AST2210 Report 2

Andreas Ellewsen

November 10, 2014

Contents

1 Introduction

This lab will go through basic use of CCD cameras, recording data, and how to analyze the images in IDL.

2 Image recording

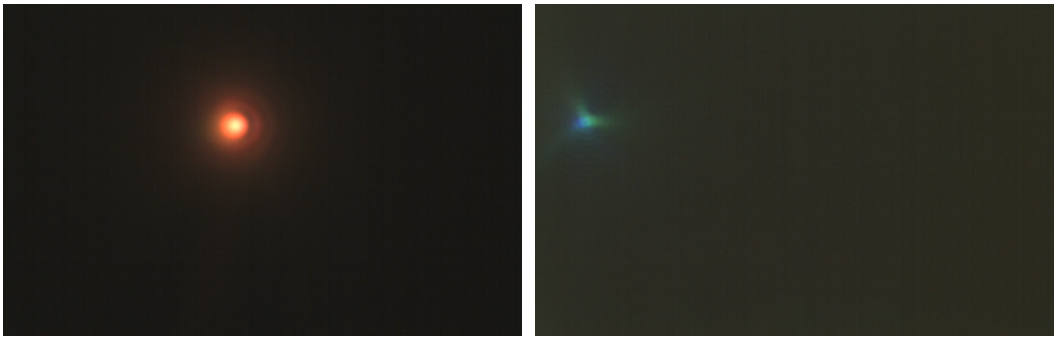
For the two first exercises the color Edmund Optics USB camera is used in a set-up with a white light lamp, a thin singlet lens, and a microscope objective. The program uc480viewer is used to control the camera. A picture is then taken of the white light source.

2.1 Exercise 1

There are many different settings to adjust in the program. This exercise focuses on three of them. These are 'histogram', 'value on horizon line', and 'camera properties'. Experimenting with the settings shows that 'Value on horizon line' brings up a graph of the intensity for red, green and blue, in the spot one chooses. 'Histogram' was forgotten when doing the lab. And 'Camera properties' brings up a window with three more settings. The first one is 'Pixel clock'. This changes the readout rate of the CCD. The second is 'Frame rate'. This changes the rate in which the program updates the picture. The last setting is 'Exposure time', which does exactly what one would expect.

2.2 Exercise 2

The focus position of the microscope objective is then changed. It is observed that the color seen on the screen changes. Increasing the distance between the camera and the objective makes the picture more blue. Decreasing the distance makes it more red. Another thing to notice here is that, since we didn't manage to set everything in a perfectly straight line, the color of the dot (including the airy disk) didn't just change. It moved both vertically and horizontally. The maximum count for each color (with fixed exposure time) should have been checked, but this was overlooked during the lab. Nonetheless it was clear that they were not the same for any of them. Red was most intense, decreasing a little for each step closer to blue. The pictures can be seen in figure 1.



(a) Red airy disk (b) Blue airy disk

Figure 1: The pictures

2.3 Exercise 3

The camera is then changed to a monochromatic Edmund Optics USB Camera in a set-up with a laser and a 100 μm slit. Using the diffraction pattern on the live view of the camera, the width of the pixels in the sensor is measured (in μm). The uncertainty of the measurement is also calculated. This is done by using the formula for single-slit diffraction, $asin(\theta) = m\lambda$, where a is the slit width, and m the order of the minimum. Using basic trigonometry $sin(\theta) = p/D$, where D is distance between slit and camera, and p is distance from max to first minimum. During the lab, D was measured to be 15.8 $\pm$ 0.1 cm. This gives a distance between the two first minimums of $p = 1.090 \pm 0.007mm$. Looking at the picture taken, the distance between the two minimums is $160 \pm 20pixel$. Putting this together one gets a resolution of $6.926 \pm 0.909\mu m/pixel$, which seems to be reasonable.

2.4 Exercise 4

A well exposed image of the diffraction pattern is recorded. Exposure time, frame rate and pixel clock of the image is noted. The image can been seen in figure 2.The settings chosen for this image can be seen in table 1.

Exposure time	0.384 ms
Frame rate	2.82 fps
Pixel clock	40

Table 1: Settings for image

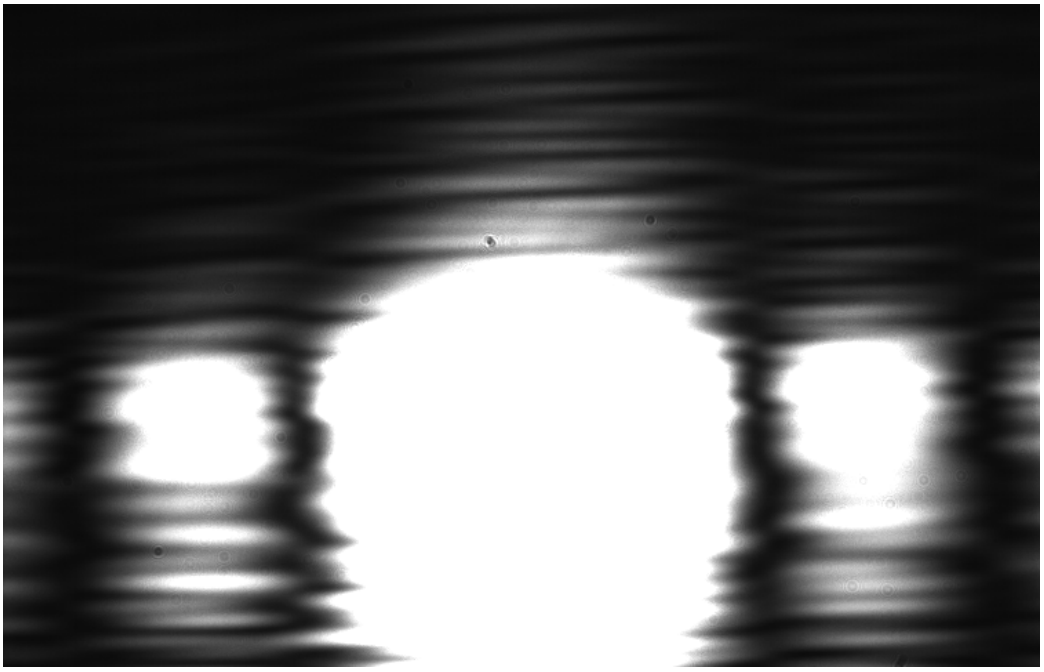


Figure 2: Single slit diffraction pattern

2.5 Exercise 5

The laser is then switched off, and the dust cover is put on the camera. The slit is then removed. A series of frames are then taken. First we record 2 bias frames by turning down the exposure time to the minimum value. Then we record 5 dark frames with the same exposure time as in exercise 4. And finally a dark frame at the maximum exposure time. The dust cover of the camera is removed and a very basic type of flat field image is recorded by using a white paper to reflect light from the ceiling into the camera. The integration time is adjusted so that the average pixel value is between halfway and one third of the maximum of the camera. This is done 16 times, noting the exposure time. The dust cover is then put back on and 5 dark frames are recorded with the same exposure time. The time that was used in this lab was 33.107 ms. The pictures don't look very interesting, but some of them can been seen in figure 3.

(a) Bias (b) Dark current (c) Flatfield

Figure 3: Examples of pictures taken

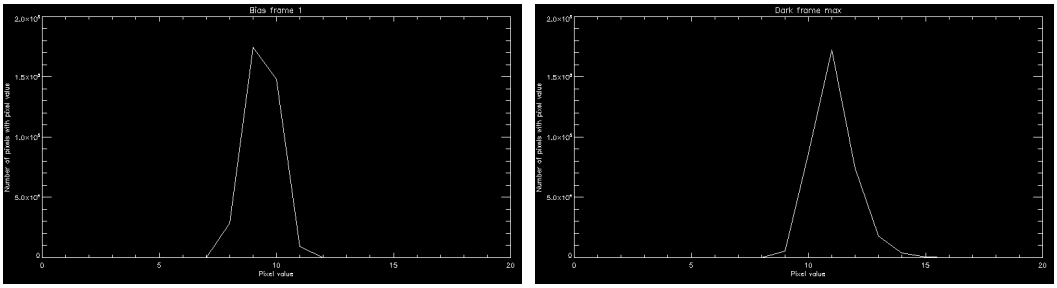
3 Post processing

3.1 Exercise 6

We start by loading one bias frame, and the dark frame with maximum exposure time into IDL. Average,minimum, and maximum pixel values are computed for both and can be seen in table 2. The histograms of the pixel values is plotted in figure 4.

	Bias1	Darkmax
Average	9.38291	11.0843
Minimum	7	8
Maximum	16	94

Table 2: Average, max, and min for the two pictures



(a) Histogram of Bias 1 (b) Histogram of Bias 2

Figure 4: The histograms for each of the bias frames

Examining the frames one notices that there are some differences between the two frames. The biggest difference is that the mean pixel value is different

for the two. When locating the pixels with the maximum value for both images it is found that they are not the same. This is not as expected, since one would assume that the "sensitive" pixel should be the same pixel for both frames.

All of this is done using the IDL code below:

```
bias1   = read_bmp('bias1.bmp')
darkmax = read_bmp('darkmax.bmp')

;Print information for each image to terminal
print,'Info for bias1'
print,'Maximum:  '      + strtrim(max(fix(bias1)),2)
print,'Minimum:  '      + strtrim(min(fix(bias1)),2)
print,'Average:  '      + strtrim(avg(fix(bias1)),2)

print,'Info for darkmax'
print,'Maximum:  '      + strtrim(max(fix(darkmax)),2)
print,'Minimum:  '      + strtrim(min(fix(darkmax)),2)
print,'Average:  '      + strtrim(avg(fix(darkmax)),2)

window,0
plot,histogram(bias1),xrange = [0,20], title='Bias frame 1', xtitle =
write_png,'histogram1.png',tvrd()
window,1
plot,histogram(darkmax),xrange = [0,20],title='Dark frame max',xtitle =
write_png,'histogram2.png',tvrd()

maxbias1 = where(bias1 eq max(bias1))
maxplacementbias1 = array_indices(bias1,maxbias1)
print,maxplacementbias1

maxdarkmax = where(darkmax eq max(darkmax))
maxplacementdarkmax = array_indices(darkmax,maxdarkmax)
print,maxplacementdarkmax

end
```

3.2 Exercise 7

Next we load both bias frames, add them together, and measure the mean value of the central square region, 300pixels on a side. The quantity referred

to as the mean value here is $\overline{B_1} + \overline{B_2}$. Subtracting one bias frame from the other, and measuring the standard deviation for the central region gives $\sigma_{B_1-B_2}$ All of this is done using the IDL code below:

```
bias1 = read_bmp('bias1.bmp')
bias2 = read_bmp('bias2.bmp')
biassum = bias1 + bias2
biasdiff = bias1 - bias2
biassquare1 = biassum[376-149:376+150,240-149:240+150]
biassquare2 = biasdiff[376-149:376+150,240-149:240+150]
meanbiassquare1 = mean(biassquare1)
stddevbiassquare2 = stddev(biassquare2)

print, 'Mean of central region sum: ' + strtrim(meanbiassquare1,2)
print, 'Std dev of central region diff: ' + strtrim(stddevbiassquare2,2
```

3.3 Exercise 8

Any fixed bias patterns are then removed by subtracting the two bias frames, leaving just the noise from the two bias frames $\sigma_{B_1-B_2}$. This should be $\sqrt{2}$ times the noise of one bias frame. This is used later, when calculating the noise in the flat fields as we increase the number of flat fields we use.

3.4 Exercise 9

The same is done for two flat frames, giving us $\overline{F_1} + \overline{F_2}$, and $\sigma_{F_1-F_2}$. All of this is done using the IDL code below:

```
flat3 = read_bmp('flat3.bmp')
flat4 = read_bmp('flat4.bmp')
flatsum  = flat3 + flat4
flatdiff = flat3 - flat4
flatsquare1 = flatsum[376-149:376+150,240-149:240+150]
flatsquare2 = flatdiff[376-149:376+150,240-149:240+150]
meanflatsquare1 = mean(flatsquare1)
stddevflatsquare2 = stddev(flatsquare2)

print, 'Mean of central region sum: ' + strtrim(meanflatsquare1,2)
print, 'Std dev of central region diff: ' + strtrim(stddevflatsquare2,2
```

3.5 Exercise 10

The measurements done so far have been done in pixel counts, or analog-to-digital units (ADU). There is a conversion factor g that relates 1 ADU to the number of actually measured electrons. And since one can expect the signal to display Poisson statistics measured in electrons, one expects $\sigma_e lectrons = \sqrt{F_{electrons}}$. In this case both σ and F have been multiplied by this factor, and thus we have $g\sigma_{electrons} = \sqrt{gF_{electrons}}$. Solving for g:

$$g = \frac{F_{electrons}}{\sigma^2_{electrons}}[electrons/ADU]. \tag{1}$$

Correcting for bias and noise related to bias:

$$g = \frac{(\overline{F_1} + \overline{F_2}) - (\overline{B_1} + \overline{B_2})}{(\sigma^2_{F_1-F_2}) - (\sigma^2_{B_1-B_2})}[electrons/ADU] \tag{2}$$

With all the information gathered so far one can compute this conversion factor g. Thus one finds that in this case: $g = 0.0107027[electrons/ADU]$ This is done using the IDL code below:

```
g = (meanflatsquare1 - meanbiassquare1) / (stddevflatsquare2^2 - stdde
print, 'Conversion factor g = ', strtrim(g,2)
```

3.6 Exercise 11

The only source of noise in a bias frame should be the readout noise. This noise is calculated in the following program, and is found to be 1.07906 electrons per pixel.

```
readoutnoise = stddevbiassquare2*g
print, 'Readout noise for bias frames = ', strtrim(readoutnoise,2)
```

3.7 Exercise 12

By adding the flats together and plotting the noise one can see that the noise is reduced by adding more flats. Unfortunately one of the flat fields taken in the lab was corrupt. Because of this I chose to use only 14 flats. The resulting plot can be seen in figure 5.

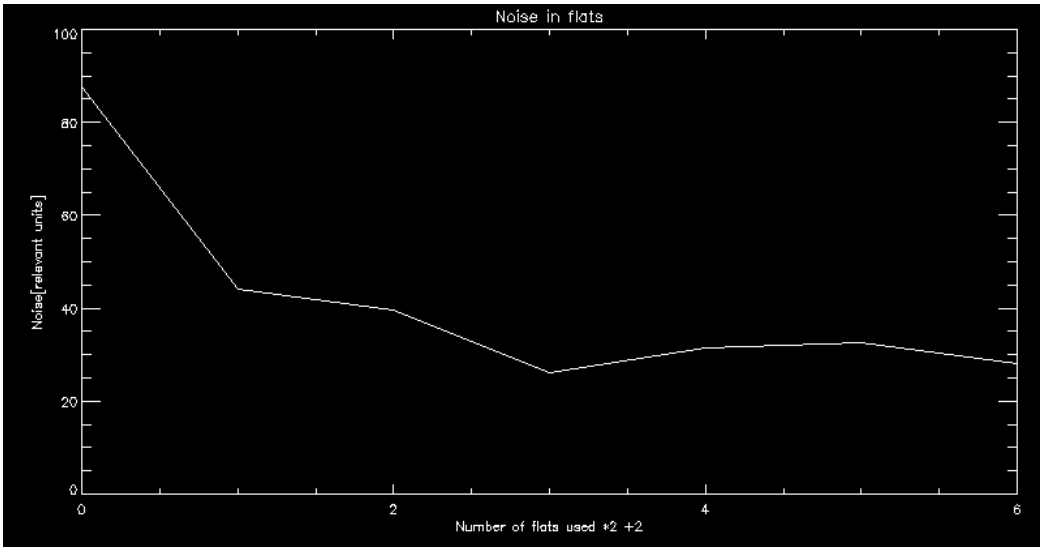


Figure 5: Noise reduction in flats

This was generated by the following code:

```
flat5  = read_bmp('flat5.bmp')
flat6  = read_bmp('flat6.bmp')
flat7  = read_bmp('flat7.bmp')
flat8  = read_bmp('flat8.bmp')
flat9  = read_bmp('flat9.bmp')
flat10 = read_bmp('flat10.bmp')
flat11 = read_bmp('flat11.bmp')
flat12 = read_bmp('flat12.bmp')
flat13 = read_bmp('flat13.bmp')
flat14 = read_bmp('flat14.bmp')
flat15 = read_bmp('flat15.bmp')
flat16 = read_bmp('flat16.bmp')

stddevflatsquarefinal = make_array(1,7,/float,value = 0)
flatdiff = flat3 - flat4
flatsquarefinal = (flatdiff[376-149:376+150,240-149:240+150])/sqrt(2)
stddevflatsquarefinal[0] = stddev(flatsquarefinal)

flatdiff = (flat3+flat5) - (flat4+flat6)
flatsquarefinal = (flatdiff[376-149:376+150,240-149:240+150])/sqrt(4)
stddevflatsquarefinal[1] = stddev(flatsquarefinal)

flatdiff = (flat3+flat5+flat7) - (flat4+flat6+flat8)
```

```
flatsquarefinal = (flatdiff[376-149:376+150,240-149:240+150])/sqrt(6)
stddevflatsquarefinal[2] = stddev(flatsquarefinal)

flatdiff = (flat3+flat5+flat7+flat9) - (flat4+flat6+flat8+flat10)
flatsquarefinal = flatdiff[376-149:376+150,240-149:240+150]/sqrt(8)
stddevflatsquarefinal[3] = stddev(flatsquarefinal)

flatdiff = (flat3+flat5+flat7+flat9+flat11) - (flat4+flat6+flat8+flat1
flatsquarefinal = flatdiff[376-149:376+150,240-149:240+150]/sqrt(10)
stddevflatsquarefinal[4] = stddev(flatsquarefinal)

flatdiff = (flat3+flat5+flat7+flat9+flat11+flat13) - (flat4+flat6+flat
flatsquarefinal = flatdiff[376-149:376+150,240-149:240+150]/sqrt(12)
stddevflatsquarefinal[5] = stddev(flatsquarefinal)

flatdiff = (flat3+flat5+flat7+flat9+flat11+flat13+flat15) - (flat4+fla
flatsquarefinal = flatdiff[376-149:376+150,240-149:240+150]/sqrt(14)
stddevflatsquarefinal[6] = stddev(flatsquarefinal)

window,0
plot,[0,1,2,3,4,5,6],stddevflatsquarefinal, title='Noise in flats',xti

write_png,'Noiseinflats.png',tvrd()
```

3.8 Exercise 13

Finally, all of the frames will be used to correct the diffraction image. This is done using the following formula:

$$\frac{Rawimage - Dark_R}{Flat - Dark_F} \times m = Correctedimage \tag{3}$$

Where $Dark_R$ is a 'master dark frame'. This 'master dark frame' is the basically the picture you get if you take the average of all the dark frames with the same exposure time as the raw image and make one picture out of it. $Dark_F$ is the "master dark frame" made in the exact same way, only using the darks with the same exposure time as the flats. And $m = avg(F - Dark_F)$. This was generated by the following code:

```
Rawimage = read_bmp('CCD single slit.bmp')
darkflat1 = read_bmp('darkflat1.bmp')
darkflat2 = read_bmp('darkflat2.bmp')
```

```
darkflat3 = read_bmp('darkflat3.bmp')
darkflat4 = read_bmp('darkflat4.bmp')
darkflat5 = read_bmp('darkflat5.bmp')
dark1 = read_bmp('dark1.bmp')
dark2 = read_bmp('dark2.bmp')
dark3 = read_bmp('dark3.bmp')
dark4 = read_bmp('dark4.bmp')
dark5 = read_bmp('dark5.bmp')

Df = (darkflat1+darkflat2+darkflat3+darkflat4+darkflat5)/5.0 ;Master d
Ddiff = (dark1+dark2+dark3+dark4+dark5)/5.0
F = (flat3+flat4+flat5+flat6+flat7+flat8+flat9+flat10+flat11+flat12+fla
m = avg(F-Df)
Corrected = (Rawimage-Ddiff)*m /(F-Ddiff)

window,1
plot_image,Rawimage
write_png,'Raw_image.png',tvrd()
window,2
plot_image,Corrected
write_png,'Corrected.png',tvrd()
```

4 Conclusion

The result of this is a corrected image, which can be seen in figure 6 along with the raw image in figure 7. One must admit that there is little improvement to be seen. This is probably because of the way the flat fields were made. The illumination was obviously not uniform which it ideally should have been.

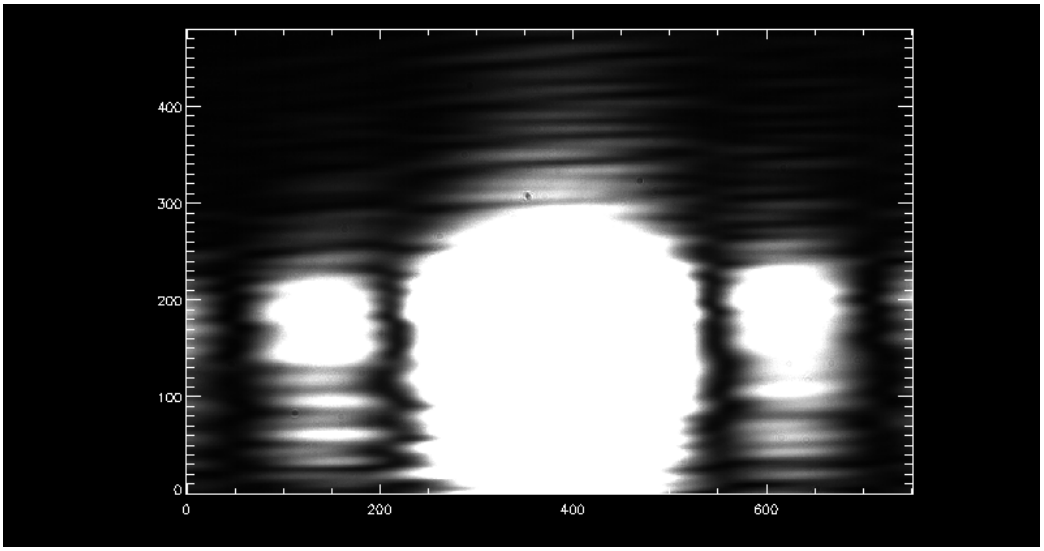


Figure 6: Raw image

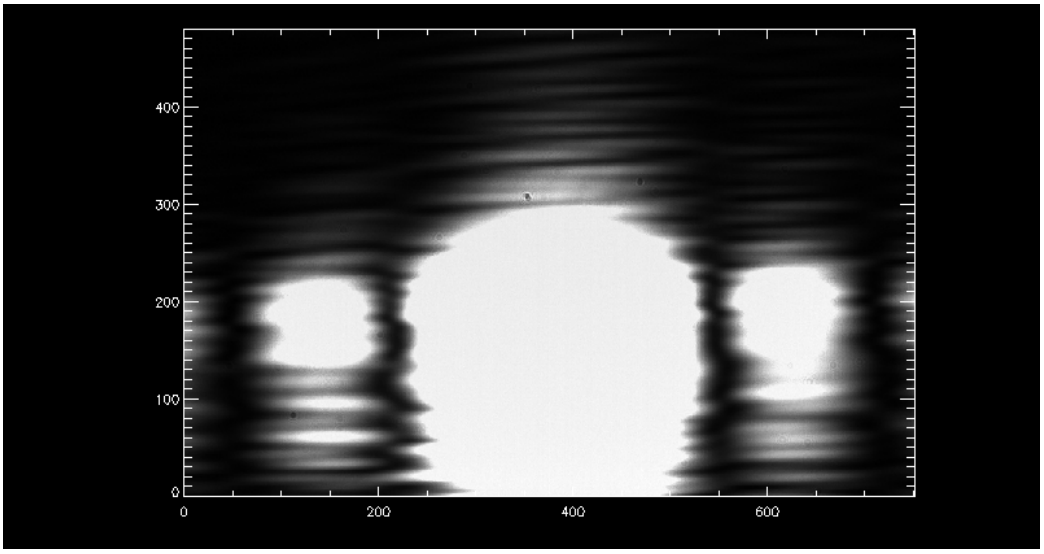


Figure 7: Corrected image