Draft version October 25, 2017
Preprint typeset using LaTeX style emulateapj v. 12/16/11

A SPECTACULAR TITLE

Daniel Heinesen

Draft version October 25, 2017

ABSTRACT

State problem. Briefly describe method and data. Summarize main results.

Subject headings: cosmic microwave background — cosmology: observations — methods: statistical

1. INTRODUCTION

2. THEORY

2.1. *Some Aberrations*

Aberrations are effect due to the lens and bending of light, that leads to curtain unwanted errors in an image.

2.1.1. *Coma*

Coma is an aberration that happens when the light hits the lens at an angle, leading some of the light to be bend more then other. This effect leads to a comet like picture.

2.1.2. *Chromatic Aberration*

Chromatic aberration happens because the reflective index is dependent on the wavelength of the light. This gives the different colors different focal points.

2.2. *The CCD*

This is a short qualitative summary of the inner workings of the charge-coupled device (CCD):

A pixel of the CCD consists of three layers: a doped semiconductor layer (usually some silica), separated from a conducting material by some isolator.

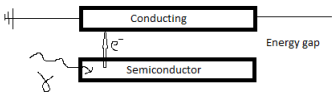

Fig. 1.— A (very) simple illustration of a CCD-chip

The isolating layer is characterized by some energy E_g (g for gap). This gap energy is so that electrons from the semiconducting layer cant overcome it and transverse to the conduction layer. But should a photon hit an electron with higher energy then E_g, the electron can overcome the gap energy and reach the conduction layer. The wavelength needed to give the electron this energy is called the cut-off wavelength

$$\lambda_c = \frac{hc}{E_g} \tag{1}$$

daniel.heinesen@sf-nett.no

Where h is Planck's constant and c the speed of light. This wavelength determines the largest wavelength the CCD is able to detect.

After the electrons reach the conduction they are lead to a register, who counts the number of electrons. The number of electrons should be proportional to the number of photons hitting the pixel, and from this the number of photons hitting the pixel can be calculated.

2.3. *Finding the Pixel Size*

To find the pixel size we used a slit to make a diffraction pattern on the screen. We can so use the single-slit diffraction pattern

$$a \sin(\theta) = m\lambda \tag{2}$$

Where a is the width of the slit, m is the order of the minimum, θ is the angle to said minimum and λ is the wavelength of the incoming light/laser.

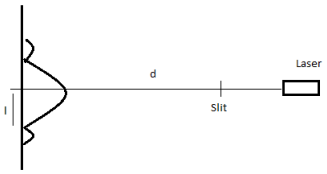

Fig. 2.— A crude illustration of the laser and slit. l is the distance to the first minimum and d the the distance from the slit to the CCD

If we look at the drawing, we see that the able θ is given as

$$\tan\theta = \frac{l}{d} \tag{3}$$

Where l is the distance to first minimum. But this angle is small, so we can use the small angle approximation $\tan\theta \approx \theta$. The distance l is also not completely known. We know the number of pixels from the center to the first minimum 3.2.1, but we do not know how this relate to the actual distance. So we say that

$$l = n_{pixels} \cdot c \tag{4}$$

Where n_{pixels} is the number of pixels, and c is some conversion factor, given in meters (the number of pixels are dimensionless). Using this and the small angle approximation we get that

$$\theta \approx \frac{n_{pixels} \cdot c}{d} \tag{5}$$

We are also going to use the small angle approximation on (2) and use our expression for θ

$$m\lambda \approx a\theta = a\frac{n_{pixels} \cdot c}{d} \tag{6}$$

This gives us a nice expression for the size of the pixels

$$c \approx \frac{m\lambda d}{n_{pixels} a} \tag{7}$$

2.4. *Different Cleaning Frame*

The three types of 'cleaning images'[1]:

2.4.1. *Bias*

When using a CCD there are a lot of electronics involved. The signal travels though a lot of cables. These cables may have either loss or gain of electrons. This normally leads to a constant gain or loss of signal. One of the most important gain in signal is from the electronics needed to take the electrons from the conduction layer of the CCD 2.2 to the register. This needs a constant potential to drive the electrons, which gives a gain in signal.

The bias is additive, meaning that to remove it we need to quantify the bias, then subtract it from the scientific image. To quantify the bias we take pictures without any light hitting the CCD, capturing only the signal from the electronics.[2]

2.4.2. *Dark Current*

As mentioned in sec. 2.2 the barrier between the semiconductor and conducting layer is given as an energy E_g. Ideally this energy gap is so large that no electron will wander across it unless exited by a photon. But this is a quantum system it is never that easy. The energy of the electron follows a Fermi-Dirac distribution *REF*

$$P(E,T) = \frac{1}{1 + e^{(E-E_f)/kT}} \tag{8}$$

Where E is the energy of the electron, T the temperature, E_f the Fermi energy and k the Boltzmann's constant. Thus for larger temperatures we are more likely to find electrons with a high energy. Meaning that for normal room temperature, there will be electrons that, due to quantum mechanics, will have enough energy to overcome the energy gap and reach the conducting layer. So even though there are no light, the CCD may register signals. This is dark current.

Just like the bias, dark current is additive, and must be subtracted from the raw scientific image. The value of the dark current is found be taking pictures without light.

2.4.3. *Flat Field*

Maybe the most important correction image is the flat field. Flat fields may be the result of many things. Due to differing efficiency from pixel to pixel in the CCD, the signal may not be uniform over the whole picture, even though the same light is shined upon all the pixels.

There may also be dust on the lenses or other imperfections. This is also adjusted with the flat field.

The flat field is multiplicative, and is adjusted for by dividing the raw scientific image by the flat frames. Since the flat field is due to imperfections in the pixels and on the lenses(and other instrument before the CCD), we want to take an image of a uniformly lit image. This gives us the flat frame.[3]

2.5. *Statistical Analysis*

To clean the images we want ideal flat field, dark frames and bias frames. But this is not the case. There are some noise in the images. This is characterized as the deviation from the mean of the image, or the standard deviation σ.

2.5.1. *Noise of the Bias*

For the bias frames we wan to look at just the noise which is not constant. We therefore subtract the two bias frames $B_1 - B_2$. This removes all the constant noise, leaving us with only the undesired variable noise. We can then find the standard deviation of this $\sigma_{B_1-B_2}$. If we look at the addition rules of variance *REF* we see

$$Var(aX - bY) = a^2 Var(X) + b^2 Var(b) \tag{9}$$

We then get:

$$\sigma^2_{B_1-B_2} = \sigma^2_{B_1} + \sigma^2_{B_2} \tag{10}$$

We expect that the noise in the bias frames should be constant over time, so the standard deviation, and thus variance, for the two bias frames should be equal[4]. So

$$\sigma^2_{B_1-B_2} = 2\sigma^2_B \Rightarrow \sigma_{B_1-B_2} = \sqrt{2}\sigma_B \tag{11}$$

Then thus we see that the standard deviation of the difference of the biases are $\sqrt{2}$ times that of a single bias frame.

We can also look at the mean of the sum of the two bias frames. Since the mean is linear *REF* we get that

$$E(B_1 + B_2) = E(B_1) + E(B_2) \tag{12}$$

2.5.2. *Noise of the Flat Frames*

The same is done with the flat frames. But we want to decrease noise of the flat field, and not increase it as it did in (11). For that we use several flat fields. So we want to calculate

$$\sigma_{(F_1+...F_{2n})-(F_2+...F_{2n+1})} \tag{13}$$

[1] for a lack of a better name

[2] We will also capture the dark current, but for simplicity this is ignored.

[3] I may use flat field and flat frame interchangeably throughout the text, but they are taken to mean the same: the image of the uniform lit background.

[4] Breaking the fourth wall: I'm not sure about this line of reason, but it gives the correct answer, so I included it.

Using n pairs of flat fields. But since we have added several frames, the scale of the noise increase with n. So we need to normalize with n. So what we want to calculate is

$$\frac{\sigma_{(F_1+...F_{2n})-(F_2+...F_{2n+1})}}{n} \tag{14}$$

If we now use the same relationship as in (11) we see what we want

$$\frac{\sigma_{(F_1+...F_{2n})-(F_2+...F_{2n+1})}}{n} = \frac{\sqrt{n}\sigma_F}{n} = \frac{\sigma_F}{\sqrt{n}} \tag{15}$$

So we see that as we include more flat frames, the random noise decreases as $1/\sqrt{n}$, and we are left with a more clean flat frame, which can be used to clean our science image.

2.5.3. *Read Out Noise*

We have now found the noise as the standard deviation of the pixel counts in the flat frame. This is in so called analog-to-digital units (ADU). We want to find the readout noise in electrons, so we need a conversion factor. This is given as

$$g = \frac{F_{electrons}}{\sigma_{electrons}} = \frac{(\bar{F}_1 - \bar{F}_2) + (\bar{B}_1 - \bar{B}_2)}{\sigma^2_{F_1-F_2} - \sigma^2_{B_1-B_2}} \tag{16}$$

Where F_i is a flat frame and B_i is the corresponding bias frame. The second expression comes from the fact that we use the bias frames to clean the flat frames before finding g. This conversion factor measures how many electrons which are needed to generate the signal. We can now find the readout noise as

$$\text{R.O.N} = g \cdot \sigma_{bias} \tag{17}$$

Which bias frame is used for this should ideally be the same, since we expect time-independent, constant bias noise.

2.5.4. *Cleaning the Image*

To clean the image we want to remove the effects made be the dark current and flat frame. As mentioned *REF* the dark current is additive while the flat field is multiplicative. To remove the noise from the the dark and flat frames, we take the average of multiple frames. But the flat frames do also have dark current, so we need to subtract the the corresponding average dark frame from the average flat frame and normalize it. This is the normalized flat frame. We can then simply clean the raw scientific image of the diffraction pattern by first subtract the average dark frame $D_{I,raw,average}$ and divide by the normalized master flat frame $F_{norm.master}$

$$I_{corrected} = \frac{I_{raw} - D_{I,raw,average}}{F_{norm.master}} \tag{18}$$

Where $I_{corrected}$ is the cleaned image of the diffraction pattern.

3. METHOD

3.1. *Color Camera, White Light and Lens*

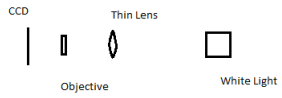

FIG. 3.— A crude illustration of the setup for the white light.

For this first experiment a white light was shined on and focus by a thin single lens. In the focal point of the lens a microscope objective was placed. The focal point of the lens was fond by holding a piece white paper in front of the lens and finding the distance where the light is most concentrated.

The CCD was then placed in front of the objective. Here we wanted to look at the properties of the different colors in the white light, so a RGB CCD was used[5].

We could now use out Graphical User Interface(GUI) to look at the resulting pattern. The result can be described as a histogram showing the amount of light received be the three sensors in the CCD 2.2. Due to chromatic aberration 2.1.2 the different colors focus at different focal length, so to look at this effect we moved the CCD carefully back- and forwards to focus the different colors; red, green and blue.

The GUI gave the possibility of viewing the pixel count of a slice of the image, for the different colors. These values were not saved, but the image itself was. The images was then used to make a plot showing the mean pixel count of each color over each column.

The GUI of the CCD also allowed us to adjust the pixel count, frame rate and exposure time, giving us the ability to study the effect this had on the picture.

Filters to filter out some of the colors in the white light was placed in front of the light, so we could look on the effect this would have on the light on the CCD.

3.2. *Monochromatic Camera and Single Slit*

3.2.1. *Setup*

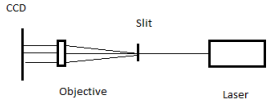

FIG. 4.— A crude illustration of the setup for the laser.

The white light was now switched out with a monochromatic red laser with wavelength $\lambda = (641 \pm$

[5] For this a Edmund Optics color USB camera was used.

12.3) nm *REF MY SELF*. To ensure that the laser didn't destroy the CCD, a damping filter was placed in front laser. Then a single slit with a width if $a = 100\mu m$ is placed in front of the laser the laser. The diffracted laser light was then captured by a monochromatic CCD[6].

The camera and slit was then adjusted until a sharp diffraction pattern appeared in the middle of the computer screen. The distance from the slit to the CCD was so measured carefully with a ruler. Two images image of the diffraction pattern were then taken. One with lower exposer used to calculate the pixel size, described here 2.3. The other one, with higher exposer, was used to test out the cleaning procedure 2.5.4.

To clean the picture the following series of pictures were taken 2.5.4. All of the apparatuses were removed, and for all of the pictures except the flat frames (see below) the dust cover was placed on the camera:

- 2 bias frames with minimum exposer
- 5 dark frames with the same exposer as the diffraction pattern
- 1 dark frame with maximum exposer
- 16 flat frames – see method below – with exposer set so that the average pixel value is about half that of max.
- 5 dark frames with the same exposer as for the flat frames

The flat frames are takes as follows: The dust cover was removed, then a white piece of paper was placed in front of the CCD and a white light was shined on the paper. The white paper ensured a homogeneous, white gradient over the whole CCD chip. With these axillary pictures we could clean the picture of the diffraction pattern.

3.2.2. *Statistical Analysis*

This part was done in Python*REF CODE*, but could be found with most other scripting languages, like R or IDL:

First a bias frame and the maximum exposer dark frames has converted to an array. From these arrays the max, min, mean and distribution of the pixel values was found and compared.

To find the readout noise we first looked at the two bias frames. To exclude the possibility of strange results near the edge of the picture, a symmetric 300x300 central part was extracted out of the to images and used. The two images(the central parts) were then added, and the mean pixel value was found. This is the same value as $\bar{B}_1 + \bar{B}_2$(12). We then calculated the noise for the biases. This was done by first subtracting the images 2.5.1, then finding the standard deviation of the resulting image $\sigma_{B_1-B_2}$(11).

The same was then done for two flat frames – 2 and 4 – : first the central regions of the images were found. A mean of the sum of the images was found $\bar{F}_1 + \bar{F}_2$, and then the noise of the difference of the images was found $\sigma_{F_1-F_2}$.

From this the conversion factor then consequently the readout noise can be calculated (16)

To see how the number of flat frames taken improved the normalized noise, the following was calculated with the same approach as the noise above: $\sigma_{F_1-F_2}$, $\sigma_{(F_1+F_3)-(F_2+F_4)}$ and so on until all the flat frames were used. For each time a new σ as calculated it was normalized with the number of pairs of flat frames used, then saved. The saved values for the noises were then plotted and compared.

To clean the image of the diffraction pattern we generally want to subtract away the noise from the dark frames, and divide away the noise from the flat frames. This was done as follows:

First an average of the 16 flat frames was found $F_{average}$. An average of the 5 dark frames corresponding to the flat frame $D_{F,average}$ was then found. We then removed the noise in the average flat frame made by the dark current, giving us the master flat frame $F_{master} = F_{average} - D_{F,average}$. This was then normalized with the scalar mean, giving us $F_{norm.master}$.

Secondly the dark frames corresponding to the image of the diffraction pattern was average to find $D_{I,raw,average}$.

Finally the final, clean image of the diffraction pattern was given as (18):

$$I_{corrected} = \frac{I_{raw} - D_{I,raw,average}}{F_{norm.master}} \tag{19}$$

This was done both for the central 300x300 image and the full field of view.

3.3. *Uncertainties*

3.3.1. *Color Camera and White Light and Lens*

To get a sharp diffraction pattern, the lens, objective and CCD has to be normal to the direction of the path of the light. If the lens is off the normal, we will get a coma, and due to the chromatic aberration in the lens (even thou it is thin, it has a physical thickness) the coma will be enhanced, leading to a smear out of colors(as seen here 3). This means that it might be difficult to get a sharp pattern, and the Airy disk might be larger then expected due to this coma, unless the lens is completely normal to the light(fortunately we are not measuring the Airy disk, so this uncertainty is only a curiosity).

When looking at the pixel counts for the different colors, we wanted to look at the pixel count when green dominated. But since green is a mix of blue and yellow, the blue is quite high when ever green is high. So there is quite a small range where green dominates all other colors, including blue. This range was difficult to find by only moving the camera back and forth, so the resulting image does not show very good results.

3.3.2. *Monochromatic Camera and Single Slit*

Much like with the white light and lens, the quality of the diffraction pattern is dependent on the single slit, objective and CCD being normal to the laser. This was adjusted as best as possible to make this uncertainty as

[6] A monochromatic Edmund Optics USB camera

low as possible. The slit has imperfections in its construction, making the diffraction pattern very uneven, instead of the straight lines we ideally want to see 16. The dominating source of error in our calculations of the pixel size came from the way the number of pixels to the first minimum was found: as one can see in the image of the diffraction pattern 16 is tilted at an angle, while the plotted slice of the image 17 is take horizontally, meaning that the measured distance to the minimum is larger than the actual distance. All of these uncertainties has been encompassed by a large, but fair uncertainty of ±5 pixels.

The distance from the slit to the camera was measured with a ruler. There was difficulty measuring this distance due to both the exact position of the slit and that of the CCD chip being obscured by their respective casings. So an uncertainty of ±0.2 cm was added.

As will be discussed later 5.2.4 the way we made the flat field was at best lacking. The bends in the paper, especially at the edges would lead to parts of the paper being darker than the uniform gradient we are looking for. But most grievous error done was taking the flat field without any slit or objective, as will be discussed later.

FIG. 6.— Image of blue light in focus.

FIG. 7.— Image of red light in focus.

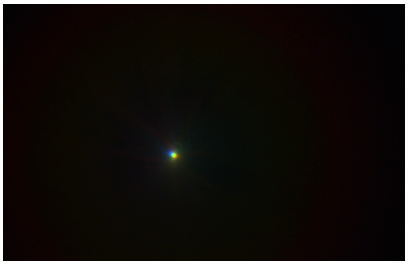

FIG. 8.— Image of green light in focus.

4. RESULTS

4.1. *Exposer Times*

	Pixel Clock [Hz]	Frame Rate [fps]	Exposer [ms]
Color Camera	7	5.32	57.309
Bias	5	0.18	0.08
Diffraction Pattern 1	40	20.61	0.08
Diffraction Pattern 2	40	20.61	0.124
Flat frames	5	2.58	36.883

4.2. *Color Camera, White Light and Lens*

Image of the full spectrum:

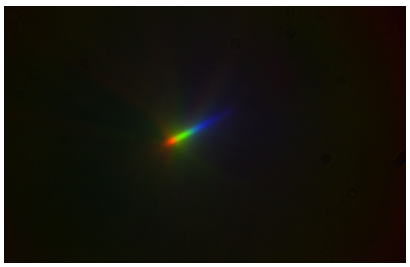

FIG. 5.— Image of the full spectrum. We can see that the comet like spread of the coma, combined with the separation of the different colors due to the chromatic aberration.

Capturing the images for the different colors gave:

We can see that for the blue focus it is easy to see that blue dominates. For red it is some what less clear, but we see that there are mostly red light there. But for green the dot in the middle almost looks white, while the ring around looks more red. If we look at the same image but with a red filter we see:

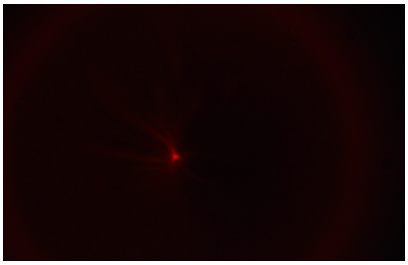

FIG. 9.— Image of green light but with a red filter. As we can see even though we had the CCD in the focal point of the green light, there is a lot of red light too. The reason is mentioned here 3.3.1

And as we expected there are alot of red in the light light imaged.

These images gave the following distribution:

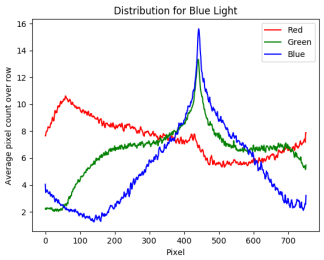

FIG. 10.— The distribution of the average pixel count for the different colors when in the focal point of the blue light. We can see that blue is dominating, but there are still quite a lot of green light.

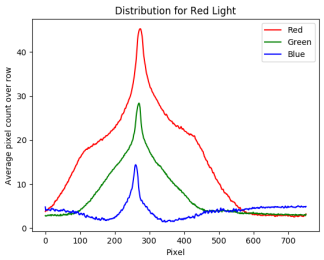

FIG. 11.— The distribution of the average pixel count for the different colors when in the focal point of the red light. We see that red is dominating, while blue is almost non-existent

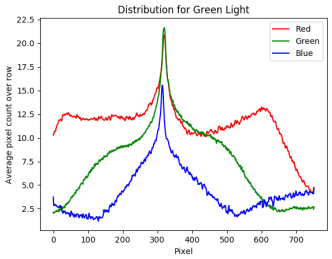

FIG. 12.— The distribution of the average pixel count for the different colors when in the focal point of the green light. We see that while green has the largest average pixel count, red is close behind and for most of the image higher than green.

The highest pixel counts for the given images are

Color in focus	Max Red	Max Green	Max Blue
Blue	24	137	202
Red	255	255	255
Green	211	255	255

TABLE 1
THE MAXIMAL PIXEL COUNT FOR THE DIFFERENT COLORS FOR THE DIFFERENT FOCAL POINTS.

4.3. *Monochromatic Camera and Single Slit*

4.3.1. *Bias and Dark Frame*

We started by looking at one bias and the dark frame for the maximum exposer.

FIG. 13.— Image of dark frame for maximum exposer.

FIG. 14.— Image of a bias frame.

As we can see there are more or less the same. Lets look at some data about them:

	Bias	Dark Frame
Max	15	75
Min	6	4
Max Location	$(439, 427)$	$(214, 478)$
Min Location	$(35, 528)$	$(441, 84)$

TABLE 2
DATA ABOUT THE BIAS AND DARK FRAME.

The distribution of pixel values for the two are given as

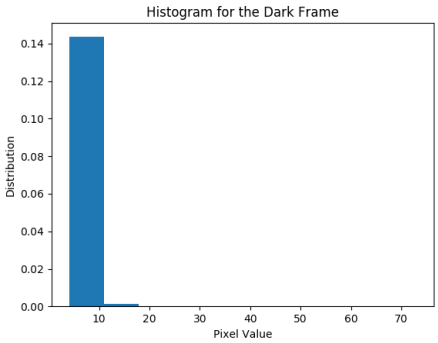

FIG. 15.— The distribution for the pixel values for the dark frame.

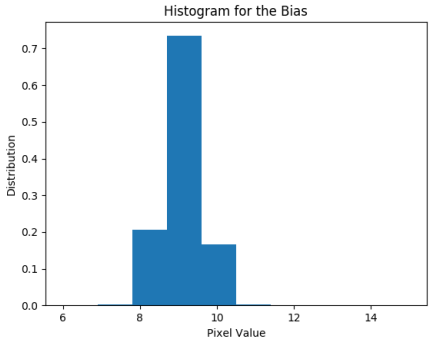

FIG. 16.— The distribution for the pixel values for the bias frame.

4.3.2. *Bias, Flat Frame and Noise*

For the analysis of the bias frames and two flat frames, the following data was obtained:

	Bias	Flat Field
Mean of sum	17.99	141.80
Noise	95.43	122.39

TABLE 3
DATA FOR THE CENTRAL REGIONS OF THE BIAS FRAMES, AND THE SECOND AND FORTH FLAT FRAMES.

From the same bias and flat frames (flat frame 2 and 4) g could be found as

$$g = 0.0210840557684 \text{ [electrons/ADU]} \tag{20}$$

And from a single bias frame the readout noise could be found

$$\text{R.O.N} = g \cdot \sigma_{bias} = 0.0210840557684 \cdot 0.592650538616 \tag{21}$$

$$= 0.0124954770073 \approx 0.0125 \tag{22}$$

Using all of the 16 flat frames, and seeing what this did to the noise, we got:

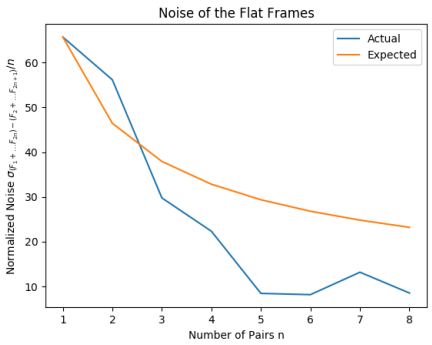

FIG. 17.— We can see as we use more and more pairs of flat frames, the noise decreases somewhere in order $1/\sqrt{n}$

4.3.3. *Pixel Size*

The image taken for this calculation was as follows:

FIG. 18.— Image of the diffraction pattern with low exposer, used to find the pixel size.

The middle slice of the image gave the following pixel values:

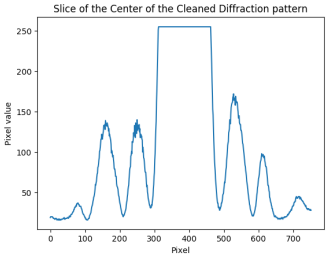

FIG. 19.— A slice of the middle of the cleaned image of the diffraction pattern.

If we look at the plot there are about 200 pixels from minimum to minimum, meaning that there are about 100 pixels from the center to the first minimum. We discussed the uncertainty of the distance in pixels in sec. 3.3.2. So adding that we get that

$$n_{pixels} = 100 \pm 5 \tag{23}$$

We can then use (7) to find the size of the pixels. We use that $m = 1$, $\lambda(641 \pm 12.3)$ nm *REF MYSELF*, $a = 100\mu m$ and $d = (8.5 \pm 0.2)$ cm:

$$c = \frac{(641 \pm 12.3) \text{ nm}(8.5 \pm 0.2) \text{ cm}}{100 \pm 5 \cdot 100\mu m} = (5.45 \pm 0.32) \ \mu\text{m} \tag{24}$$

So the size of one pixel of the CCD is (5.45 ± 0.32) μm.

4.3.4. *Cleaning the Scientific Image*

The raw of the diffraction pattern with slightly higher exposer was

FIG. 20.— Image of the diffraction pattern, used to test the cleaning procedure.

If we look close at the image one can see small circles. This is due to light diffracting of tiny dust particles some-

where on the instruments. This is some of the things we ideally want to remove with the flat fields.

After applying the cleaning procedure on the image we got the following result:

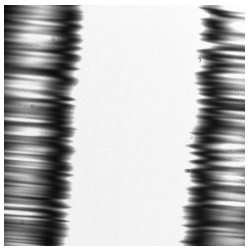

FIG. 21.— 300x300 center part of the corrected image.

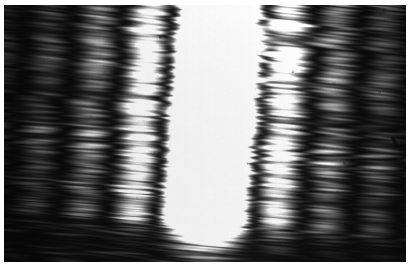

FIG. 22.— The full field of view of the corrected image.

5. CONCLUSIONS

5.1. *Color Camera, White Light and Lens*

As we saw in sec. 4.2 we got different colors due to the chromatic aberrations of the lens. This effect made it so that we could move the CCD into the focal point of the different colors, letting us isolate the different colors. If we then looked at the distribution of the mean pixel value of the different colors in the different (4, 10 and 5) we saw that even though the images looked – arguably – monochromatic, all the colors contributed to some degree. This was most apparent for the green light, where both blue and red was abundant. We confirmed this by placing a red filter in front of the lamp, and while in the green focal point, the image lighted up red 7. Looking at table 1 we see that for the most part the color in focus has the largest maximum value. But it looks like we used a bit high exposer, making more that one color having the maximum pixel value (255) in some images. The strange part is that in the image of the green light we expected red to also have a high maximum pixel count, but as seen from the table blue has a higher maximum value than red. The reason for this remains unknown to the author, and requires more research.

5.2. *Monochromatic Camera and Single Slit*

5.2.1. *Bias and Dark Frame*

As we can see from fig. 12 and 11 the dark frame and bias is quite similar –mostly black. This is due to them both being made in the same way, with the dust cover on the camera. But if we look at the maximum and minimum values we see some difference. The minima of the two images are more or less the same, but the maxima are far apart. This become more apparent from the distributions of pixel values, fig. 14 and 13. Here we see that the dark frame has a larger share of higher pixel values. This may be due to the fact that the dark frame was recorded with the maximum exposer time, hence making the CCD more sensitive and recording higher pixel counts.

5.2.2. *Bias, Flat Frame and Noise*

From table 3 we see that both the bias and the flat frame has quite large mean and noise, but as expected those of the flat frames are higher – since it is lit –.

The value for $g = 0.0211$ [electrons/ADU] seems quite small, since this means that we need about 1/50 electrons to get signal. This also makes the readout noise small, R.O.N= 0.0125 electrons. Other teams have calculated readout noise of close to 100 [7]. The author are not why this is the case, but think it may have to do with the exposer of the camera: that changing the exposer drastically increases the readout noise[8]. The conversion factor also varies drastically with different flat frames, and for frame 1 and 2 a negative factor is obtained.

If we look at fig. 15 we see that we expected the noise to decrease with $1/\sqrt{n}$, but if we look at the actual noise it actually decreases as $1/n$. The reason for this is unknown. It may simply be that we some noise in the noise, or that we have to few points to see that trend well. It may also be something wrong in the authors calculation of the noise. What ever the difference in the expected decrease and the calculated decrease, this is something that needs more research.

5.2.3. *Pixel Size*

We got a pixel size of (5.45 ± 0.32) μm, which is quite a good result, with a small uncertainty. But as discussed the number of pixels to the minimum may have been overestimated, and a larger uncertainty may be justified, or a better method of finding this distance needs to be found.

A better slit is also needed, to get more straight lines in the diffraction pattern.

5.2.4. *Cleaning the Scientific Image*

As seen in fig. 18 there are a lot of impurities in the picture. The most noticeable being the diffraction patterns made by dust on the instruments. Most of these impurities can be improved by the flat field[9]. But if we look at the cleaned figures 19 and 20 we see little difference. The diffraction patterns from the dust particles are still there. This is due to the ineffective way we made the flat frames:

Firstly: As discussed in under uncertainties, a piece of paper bends, making the flat frame less uniform. But the largest flaw was placing the paper right in front of the CCD. The dust observed in the images are most likely on the slit or objective. And since there were excluded

[7] Can not cite this since it is from other students rapports

[8] Not sure if it will increase with increasing or decreasing exposer.

[9] Dark current and bias being so small that cleaning these will have only a small difference.

from the flat frames, we have no way of correcting for these!

So to get a better, cleaner image we have to go back into the laboratory and get better flat frames with the objective and slit.

6. CODE

All the code have been tested on window 10, 64 bit, Python 3.5.2. The author can not guaranty that it will work on other computers and operating systems.

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 14 13:49:42 2017

@author: Daniel
"""

import numpy as np
import matplotlib.pyplot as plt
from PIL import Image



class image_processor:
    def __init__(self, file_folder="",file_extension=".bmp"):
        self.folder = file_folder
        self.extension = file_extension


    def read_image(self,name,fullFOV=True):
        im = Image.open(self.folder + name + self.extension)

        if fullFOV:
            return np.copy(np.array(im))
        elif not fullFOV:
            return self.get_center(np.array(im))

    def read_images(self,names,fullFOV=True):
        ims = []
        for name in names:
            ims.append(self.read_image(name,fullFOV))

        return np.array(ims)

    def make_histogram(self,image, title=""):
        n, bins, patches = plt.hist(image.flatten(),normed=True)
        plt.title(title)
        plt.xlabel("Pixel Value")
        plt.ylabel("Distribution")
        plt.show()

    def get_extrema(self, image):
        return np.min(image), np.max(image)

    def get_position_extrema(self,image):

        return np.unravel_index(np.argmin(image),image.shape), \
                    np.unravel_index(np.argmax(image),image.shape)

    def get_mean(self,image):
        return np.mean(image)

    def average_image(self,images):
        return np.mean(images,axis=0)
        im_sum = images[0]
        for i in images[1:]:
            im_sum += i

        return im_sum/images.shape[0]


    def get_std(self,image): #get std...
        return np.std(image)

    def get_center(self, image, size=300):
```

```
        image_shape = image.shape
        center = (int(image_shape[0]/2),int(image_shape[1]/2))
        return np.copy(image[center[0] - int(size/2):center[0] + int(size/2),\
                             center[1] - int(size/2):center[1] + int(size/2)])


    def process_image_mean_and_noise(self,image_names):
        b1 = self.read_image(image_names[0])
        b2 = self.read_image(image_names[1])

        b_sum = b1 + b2
        b_center = self.get_center(b_sum)
        b_center_mean = self.get_mean(b_center)

        b_sub = b1-b2
        b_center_sub = self.get_center(b_sub)
        b_center_std = self.get_std(b_center_sub)

        return b_center_mean, b_center_std

    def get_g(self,bias_names,flat_names):
        f_mean,f_std = self.process_image_mean_and_noise(flat_names)
        b_mean,b_std = self.process_image_mean_and_noise(bias_names)

        return (f_mean - b_mean)/(f_std**2 - b_std**2)

    def get_RON(self,bias_names,flat_names):
        bias_noise = np.std(self.read_image(bias_names[0]))
        g = self.get_g(bias_names,flat_names)
        print(bias_noise)

        return g*bias_noise


    def get_noise(self,image_names, save=False):
        even_sum = self.read_image(image_names[0],fullFOV=False)
        odd_sum = self.read_image(image_names[1],fullFOV=False)

        if save:
            noises= np.zeros(int(len(image_names)/2))
            noises[0] = self.get_std(odd_sum-even_sum)

        for i in range(2,len(image_names),2):
            even_sum += self.read_image(image_names[i],fullFOV=False)
            odd_sum += self.read_image(image_names[i+1],fullFOV=False)
            if save:
                noises[int(i/2)] = self.get_std(odd_sum-even_sum) /(i/2)

        if save:
            return noises, self.get_std(odd_sum-even_sum)

        return self.get_std(odd_sum-even_sum)

    def get_picture_slice(self,name):
        if isinstance(name,np.ndarray):
            data = name
        else:
            data = self.read_image(name,fullFOV=True)

        return data[int(data.shape[0]/2),:]

    def plot_picture_slice(self,name):
        data = self.get_picture_slice(name)

        plt.plot(data)
        plt.title("Slice of the Center of the Cleaned Diffraction pattern")
        plt.xlabel("Pixel")
        plt.ylabel("Pixel value")
        plt.show()
```

```
    def clean_image(self,I_raw_name,raw_dark_names,flat_names,flat_dark_names,fullFOV=True):
        I_raw = self.read_image(I_raw_name,fullFOV=fullFOV)
        raw_darks = self.read_images(raw_dark_names,fullFOV=fullFOV)
        raw_dark_avr = self.average_image(raw_darks)

        flats = self.read_images(flat_names,fullFOV=fullFOV)


        flat_darks = self.read_images(flat_dark_names,fullFOV=fullFOV)

        flat_dark_avr = self.average_image(flat_darks)
        flat_avr = self.average_image(flats)

        flat_master = flat_avr - flat_dark_avr


        flat_master_norm = flat_master/self.get_mean(flat_master)

        return (I_raw - raw_dark_avr)/(flat_master_norm)


    def save_image(self,image,name):
        im = Image.fromarray(np.uint8(np.where(image>255, 255,image)))
        im.save(name)


    def plot_color_hist(self,name):
        data = self.read_image(name)


        value_count = np.zeros((3,256))
        for i in range(3):
            for val in range(256):
                value_count[i,val] = np.sum(data[:,:,i] == val)

        print("The maximum value for red is: ",np.max(data[:,:,0]))
        print("The maximum value for blue is: ",np.max(data[:,:,2]))
        print("The maximum value for green is: ",np.max(data[:,:,1]))

        plt.plot(value_count[0],"r")
        plt.plot(value_count[1],"g")
        plt.plot(value_count[2],"b")

        plt.show()

    def plot_highest_row_color(self,name,title=""):
        data = self.read_image(name)

        print("The maximum value for red is: ",np.max(data[:,:,0]))
        print("The maximum value for blue is: ",np.max(data[:,:,2]))
        print("The maximum value for green is: ",np.max(data[:,:,1]))

        data = np.mean(data,axis=0)

        plt.plot(data[:,0],"r",label="Red")
        plt.plot(data[:,1],"g",label="Green")
        plt.plot(data[:,2],"b",label="Blue")
        plt.title(title)
        plt.xlabel("Pixel")
        plt.ylabel("Average pixel count over row")
        plt.legend()

        plt.show()

    def convert_images(self,names):
```

```
        data = self.read_images(names)

        for i,im in enumerate(data):
            self.save_image(im,names[i] + ".png")


if __name__=="__main__":
    ip = image_processor()
    bias = ip.read_image("bf1")
    dark = ip.read_image("df_max_exp")

    ip.make_histogram(bias,title="Histogram for the Bias")
    ip.make_histogram(dark,title="Histogram for the Dark Frame")

    print("For bias: min = {}, max = {}".format(ip.get_extrema(bias)[0],ip.get_extrema(bias)
        [1]))
    print("For dark frame: min = {}, max = {}".format(ip.get_extrema(dark)[0],ip.get_extrema
        (dark)[1]))

    print("For bias, the position is: min = {}, max = {}".format(ip.get_position_extrema(
        bias)[0],ip.get_position_extrema(bias)[1]))
    print("For dark frame, the position is: min = {}, max = {}".format(ip.
        get_position_extrema(dark)[0],ip.get_position_extrema(dark)[1]))

    print("For bias: mean = {}".format(ip.get_mean(bias)))
    print("For dark frame: mean = {}".format(ip.get_mean(dark)))

    print("Statistics for bias and flat")
    print("----------------------------")

    bias_mean, bias_std = ip.process_image_mean_and_noise(["bf1","bf2"])
    flat_mean, flat_std = ip.process_image_mean_and_noise(["ff2","ff4"])

    print("For bias: mean = {}, std = {}".format(bias_mean,bias_std))
    print("For flat: mean = {}, std = {}".format(flat_mean,flat_std))




    print("The convertion constant: g = ",ip.get_g(["bf1","bf2"],["ff2","ff4"]))
    print("The readout noise RON = ",ip.get_RON(["bf1","bf2"],["ff2","ff4"]))

    print("Plotting noise over flates")
    print("--------------------------")

    flat_frames = ["ff" + str(i) for i  in range(1,17)]
    ns = np.arange(1,9)
    noise = ip.get_noise(flat_frames,save=True)[0]
    expected_noise = noise[0]*1/np.sqrt(ns)
    plt.plot(ns,noise,label="Actual")
    plt.plot(ns,expected_noise,label="Expected")
    plt.legend()
    plt.title("Noise of the Flat Frames")
    plt.xlabel("Number of Pairs n")
    plt.ylabel(r"Normalized Noise $\sigma_{(F_1 + \ldots F_{2n})- (F_2 + \ldots F_{2n+1})}/
        n$")
    plt.show()

    print("Cleaning")
    print("--------")


    raw = "3_2"
    raw_darks = ["df" + str(i) + "_4" for i in range(1,6)]
    flat_frames = ["ff" + str(i) for i  in range(1,17)]
    flat_darks = ["df_ff" + str(i) for i in range(1,6)]
```

```
    corr_I = ip.clean_image(raw,raw_dark_names=raw_darks,flat_names=flat_frames,
        flat_dark_names=flat_darks,fullFOV=True)
    #ip.save_image(corr_I,"corrected_image_fov.png")


    print("Slicing and Color Images")
    print("------------------------")

    ip.plot_picture_slice("3_1")

    #ip.plot_color_hist("rod_fokus")
    print("For Green Focus:")
    ip.plot_highest_row_color("gront_fokus",title="Distribution for Green Light")
    print("For Red Focus:")
    ip.plot_highest_row_color("rod_fokus",title="Distribution for Red Light")
    print("For Blue Focus:")
    ip.plot_highest_row_color("blatt",title="Distribution for Blue Light")

    ip.convert_images(["df_max_exp"])
```