
83	94	93	95	84
81	89	100	94	82
82	91	92	93	81
83	84	81	79	80

Once this filter has been run, however, the image looks like:

0	0	0	0	0
0	94	93	95	0
0	89	100	94	0
0	91	92	93	0
0	0	0	0	0

Now that the noise has been removed, it is quite clear where the star lies. Both threshold functions and the dark noise filter are best run on grayscale images as the running of these filters on colour images could produce tinted images any of the colour channels.

Gamma Correction

When an image is displayed on a computer screen, especially when the image has been captured using a video capture card, there can be a problem in that the image may appear too dark or too light. This can be adjusted by using the gamma correction filter which changes the perceived brightness of the image and normalises the image so that it is displayed more realistically.

```
Procedure Gamma_Correction(Pixel:Array[R,G,B],
                           Gamma:Real);
begin
     Pixel[R] := Round(Pixel[R]^Gamma)
     Pixel[G] := Round(Pixel[G]^Gamma)
     Pixel[B] := Round(Pixel[B]^Gamma)
End.
```

This function can be applied to both grayscale and colour images as the gamma correction is the constant of proportionality in the equation:

$$\log_{10}(P'_x)=\gamma\log_{10}(P_x) \quad \text{where} \quad x\in\{R,G,B\}$$

4.1.2 Python Program

After the partial success with CCD_Process and its ability to perform the above operations on bitmap images, it became clear that the next stage would be to apply the same operations on images in the JPEG format. This however, is harder due to the compression on a JPEG image.

A JPEG image doesn't store the individual pixel data like a bitmap does and therefore it cannot use the scan line function that is available in Delphi. The image is stored as a set of initial values and then an interpolation function for each of the colour channels. This will become clearer as the data is analysed later.

However, special filters need to be applied for the JPEG image to be successfully loaded, these are available in Delphi but it is easier, faster and simpler to write a second program in python that can apply the above functions, and others, to images in JPEG, TIFF and Bitmap format.

As explained above, python is language that the developers have effectively said; "I need to do x therefore I will right a library to do it." This is effectively what has happened with image processing. The Python Image Library (PIL) already contains the code for loading and displaying images in any standardised format, by standardised I mean any that can be opened using Adobe Photoshop excluding the Photoshop Projects.

Python Program Code

The libraries are loaded in a similar way to C using the import command

```
from Tkinter import *
import Tkinter
import Image, ImageTk, sys, math, os, getopt
```

The GUI is set up by giving the program what needs to be displayed and then the interpreter will display it in the most optimal way

```
# Setting Up the GUI
root = Tk()
minx=Tkinter.StringVar()
maxx=Tkinter.StringVar()
miny=Tkinter.StringVar()
maxy=Tkinter.StringVar()
```

The image is loaded from the file name specified when then program is run from the console of an OS. A second buffer is created for a copy of the image that is 5 times smaller than the original however, all operations are performed on the full size image.

```
# Initialising the Program
im = Image.open(sys.argv[1])
img = im.resize((im.size[0]//5,im.size[1]//5))

# Dispalying the Image
img_canvas=Canvas(root, height=img.size[1],
                        width=img.size[0])
img_canvas.pack(side=LEFT, fill=BOTH, expand=1)
photo=ImageTk.PhotoImage(img)
item = img_canvas.create_image
                 (10,10,anchor=NW,image=photo)
```

As with the initial program, functions are performed pixel by pixel. In order to produce a function that can be Fourier Transformed the pixel data must first be extracted from the image

```
def Extract():
     global minx
     global maxx
     global miny
     global maxy
     minx=int(minx.get())
     maxx=int(maxx.get())
     miny=int(miny.get())
     maxy=int(maxy.get())
     Tkinter.Label(text='Extracted Pixel Data').pack()

     Red_Data=open('Red.csv',mode='w')
     Green_Data=open('Green.csv',mode='w')
     Blue_Data=open('Blue.csv',mode='w')
```

Exporting the pixel data to a file requires the files to be created using the open command. This then produces a list of numbers in one line.

```
     def Export_Pixel(i,j):
          Pixel=im.getpixel((i,j))
          Red=str(Pixel[0]),','
          Green=str(Pixel[1]),','
          Blue=str(Pixel[2]),','
          Red_Data.writelines(Red)
          Green_Data.writelines(Green)
          Blue_Data.writelines(Blue)
          return NONE

     for i in range(minx,maxx,1):
          for j in range(miny,maxy,1):
               Export_Pixel(i,j)

     Red_Data.close()
     Green_Data.close()
     Blue_Data.close()
     return NONE
```

Again, to test the image processing power of the program a test function was written. The grayscale function produces and optically visible change to the image.

```
def Grayscale():
     Tkinter.Label(text='Grayscaled Image').pack()
     global photo, item, img, im

     def Gray_Image(i,j):
          Pixel=im.getpixel((i,j))
          Gray = (Pixel[0]+Pixel[1]+Pixel[2])//3
          im.putpixel((i,j),(Gray,Gray,Gray))

     for i in range(0,im.size[0],1):
          for j in range(0,im.size[1],1):
               Gray_Image(i,j)

     img = im.resize((im.size[0]//5,im.size[1]//5))
     photo=ImageTk.PhotoImage(img)
     item = img_canvas.create_image
                     (10,10,anchor=NW,image=photo)
```

As the extraction of the pixel data outputs the data to a file all on one line there must be some post production done on the data to turn it into a usable form, this being a list of lists that can then be processed using mathematica. To accomplish this a second program was written in delphi for formatting the data.

4.2 Data Formatting

The following program was written to take the data into a string (datastr) and then, using a for loop would output the data as a list of lists that could be processed. There are some bug in this in that the there will be the addition of a couple of curly brackets at the start and end to enable the image processing to take place.

```
procedure write_to_file();
var imax:integer;
    i,j :integer;
begin
  open_file(outputfile);
  rewrite(outputfile);
  imax:=length(datastr);
  for i:=0 to imax do begin
     if j=30 then begin
          writeln(outputfile,datastr[i],'}',',');
          j:=0;
          write(outputfile,'{');
          end;
     if datastr[i]=',' then begin
          write(outputfile,datastr[i]);
          j:=j+1
          end
     else write(outputfile,datastr[i])
          end;
  closefile(outputfile);
end;
```

5. Results and Analysis

The most obvious feature in the image is the star Lyra, however this has saturated the camera therefore a fainter star was taken. The section of the image used is shown below.

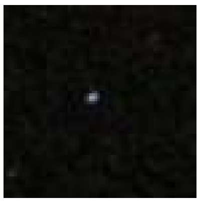


Figure 5.1: The chosen section of Image 1

5.1 Data Processed Results

Red Colour Channel

$$\begin{pmatrix}
10 & 12 & 11 & 11 & 9 & 9 & 11 & 11 & 12 & 8 & 5 & 4 & 7 & 9 & 9 & 12 & 16 & 15 & 13 & 9 & 8 & 10 & 11 & 14 & 15 & 18 & 14 & 14 & 14 & 14 & 1 \\
12 & 10 & 11 & 13 & 10 & 9 & 10 & 10 & 11 & 10 & 6 & 6 & 11 & 14 & 14 & 15 & 18 & 16 & 13 & 10 & 10 & 13 & 14 & 18 & 19 & 19 & 14 & 13 & 14 & 14 & 9 \\
9 & 9 & 6 & 8 & 9 & 8 & 8 & 11 & 10 & 9 & 7 & 8 & 12 & 17 & 15 & 10 & 16 & 19 & 14 & 9 & 8 & 13 & 19 & 21 & 20 & 15 & 14 & 11 & 10 & 12 & 9 \\
9 & 6 & 3 & 7 & 9 & 9 & 10 & 12 & 10 & 14 & 12 & 8 & 10 & 9 & 8 & 11 & 10 & 14 & 10 & 8 & 10 & 16 & 19 & 17 & 12 & 10 & 11 & 11 & 11 & 12 & 9 \\
9 & 7 & 4 & 6 & 9 & 8 & 11 & 13 & 9 & 6 & 10 & 8 & 12 & 9 & 10 & 17 & 10 & 14 & 8 & 3 & 4 & 6 & 8 & 10 & 8 & 4 & 8 & 11 & 12 & 14 & 9 \\
9 & 8 & 5 & 6 & 7 & 8 & 12 & 14 & 11 & 5 & 7 & 10 & 10 & 8 & 8 & 10 & 10 & 17 & 9 & 3 & 1 & 2 & 4 & 5 & 5 & 12 & 15 & 15 & 11 & 10 & 9 \\
9 & 9 & 6 & 6 & 9 & 9 & 15 & 19 & 16 & 6 & 5 & 10 & 7 & 8 & 11 & 6 & 14 & 12 & 12 & 11 & 11 & 11 & 9 & 6 & 4 & 10 & 13 & 11 & 7 & 6 & 1 \\
11 & 10 & 8 & 10 & 11 & 11 & 17 & 22 & 21 & 10 & 8 & 16 & 15 & 12 & 12 & 6 & 7 & 8 & 15 & 17 & 18 & 17 & 16 & 13 & 8 & 4 & 8 & 8 & 7 & 9 & 1 \\
13 & 12 & 12 & 15 & 13 & 11 & 16 & 20 & 18 & 12 & 17 & 19 & 23 & 14 & 12 & 22 & 2 & 5 & 15 & 20 & 16 & 13 & 17 & 16 & 12 & 10 & 12 & 10 & 9 & 11 & 1 \\
13 & 12 & 14 & 17 & 13 & 10 & 12 & 15 & 12 & 12 & 21 & 16 & 29 & 19 & 29 & 67 & 33 & 0 & 13 & 20 & 15 & 11 & 14 & 14 & 9 & 10 & 11 & 6 & 5 & 9 & 1 \\
12 & 9 & 16 & 18 & 15 & 13 & 11 & 11 & 12 & 14 & 19 & 23 & 25 & 17 & 82 & 145 & 106 & 37 & 4 & 10 & 16 & 11 & 6 & 12 & 14 & 15 & 8 & 9 & 11 & 10 & 1 \\
10 & 12 & 19 & 21 & 16 & 11 & 10 & 11 & 14 & 17 & 8 & 21 & 19 & 84 & 176 & 207 & 171 & 128 & 40 & 11 & 13 & 16 & 15 & 15 & 11 & 16 & 11 & 13 & 14 & 12 & 1 \\
19 & 13 & 19 & 21 & 14 & 10 & 9 & 11 & 16 & 9 & 20 & 15 & 18 & 98 & 197 & 200 & 217 & 194 & 65 & 14 & 17 & 17 & 14 & 14 & 17 & 16 & 13 & 14 & 14 & 12 & 1 \\
15 & 12 & 16 & 17 & 11 & 9 & 9 & 11 & 14 & 10 & 11 & 16 & 24 & 102 & 192 & 220 & 215 & 194 & 72 & 26 & 26 & 26 & 20 & 13 & 17 & 14 & 12 & 11 & 11 & 8 & 9 \\
9 & 11 & 12 & 12 & 11 & 10 & 10 & 10 & 10 & 5 & 12 & 13 & 39 & 103 & 151 & 180 & 194 & 159 & 82 & 55 & 42 & 31 & 14 & 8 & 19 & 10 & 11 & 10 & 10 & 8 & 1 \\
16 & 15 & 13 & 13 & 13 & 11 & 11 & 10 & 7 & 5 & 10 & 16 & 29 & 42 & 81 & 97 & 107 & 119 & 76 & 53 & 38 & 31 & 17 & 9 & 11 & 7 & 10 & 11 & 13 & 13 & 1 \\
13 & 15 & 12 & 12 & 14 & 11 & 10 & 11 & 8 & 8 & 6 & 15 & 9 & 28 & 29 & 40 & 61 & 68 & 51 & 42 & 30 & 22 & 5 & 8 & 8 & 11 & 12 & 14 & 14 & 17 & 1 \\
10 & 13 & 11 & 12 & 13 & 9 & 9 & 11 & 10 & 10 & 14 & 14 & 23 & 21 & 27 & 35 & 34 & 39 & 32 & 31 & 30 & 23 & 6 & 13 & 10 & 15 & 15 & 14 & 13 & 16 & 1 \\
10 & 12 & 13 & 13 & 13 & 10 & 12 & 14 & 13 & 10 & 10 & 14 & 19 & 19 & 21 & 24 & 22 & 18 & 22 & 25 & 23 & 17 & 15 & 16 & 14 & 17 & 18 & 19 & 18 & 18 & 1 \\
12 & 12 & 12 & 12 & 12 & 10 & 13 & 15 & 14 & 12 & 13 & 13 & 14 & 13 & 11 & 10 & 9 & 14 & 14 & 14 & 14 & 13 & 14 & 15 & 14 & 19 & 19 & 18 & 16 & 16 & 1 \\
13 & 10 & 8 & 9 & 9 & 11 & 14 & 14 & 13 & 9 & 13 & 11 & 12 & 12 & 11 & 8 & 10 & 14 & 11 & 10 & 11 & 12 & 11 & 12 & 13 & 19 & 19 & 17 & 14 & 12 & 1 \\
12 & 7 & 5 & 7 & 10 & 14 & 15 & 14 & 13 & 5 & 11 & 11 & 14 & 17 & 16 & 12 & 17 & 17 & 13 & 11 & 13 & 16 & 14 & 13 & 15 & 20 & 19 & 16 & 13 & 10 & 1 \\
15 & 9 & 6 & 9 & 14 & 15 & 14 & 12 & 10 & 7 & 13 & 12 & 13 & 17 & 12 & 7 & 12 & 13 & 11 & 7 & 10 & 14 & 10 & 7 & 10 & 18 & 18 & 14 & 11 & 9 & 1 \\
18 & 13 & 9 & 13 & 18 & 17 & 13 & 10 & 10 & 10 & 13 & 13 & 14 & 14 & 9 & 5 & 8 & 10 & 10 & 7 & 9 & 13 & 10 & 7 & 9 & 17 & 16 & 13 & 10 & 8 & 1 \\
18 & 12 & 9 & 12 & 17 & 16 & 11 & 9 & 11 & 12 & 11 & 13 & 14 & 11 & 8 & 9 & 11 & 13 & 15 & 11 & 10 & 13 & 12 & 8 & 10 & 14 & 13 & 11 & 7 & 6 & 1 \\
15 & 9 & 6 & 9 & 14 & 13 & 9 & 9 & 12 & 14 & 10 & 12 & 13 & 8 & 7 & 11 & 12 & 17 & 20 & 14 & 10 & 13 & 11 & 8 & 10 & 14 & 13 & 10 & 6 & 5 & 1 \\
13 & 7 & 16 & 17 & 10 & 11 & 10 & 9 & 12 & 18 & 14 & 12 & 11 & 12 & 13 & 14 & 15 & 21 & 19 & 15 & 12 & 9 & 9 & 9 & 10 & 12 & 11 & 11 & 9 & 7 & 1 \\
10 & 9 & 15 & 15 & 10 & 11 & 10 & 12 & 15 & 20 & 16 & 13 & 10 & 11 & 13 & 13 & 14 & 15 & 13 & 11 & 12 & 11 & 9 & 9 & 10 & 13 & 12 & 12 & 11 & 10 & 8 \\
8 & 10 & 13 & 14 & 12 & 14 & 13 & 12 & 18 & 19 & 18 & 13 & 9 & 9 & 11 & 13 & 13 & 12 & 9 & 9 & 12 & 13 & 11 & 10 & 11 & 13 & 12 & 11 & 11 & 11 & 8
\end{pmatrix}$$

Green Colour Channel

10	8	10	9	9	9	9	10	8	8	5	6	7	9	7	8	10	11	9	8	8	10	11	12	13	13	10	10	12	12	1
11	9	11	11	10	9	8	9	7	9	4	6	11	14	12	11	12	10	9	8	10	11	12	14	15	13	10	9	10	10	1
12	11	8	10	11	9	8	10	9	5	3	6	10	15	13	8	11	8	8	7	8	11	14	15	16	14	13	10	11	13	1
11	10	8	9	11	10	10	11	8	11	8	6	8	7	7	10	5	5	6	8	12	14	14	13	11	10	11	11	12	13	1
10	9	6	8	10	9	10	12	8	11	12	10	14	10	9	16	8	14	11	9	10	11	13	12	13	4	8	11	14	15	8
8	8	5	7	8	9	11	13	10	12	12	12	10	7	7	9	8	20	15	9	7	7	9	11	15	12	15	15	13	11	8
8	9	6	7	8	8	14	18	15	18	11	12	5	4	7	4	13	17	15	14	12	12	11	11	11	10	13	13	9	8	7
7	9	7	9	10	10	16	21	20	17	13	14	9	6	8	6	8	14	17	18	14	13	15	15	13	4	8	10	9	11	7
7	8	8	11	12	10	15	19	17	14	15	13	15	6	13	28	12	17	21	18	10	8	13	17	14	10	12	12	11	14	7
7	8	10	13	12	9	11	14	11	8	15	7	19	15	35	80	53	16	20	19	9	5	10	13	9	10	11	8	7	12	6
6	3	12	14	12	12	10	10	12	8	13	13	20	20	97	173	139	61	17	12	10	7	5	14	14	14	8	10	14	13	4
4	6	15	17	13	10	9	10	14	11	2	12	16	92	197	241	211	151	50	7	5	11	14	15	9	15	11	14	17	15	1
15	9	15	17	13	9	8	10	15	5	16	13	23	110	223	237	255	205	65	3	1	8	13	13	11	15	12	15	15	13	1
11	8	12	13	10	8	8	10	13	10	11	16	29	115	215	254	251	194	60	2	1	12	14	8	7	13	11	12	12	9	7
7	10	11	11	10	9	9	9	9	8	14	17	44	112	166	201	219	144	56	20	12	13	8	3	9	6	7	9	9	9	1
14	14	12	12	12	10	10	9	6	10	12	17	32	45	87	104	116	97	42	13	4	13	11	3	0	3	6	10	12	14	1
13	15	12	12	13	10	9	10	6	10	8	15	7	26	26	37	58	46	21	5	3	11	7	10	5	5	8	10	13	16	1
10	13	11	12	12	8	8	10	8	12	14	12	18	16	20	25	26	21	8	4	10	15	8	15	7	9	11	10	12	15	1
10	14	13	12	9	6	8	10	9	11	9	13	15	14	18	22	19	8	10	11	12	12	13	14	12	12	13	14	15	15	1
13	14	12	11	8	6	9	11	10	13	12	12	10	8	9	9	8	8	8	9	9	8	9	11	10	14	14	13	13	13	1
14	12	8	8	8	7	10	13	12	10	12	10	8	10	9	8	9	13	10	9	11	12	11	13	14	15	15	13	10	11	1
13	9	5	6	9	10	11	13	12	6	10	10	10	15	14	12	16	17	13	11	13	16	14	14	16	16	15	12	9	9	1
16	10	7	8	13	14	13	12	10	6	12	11	12	16	11	7	12	13	11	9	12	17	15	12	16	14	14	13	10	8	1
19	14	10	12	17	16	12	10	10	9	12	12	13	13	8	5	8	11	11	9	11	16	15	12	15	13	12	12	9	7	1
19	13	8	11	16	15	10	9	11	11	10	12	13	12	9	10	13	14	16	12	11	16	15	13	15	13	12	9	7	6	1
16	10	5	8	13	12	8	9	12	13	9	11	12	9	8	12	14	18	21	15	11	14	13	11	13	13	12	8	6	5	1
14	6	15	13	6	7	9	9	12	14	13	11	12	13	14	17	18	20	18	14	11	10	10	11	12	11	10	10	9	7	1
11	8	14	11	6	7	9	11	15	16	15	12	11	12	14	16	17	14	12	10	11	10	9	10	11	12	11	11	11	10	9
9	9	12	10	8	8	9	11	17	15	14	12	10	11	14	16	16	11	8	8	10	11	9	9	10	12	11	10	11	11	9

Blue Colour Channel

2	5	8	10	9	9	10	8	5	8	5	5	7	9	8	9	12	10	8	6	6	8	11	13	14	10	9	9	13	13	6
6	7	9	12	10	9	9	7	6	7	5	6	11	14	13	12	16	12	10	9	10	12	13	15	14	13	9	8	11	11	5
5	8	5	7	6	4	6	8	7	2	2	9	13	16	14	9	15	16	12	8	8	12	18	15	13	12	11	8	6	8	6
6	9	4	6	6	5	8	9	9	6	7	9	11	8	5	8	9	10	7	6	11	15	18	12	6	10	9	9	7	8	5
5	6	3	3	5	4	6	10	6	5	9	9	13	5	5	12	9	6	2	0	0	7	9	9	7	4	6	9	9	10	4
4	6	3	2	3	4	7	11	8	5	8	11	10	5	3	5	9	9	3	0	0	1	5	7	7	12	13	13	8	6	6
6	7	4	2	4	4	10	16	13	8	7	11	6	3	6	7	19	10	6	5	4	7	6	5	4	10	11	8	4	3	6
6	7	5	5	6	6	12	19	18	10	9	15	11	6	7	8	13	10	14	13	11	10	11	10	7	4	6	5	4	6	7
7	7	5	8	7	5	11	15	15	9	16	15	13	3	7	24	11	17	21	21	10	5	10	12	11	8	10	7	6	7	7
7	7	7	10	7	4	7	10	9	9	17	10	17	6	23	70	44	20	28	27	13	5	7	9	9	8	9	3	2	5	8
8	3	9	11	7	7	6	6	10	10	17	14	14	3	76	151	120	73	33	27	20	6	1	11	14	12	6	5	7	6	6
6	6	12	14	8	5	5	6	12	15	6	13	11	77	180	225	200	182	85	40	26	17	12	13	12	13	9	9	10	8	1
16	8	14	14	9	5	4	8	13	4	15	14	19	106	220	243	255	255	129	59	40	25	11	9	11	13	10	10	9	7	1
12	7	11	10	6	4	4	8	11	8	9	18	32	121	229	255	255	255	146	78	56	37	16	5	8	11	9	7	6	3	1
10	8	9	7	6	5	5	7	7	1	11	18	50	129	195	248	255	245	157	110	72	39	12	0	10	5	4	5	4	3	1
17	12	10	8	8	6	6	7	4	4	9	21	39	64	121	158	183	195	139	101	64	37	13	3	4	2	3	6	7	8	1
15	15	10	10	9	6	7	8	7	5	7	15	12	39	53	82	113	119	95	72	46	27	4	7	12	5	5	7	8	11	1
10	13	9	10	8	4	6	8	9	9	14	13	22	23	38	59	67	71	60	49	38	26	5	14	14	9	8	7	7	10	8
8	11	11	10	8	3	5	7	6	6	7	11	16	20	27	36	36	33	34	34	28	18	14	15	15	8	9	10	10	10	8
8	11	10	9	7	3	6	8	7	8	10	10	11	12	14	15	14	20	18	16	15	12	13	12	11	10	10	9	8	8	9
9	9	6	6	6	4	7	9	8	5	10	8	9	11	12	10	14	19	16	14	13	12	9	8	9	12	12	10	7	7	8
8	6	3	4	7	7	8	9	8	1	8	8	11	16	17	14	21	19	15	11	13	14	12	9	11	13	12	9	6	5	1
11	5	2	4	9	12	11	10	8	4	10	9	10	14	9	7	12	11	9	6	7	10	8	5	6	13	13	11	8	6	1
14	9	5	8	13	14	10	8	8	7	10	10	11	11	6	5	8	6	6	4	6	9	8	5	5	12	11	10	7	5	1
14	8	4	7	12	13	8	7	9	9	8	10	11	7	4	5	8	6	10	6	5	9	8	6	8	11	10	10	7	6	1
11	5	1	4	9	10	6	7	10	11	7	9	10	4	3	7	9	10	15	9	6	9	8	4	6	11	10	9	6	5	9
9	2	11	10	3	6	7	7	10	13	11	9	7	8	8	10	11	15	13	10	7	5	5	6	7	9	8	8	9	7	6
6	4	10	8	3	6	7	9	13	15	13	10	6	7	8	9	10	9	7	6	9	8	7	5	6	10	9	9	11	10	4
4	5	8	7	5	8	8	9	15	14	13	10	5	6	7	9	9	6	4	6	11	12	10	7	8	8	7	8	9	9	4

It is not that clear from these matrices what the results actually look like therefore, the following graphs have been plotted to show the intensities of each pixel in both a 2D and 3D format.

5.2 Graphs of the Matrices

Red Colour Channel

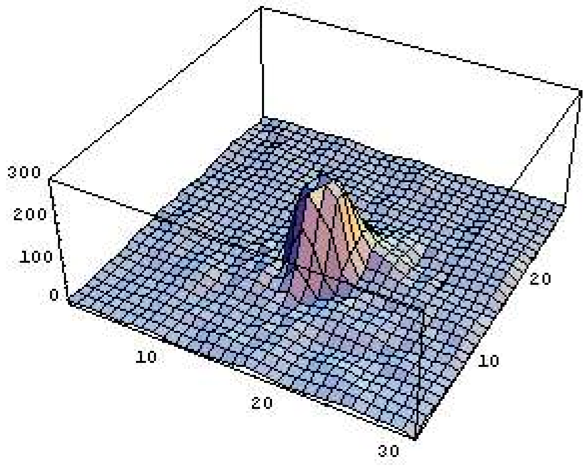


Figure 5.2: 3D Plot of the Red Colour Channel

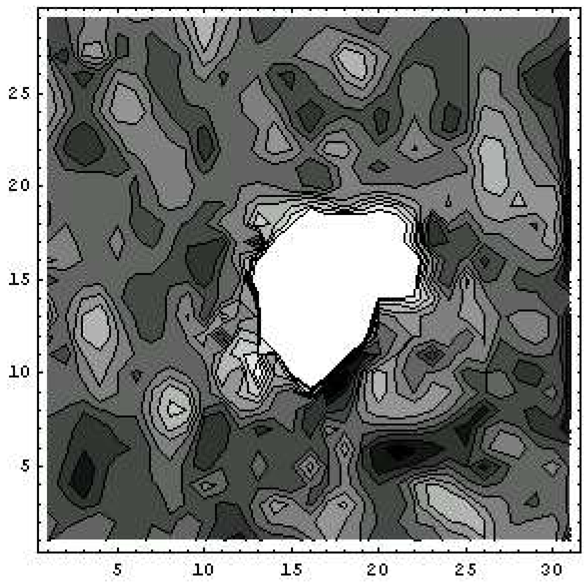


Figure 5.3: Contour Plot of the Red Colour Channel Matrix

Green Colour Channel

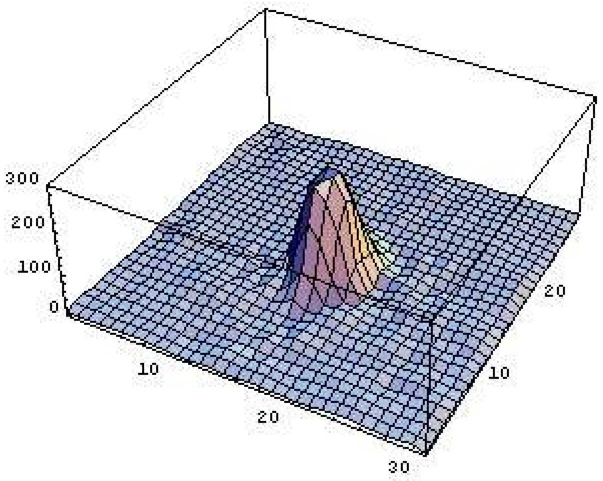


Figure 5.4: 3D Plot of the Green Colour Channel

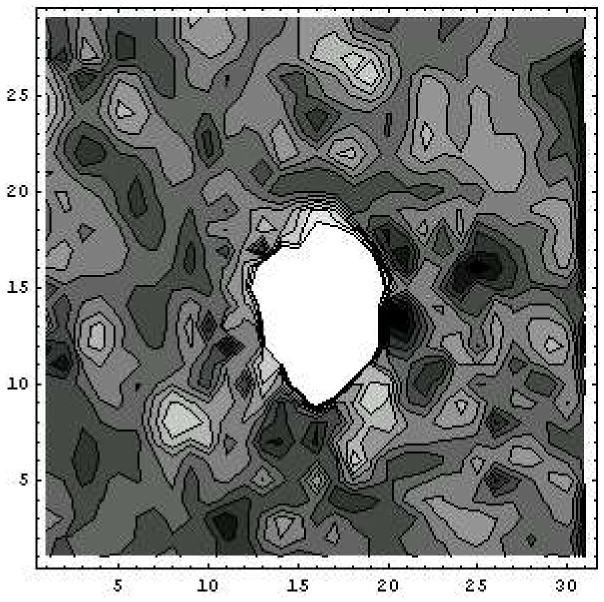


Figure 5.5: Contour Plot of the Green Colour Channel

Blue Colour Channel

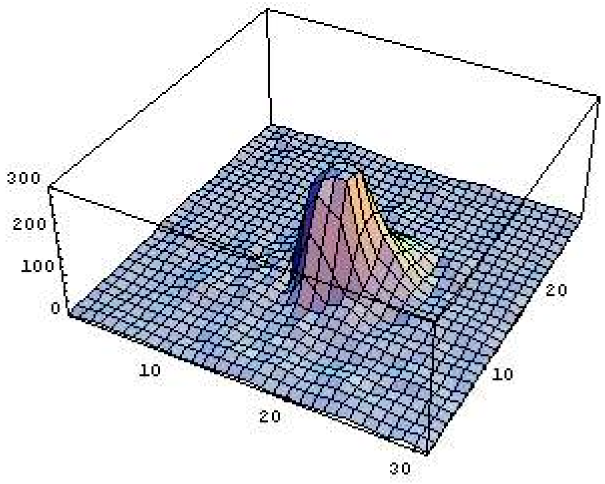
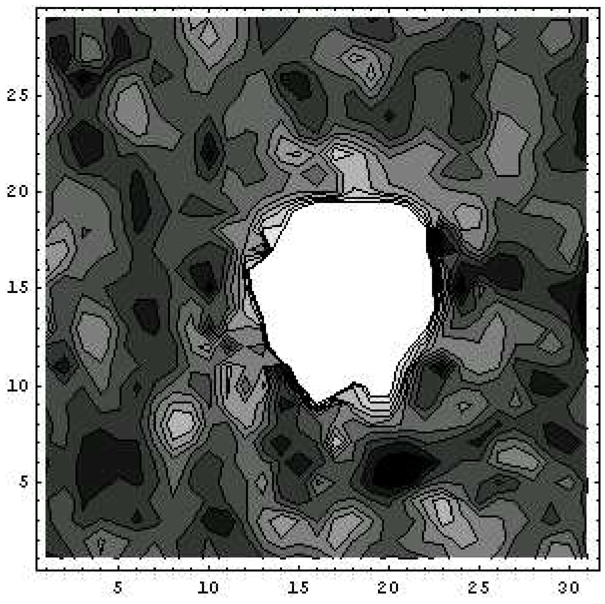


Figure 5.6: 3D Plot of the Blue Colour Channel



5.7: Contour Plot of the Blue Colour Channel

5.3 Analysis and Interpretation of Results

As with all results, these results will have to be analysed with reference to a model. The easiest and therefore simplest model to use will be solar model that is used to model a stars evolution and behaviour.

This assumes that the star is a perfectly formed sphere of gas that is held together by a careful balance between the gravitational force of the gas pulling it together and the pressure of the reaction pushing it apart. We can use this model to assume that, as the star is a sphere, then for stars far enough away the star would behave as a perfectly circular pin spot of light.

Therefore, the situation that is now present is to find either one filter that will perform the operation of returning each star in the image to a pin spot while still maintaining the colour intensity values for the star; or three separate filters that will return each colour channel to a circular pin-spot of light which will have the same effect as the single filter.

As each colour channel has a different problem and the distortion is different on all of them then it is more probable that the second option would be used rather than the first.

5.3.1 Analysis of the Red Colour Channel

Using the above model and figure 5.3, there are a couple of obvious anomalies with the original photograph:

Elongation in both the horizontal and vertical

The main star in the section of image can be seen to be skewed horizontally and vertically so that it has a slight elliptical shake this could be due to a number of factors which include:

- **Camera Shake**: When the shutter was pressed the camera shook just slightly so that it would be invisible to the naked eye when the image is viewed as a whole but when the pixel data is analysed then it can be seen.
- **Atmospheric Distortion**: This will have an effect over all the objects in this section of the image. This can be seen in the vertical direction as there are more vertically elliptic anomalies in this direction

Saturation and Noise

As there are patches in the graph that should be completely black as they represent the space between the stars there is a lot of noise on the red colour channel. Also the main star has a slightly flattened peak to it which shows that saturation has occurred on the red channel.

5.3.2 Analysis of the Green Colour Channel

This will be done in much the same way as the analysis on the red channel, i.e. using the same model and obviously using figures 5.4 and 5.5

Elongation in the Horizontal and Vertical

Unlike the red channel there is not much distortion in the horizontal direction on this colour channel however there is still distortion in the vertical direction. Therefore, one possibility is that the camera shake is not the only thing that is causing the elongation in the horizontal direction on the red channel.

However, the atmospheric distortion is almost certainly in the vertical direction as both the red and the green have distortions in that direction and at the same quantities.

Noise and Saturation

Unlike the red channel there is a lot of noise on the green. This, combined with the noise on the red will produce a brown/orange tint to the image and this is probably due to a not very well placed street lamp and it could be this that is causing the larger amount of noise than the red and the green channel.

Again the exposer time was too long and the main star has saturated the camera. There is really nothing that can be done about this.

5.3.3 Analysis of the Blue Colour Channel

The most obvious thing about the blue channel is that there is a huge area of very high intensity. This is almost certainly due to saturation of the camera. Another feature about the blue channel is that there is a larger darker area that appears on the other two channels. This confirms that the noise on the red and the green is mainly due to a street lamp as, even when covered, there will be some faint orange/brown light getting to the camera.

Finally, the vertical distortion on the colour channels is due to the atmospheric distortion and the camera shake adds the horizontal distortion.

From this filters and functions can be constructed that can remove these anomalies.

5.4 Possible Filters that can be written

5.4.1 Removal of Street Lamp Noise

As has been found in the analysis of the results there is noise due to a street lamp that was in the vicinity of the camera when the image was taken. Now looking at the raw data has shown that the Red and the Green are up by approximately 10 to 12 intensity values than the blue. Therefore, the following filter can be constructed using the equation:

$$P'_x = P_x - 12 \quad \text{where} \quad x \in \{R, G\} \quad \text{- (5.1)}$$

```
Def Remove_StreetLamp():
     def Remove(i,j):
          Pixel=im.getpixel((i,j))
          Pixel[0]=Pixel[0]-12
          Pixel[1]=Pixel[1]-12
          im.putpixel((i,j),(Pixel[0],Pixel[1],Pixel[2])
          return NONE

     for i in range(0,im.height,1):
          for j in range(0,im.width,1):
               Reomve(i,j)
     return NONE
```

This filter has been tested and really does improve the image quality as the noise from the street lamp is removed.

5.4.2 Other Filters

Another filter that has to be written is a filter that will change the geometry of the stars in the image. It must take a correction in x and y and then apply this to the image so that the stars can appear as pin-spots of light in image. Due to a lack in time I have been unable to write this filter however it could be written by someone else.

6. Conclusion and Evaluation

6.1 Conclusion

6.1.1 How far the initial objectives were achieved

The initial objectives were:

1. The Pixel Data is to be extracted from the image so that a graph can be plotted
2. The Pixel Function is to be found from the data points
3. A Fourier Transform is to be applied so that the point spread function can be found
4. A function is to be found so that when applied to the image a clearer version of the image is produced

With regard to these objectives the pixel data was extracted and a pixel function was found, however this function was produced in the Interpolation Function type in Mathematica and therefore a Fourier Transform could have been applied however, this would not have produced the required $f(x, y)$ and therefore no filter could have been written.

However, a filter was written that performed the necessary operation of removing the noise from the image. This did produce a result of a clearer image and stars that were indistinguishable from the noise were able to be seen.

6.1.2 Overall Conclusion and Suggestions for Further Work

In terms of the overall project, although some of the objectives were not completed due to a lack of time, the project went very well. There was a filter written that did improve the images and a program was written that could take an image in JPEG, BMP or TIFF format and it would apply the required operations.

Further work that could be done on this project could be:

- The conversion of the Pixel Function from the mathematica interpolation function type into f(x,y) format so that a Fourier Transform could be applied.
- A function written into the program that could perform this without the need for mathematica.
- A section in the main program that could do everything automatically.

Bibliography

Astronomy

[1] “History of Astronomy” - G. FORBES (Watts & Co, London, 1909)
[2] “A Practical Guide to CCD Astrophotography” - P. MARTINEZ and A. KLOTZ (Cambridge University Press, 1998
[3] “A History of Astronomy – from Thales to Kepler” - J.L.E. DREYER (Cambridge University Press, 1953)
[4] “The Art and Science of CCD Astrophotography” - D. RATLEDGE (Ed.) (Springer-Verlag London, 1997)

Image Processing

[5] “Introductory Computer Vision and Image Processing” - A. LOW (McGraw – Hill, 1991)

Mathematics

[6] “Principia Mathematica et Philosiphae Natralis” - I. NEWTON (Translated by A. MOTTE) (Prometheus Books, 1995)
[7] “Mathematical Methods for Science Students” - G. STEPHENSON (Longman's, London, 1962)

Programming Languages

[8] “Python in a Nutshell” - A. MARTELLI (O'Reilly & Associates Inc. , 2003)
[9] “Delphi 5 Developer's Guide” - S. TEIXEIRA and X. PACHECO (SAMS Publishing, 2000)