Draft version October 23, 2017
Preprint typeset using LaTeX style emulateapj v. 11/26/04

CHARGE-COUPLED DEVICE - CCD

Hans-Petter Harveg

Draft version October 23, 2017

ABSTRACT

In this experiment we went through the basics of operating a *Charge-couple device (CCD)* cameras which is a digital technique for recording images. In this digital proccess, errors due to electronics do occur, which we have analyzed. We recorded data; a raw image and images used for analyzing the the images namely *bias*, *dark frames* and *flat frames*. We have used the interactive data language *IDL* to perform all image analysing and image proccesing to remove noise from the raw image and produced a cleaner image. The result cannot be detected by the naked eye, but analysis shows reduction in the final image noise. The procces of using CCD's for recording images shows promesing results with a quantum efficiency of up to 95%.

Subject headings: ccd — astronomy: photography, image analysis — methods: experimental

1. INTRODUCTION

In astrophysics, we rely completely on information transmitted by signals from the universe. For this reason, the method of collecting these signals, namely photons is crucial. Using photographic plates, the quantum effiency is bound by a theoretical limit of about 0.5% 5%. By using semi conductor technology, we have managed to digitalize the process and developed what we call *Charge-coupled device (CCD)*. By taking advantage of the photoelectric effect, we have been able to calculate a pixel value by counting electrons being released. This technique has proven to increase the quantum efficiency tremendiously. Experiments shows the quantum efficiency to be as high as up to 95%. Fig.1 shows a schematic view of a CCD.

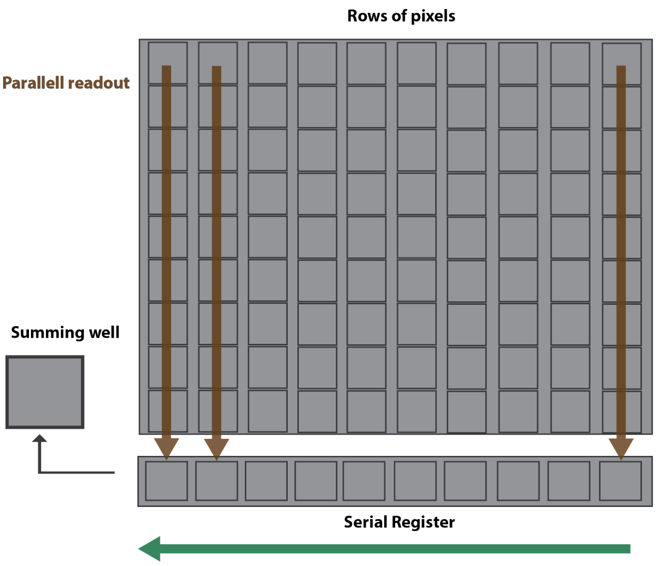

Fig. 1.— A schematic view of a CCD. Electrons released due to the photoelectric effect is counted which is used to calculate a pixel value.

Standard operations to process the "raw" image are corrections for *bias*, *dark current* and *flat field*.

Electronic address: hanspepg@student.matnat.uio.no
[1] Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, N-0315 Oslo, Norway

The bias level is measured in total darkness and with the shortest exposure time possible. The image obtained only contains unwanted signal due to the electronics that elaborate the sensor data, and not unwanted signal from charge accumulation (*dark current*) within the sensor itself. One would expect this level to correspond to a pixel value of 0 but in practice this level is set to a small value to account for digitization noise. In theory, the bias level should be identical for every pixel since no photoelectrons nor thermal electons are generated. However, in reality, teh bias level varies from pixel to pixel caused by various sources of noise.

A dark current is generated during integration due to generation of thermal electrons. The dark current is measured by blocking the light and exposuring with the same integration time as the raw image that is to be corrected.

Flat field correction compensated for sensitivity variations over the field of view.

bias and *dark current* are additive errors, while *flat field* is a multiplicative error meaning we need to subtract for *bias* and *dark current* and divide by the *flat frames*.

2. METHOD

2.1. *Camera properties*

First, we used a color Edmund Optics USB camera in a set-up with a white lamp, a thin singlet lens and a microscope objective (fig.2). We connected the camera to the computer and used a graphical user interface to adjust the image exposure time and to adjust the amount of light being detected. When changing the focus, we could see the effect of *chromatic aberration* (fig.3), which is when different wavelenghts will have slightly different focal point. To compensate for chromatic aberration we adjusted the objective location.

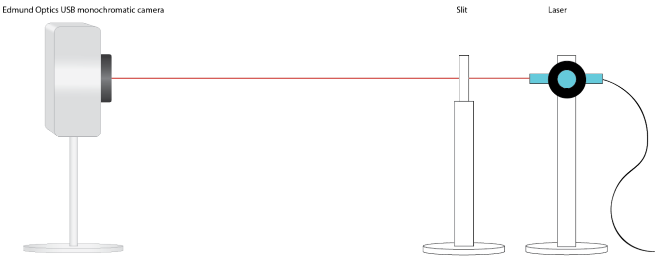

FIG. 2.— Edmund Optics USB camera in a set-up with a white lamp, a thin singlet lens and a microscope objective.

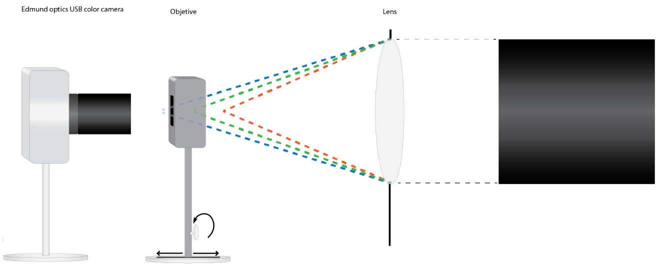

FIG. 3.— The figure shows the phenomena of chromatic aberration. Different wavelength will have a slight different focus point

Next we used a monochromatic Edmund Optics USB camera in a setup with a laser light and a $100\mu m$ slit. From the diffraction pattern in the live view of the camera, we measured the width of the pixels in μm, using the formula for single-slit diffraction

$$a \sin\theta = m\lambda \tag{1}$$

where a is the slit width, and m is the order of the minimum of the diffraction pattern. Using small angula approximation, we can write $sin\theta$ as $\frac{h}{L}$, where h is the distance to the first minima of the diffraction pattern and L is the distance from the slit to the camera. We used this to find an expression for h and calculated the pixel width by equation (3).

$$h = \frac{Lm\lambda}{a} \tag{2}$$

$$w_{px} = \frac{h}{N_{px}} \tag{3}$$

where N_{px} is the number of pixels between minima diveded by to. We used MATLAB to approixmate the number of pixels to the first minima in the diffraction pattern.

2.2. *Recording data; bias, dark frames, flat frames*

To record a proper image, one needs to make sure the image is well exposed. The signal level should be as high as possible in order to minimize noise but one should aoid over-exposure. One should avoid to get too close to the maximum exposure level in order to avoid non-linear behaviour of the sensor. We recorded a well exposed image of the diffraction pattern, writing down the exposure time, frame rate and pixel clock. We switched off the laser light and put the dust cover on the camera, removed the slit from the beam, then recorded the following images:

- 2 bias frames by turning down the exposure time to the minimum value
- 5 dark frams
- 1 dark frame at the maximum exposure time
- 16 flat frames
- 5 dark frames with same exposure time as the flat frames

Bias and dark frame was taken with the dust cover on. To record the flat field images, we used a simple A4-size white sheet of paper to reflect to reflect light from the ceiling into the camera. To make sure the image was well exposed, we adjusted the integration time so the average pixel value was between halfway to one-third to the maximum output of the camera.

2.3. *Image analysis*

For the image analysis, we used the interactive data language IDL. First, we loaded one bias frame and the dark frame with maximum exposurte time and compared the two images

- computed average pixel value of the two images.
- found minimum and maximum pixel values.
- ploted histograms of the pixel values
- located the pixel coordinate with the maximum value for both the bias and the dark frame.

Next, we loaded both bias frames, B_1 and B_2

- added them togetter and measured the mean value, $\bar{B}_1 + \bar{B}_2$ of the central region, which we defined to be region in the middle of the frames with each side 300 px wide/tall.
- subtracted one *bias frame* from the other and measured the standard deviation for the centeral region.

Next, we repeated the proces for the *flat frames*: computed the centeral region and the noise for the two flat frames.

Next, using equation (4), we computed the conversion factor g and used the result to compute the readout noise in electrons, which is the factor g times the standard eviation for the bias, $g\sigma_{bias}$

$$g = \frac{(\bar{F}_1 + \bar{F}_2) - (\bar{B}_1 + \bar{B}_2)}{(\sigma^2_{F_1-F_2}) - (\sigma^2_{B_1-B2})}[electrons/AU] \tag{4}$$

From the 16 flat frames we recored, we could see how the normalized noise decreased as more flat frames was added.

Finally, we corrected the diffraction image by first correcting from dark current, and then divide by the master normalized flat.

3. DATA

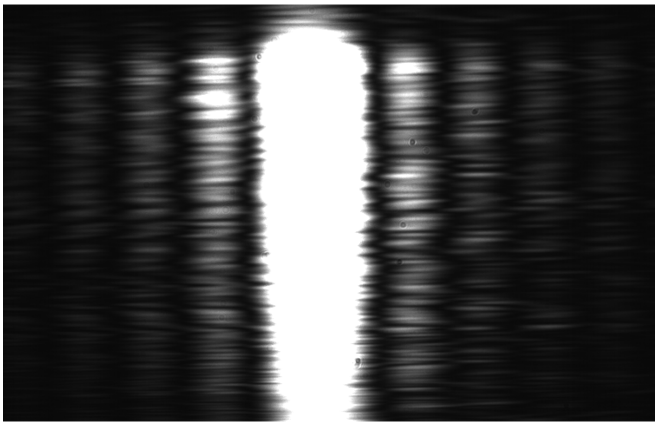

FIG. 4.— The figure shows the diffraction pattern where the exposure time has been adjusted for the best image.

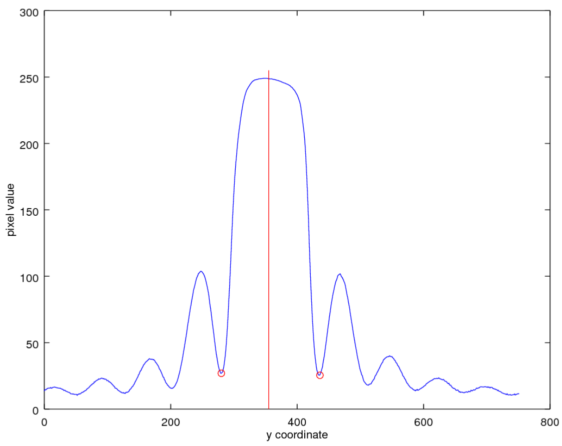

FIG. 5.— The figure shows the pixel positions of the two first locations of the minima of the interference pattern.

4. RESULTS

4.1. *Camera*

Fig. 6 shows the result of moving the objective location due to *chromatic aberratoin.*

By equations (1), (2) and (3) we found the width of the pixels to be $5.797435897 \pm 0.2439008863\mu m$

$$h = \frac{L\lambda}{a} \rightarrow \frac{7 \times 10^{-2} \times 645nm}{100\mu m} \quad (5)$$

$$w_{px} = \frac{h}{N_{px}} \rightarrow \frac{7 \times 10^{-2}m}{78} \quad (6)$$

$$\approx 5.797435897 \pm 0.2439008863\mu m \quad (7)$$

4.2. *Result of the recorded data*

Fig. 4 shows the recorded image of the diffraction pattern, table 1 shows the exposure time, frame rate and pixel clock.

4.3. *Image analysis*

Table 2 shows the averate pixel value, minimum and maximum pixel value of the two bias images and the dark frame with maximum exposure time. Fig. 7 shows histograms of the two images ploted in one figure. The pixel coordinate with the maximum value is the same for both the bias frame, and for the dark current. Table 1 shows the results.

Next, using equation (4), we computed the conversion factor g and used the result to compute the readout noise in electrons, which is the factor g times the standard eviation for the bias, $g\sigma_{bias}$

$$g = \frac{(\bar{F_1} + \bar{F_2}) - (\bar{B_1} + \bar{B_2})}{(\sigma^2_{F_1 - F_2}) - (\sigma^2_{B_1 - B2})}[electrons/AU] \quad (8)$$

Fig.8 shows the reduction in noise as more flat frames was added.

Fig.9 shows the corrected image from first correcting from dark current, then divide by the master normalized flats.

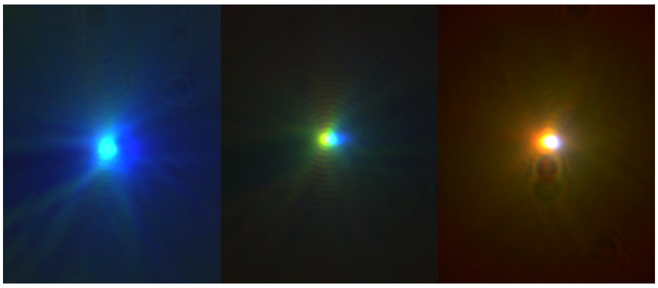

FIG. 6.— The figure shows the effect from chromatic aberration by changing the location of the objective. The three images are three different snapshots done after adjusting the objective location.

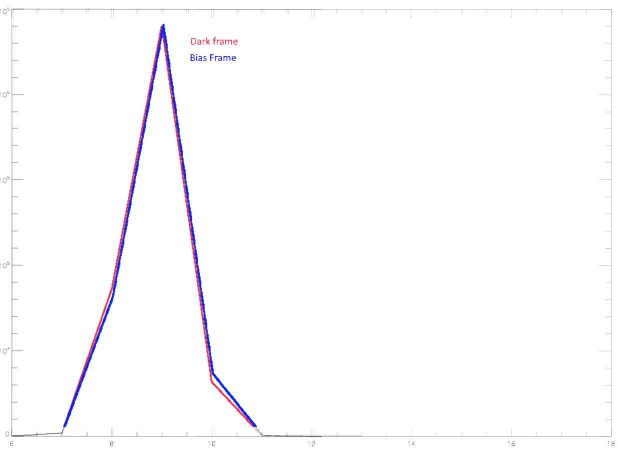

FIG. 7.— The figure shows the histograms for bias frame two and the dark frame with the highest exposure time.

TABLE 1

Image(s)	Pixel clock	Framerate	Exposure time
Diffraction image 1	$33Mhz$	$90FPS$	$0.097ms$
Diffraction image 2	$40Mhz$	$90FPS$	$1.164ms$
Flat frames	$5Mhz$	$90FPS$	$13.474ms$

Note. — Settings for recorded images

TABLE 2

Image	Average	Min	Max	Location of max pixel
BIAS frame	8.8225482	6.0	17.0	(427,40)
Dark frame	8.8420684	6.0	18.0	(427,40)

Note. — Estimate from bias frame 1 and the dark frame with the highest exposure time

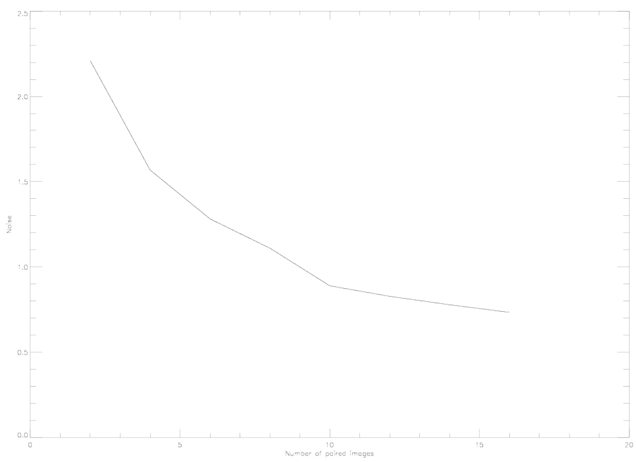

Fig. 8.— The figure shows the rerudction in noise.

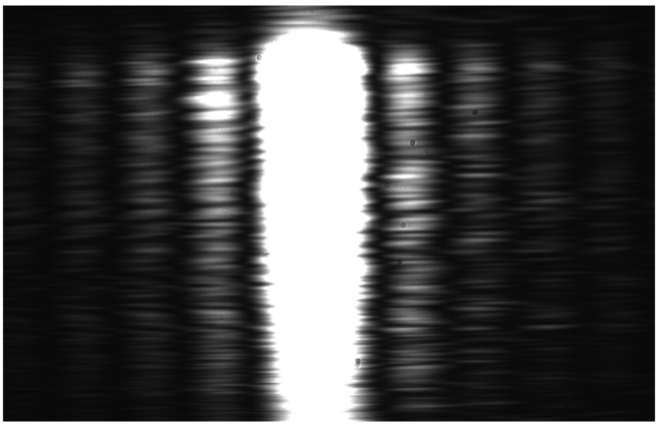

Fig. 9.— The figure shows the corrected image of the diffraction pattern.

All data used in the experiment can be found on GitHub `https://github.com/Spillerom/AST2210/tree/master/ccd`.

5. CONCLUSIONS

Digitalizing the proccess of regording an image using *Charge-couple devices (CCD's)* shows promesing results with a quantum efficiency of up to 95%.

By examining plots of the noise in the image before and after the image has been corrected shows noise is beeing reduced. However, what seems to be imperfections in the optical path did not disappear in the corrected image. The conclution must be that either the error does not come from irriggularities on the lens, but occured earlier on the path or that the recorded flat frames was over exposted. In both cases, flat frame correction will not have any effect and cannot be used to correct the image.

For short exposure times, noise due to *dark current* does not seem to have large implications on the recorded image, but as the integration time increases, noise will increase due to *dark current*. Experiments needs to be done in order to see the effect from this.

Compared to photographic plates, using *Charge-couple device (CCD)* to record images improve image quality tremendiously. Noise due to *bias*, *dark current* and *flat fields* do occur, however, using the techniques described above will reduce the amount noise.

I want to thank Aynar Drews for guiding us through the experiment as well as helping out with IDL and my lab partners Markus Bjrklund and Andreas Helland for constructive discusions. Also a special thank to Andreas Helland for letting me use all his *beatiful* figures.

REFERENCES

[1]AST2210 - Lab exercise: CCD, `http://folk.uio.no/ainard/AST2210/CCDLab/ccd_lab.pdf`

APPENDIX

CODE

Listing 1: Code for calculating the pixel position of the minima of the diffraction pattern

```

A = imread('interference_1.png');

[h, w] = size(A)

x = linspace(0, w-1, w);
y = sum(A)./h;

center = 355

minima_1 = 1000000000;
minima_xpos_1 = 0;

minima_2 = 1000000000;
minima_xpos_2 = 0;

for i=(center-100):(center)
    if y(i) < minima_1
        minima_1 = y(i);
        minima_xpos_1 = i
    end
end

for i=(center):(center+100)
    if y(i) < minima_2
        minima_2 = y(i);
        minima_xpos_2 = i
    end
end

distance_to_minima = (minima_xpos_2-minima_xpos_1)/2

%imshow(A)
%hold on
%plot([center center], [0 h], '-r')

plot(x, y)
hold on
plot(minima_xpos_1, minima_1, '-ro')
plot(minima_xpos_2, minima_2, '-ro')
plot([center center], [0 255], '-r')
xlabel('y coordinate')
ylabel('pixel value')


pause()
```

Listing 2: Code for computing one average, min and max pixel values and plot histograms of pixel values for one bias frame and one dark frame.

```



bf2 = double(read_bmp('bf2.bmp'))

print, 'BIAS frame 2, average', avg(bf2)
print, 'BIAS frame 2, min', min(bf2)
print, 'BIAS frame 2, max', max(bf2, location)

max_index = array_indices(bf2, location)
print, max_index

hist_bf2 = histogram(bf2, locations = values)
window, 0
plot, values, hist_bf2

window, 1
plot_image, bf2

df1_2 = double(read_bmp('df1_2.bmp'))

```

```
print, 'Dark␣frame␣(1_2),␣average', avg(df1_2)
print, 'Dark␣frame␣(1_2),␣min', min(df1_2)
print, 'Dark␣frame␣(1_2),␣max', max(df1_2, location)

max_index = array_indices(df1_2, location)
print, max_index

hist_df1_2 = histogram(df1_2, locations = values)
window, 2
plot, values, hist_df1_2

window, 3
plot_image, df1_2

;save_plots, 'new_test_file', 'eps', df1_2, xtitle='Test'


end
```

Listing 3: Code for loading two bias frams. Adding them togetter and measure the mean value of the centeral region.

```


bf1 = double(read_bmp('bf1.bmp'))
bf2 = double(read_bmp('bf2.bmp'))

bias_add = bf1 + bf2

print, 'Average␣avg()', avg(bias_add[226:526,90:390])

bias_sub = bf1 - bf2
print, 'Standard␣deviation␣of␣(bf1␣-␣bf2)', stddev(bias_sub[226:526,90:390])

print, 'sqrt(2)*stddev(bf1)', sqrt(2)*stddev(bf1[226:526, 90:390])


end
```

Listing 4: Code for loading two flat frames. Adding them togetter and measure the mean value of the centeral regian.

```


ff1 = double(read_bmp('ff1.bmp'))
ff2 = double(read_bmp('ff2.bmp'))

ff_add = ff1 + ff2

print, 'Average␣avg(ff_add)', avg(ff_add[226:526,90:390])
print, 'Average␣avg(ff1)␣+␣avg(ff2)', avg(ff1[226:526,90:390]) + avg(ff2[226:526,90:390])

ff_sub = ff1 - ff2
print, 'Standard␣deviation␣of␣(ff1␣-␣ff2)', stddev(ff_sub[226:526,90:390])

print, 'sqrt(2)*stddev(ff1)', sqrt(2)*stddev(ff1[226:526, 90:390])


end
```

Listing 5: Code for computing the conversation fagtor g.

```

bf1 = double(read_bmp('bf1.bmp'))
bf2 = double(read_bmp('bf2.bmp'))

bias_add = bf1 + bf2
bias_sub = bf1 - bf2

ff1 = double(read_bmp('ff1.bmp'))
ff2 = double(read_bmp('ff2.bmp'))

ff_add = ff1 + ff2
ff_sub = ff1 - ff2

b = (mean(ff_add[226:526,90:390]) - mean(bias_add[226:526,90:390]))/(stddev(ff_sub[226:526,90:390])^2 - stddev(bias_sub[226:526,90:390])^2)

print, 'b␣=␣', b

```

```
end
```

Listing 6: Code for computing the readout noise in electrons.

```

bf1 = double(read_bmp('bf1.bmp'))
bf2 = double(read_bmp('bf2.bmp'))

bias_add = bf1 + bf2
bias_sub = bf1 - bf2

ff1 = double(read_bmp('ff1.bmp'))
ff2 = double(read_bmp('ff2.bmp'))

ff_add = ff1 + ff2
ff_sub = ff1 - ff2

b = (avg(ff_add[226:526,90:390]) - avg(bias_add[226:526,90:390]))/(stddev(ff_sub[226:526,90:390]) -
    stddev(bias_sub[226:526,90:390]))

print, 'b = ', b

ro = g*stddev(bf1[226:526,90:390])

print, 'Readout noise in electrons', ro



end
```

Listing 7: Code for computing normalized noise from 16 flats.

```

cgCleanUp

; Load files
ff = list()
for i=0,15 do begin
    filename = 'ff' + strtrim(i+1,2) + '.bmp'

    image = double(read_bmp(filename))
    ff.add, image[226:526, 90:390]
endfor

; Calculate sigma
sigma = list()
sigma.add, stddev(ff[0] - ff[1])
sigma.add, stddev((ff[0]+ff[2]) - (ff[1]+ff[3]))
sigma.add, stddev((ff[0]+ff[2]+ff[4]) - (ff[1]+ff[3]+ff[5]))
sigma.add, stddev((ff[0]+ff[2]+ff[4]+ff[6]) - (ff[1]+ff[3]+ff[5]+ff[7]))
sigma.add, stddev((ff[0]+ff[2]+ff[4]+ff[6]+ff[8]) - (ff[1]+ff[3]+ff[5]+ff[7]+ff[9]))
sigma.add, stddev((ff[0]+ff[2]+ff[4]+ff[6]+ff[8]+ff[10]) - (ff[1]+ff[3]+ff[5]+ff[7]+ff[9]+ff[11]))
sigma.add, stddev((ff[0]+ff[2]+ff[4]+ff[6]+ff[8]+ff[10]+ff[12]) - (ff[1]+ff[3]+ff[5]+ff[7]+ff[9]+ff
    [11]+ff[13]))
sigma.add, stddev((ff[0]+ff[2]+ff[4]+ff[6]+ff[8]+ff[10]+ff[12]+ff[14]) - (ff[1]+ff[3]+ff[5]+ff[7]+ff
    [9]+ff[11]+ff[13]+ff[15]))

; or, if you want:

;t1 = ff[0]
;t2 = ff[1]
;sigma_1 = list()
;k = 0
;for i= 1,7 do begin
;    sigma_1.add, stddev(t1 - t2)
;    t1 = t1 + ff[2*i]
;    t2 = t2 + ff[(2*i)+1]
;endfor
;sigma_1.add, stddev(t1 - t2)

;help, ff
;plot_image, ff[0]

;for i= 0,7 do begin
;    print, sigma[i]
;    print, sigma_1[i]
```

```
;     print, '----'
;endfor

; Calculate the normalized values:
print, 'Normalized_values'
sigma_norm = list()
for i= 0,7 do begin
    sigma_norm.add, sigma[i]/((i+1))
    print, sigma[i]
    print, sigma_norm[i]
    print, '----'
endfor

x = [2,4,6,8,10,12,14,16]
plot, x, sigma_norm.toarray()

save_plots, 'decreasing_normalized_noise','eps', x, sigma_norm.toarray(), xtitle='Number_of_paired_images', ytitle='Noise'

end
```

Listing 8: Code for correcting the raw diffracton pattern image.

```

; Load original image
diff= double(read_bmp('interference_1.bmp'))
window, 0
plot_image, diff

; Load flats
ff = list()
for i=0,15 do begin
    filename = 'ff' + strtrim(i+1,2) + '.bmp'

    image = double(read_bmp(filename))
    ;ff.add, image[226:526, 90:390]
    ff.add, image
endfor
F_avg = (ff[0]+ff[1]+ff[2]+ff[3]+ff[4]+ff[5]+ff[6]+ff[7]+ff[8]+ff[9]+ff[10]+ff[11]+ff[12]+ff[13]+ff[14]+ff[15])/16.0
window, 1
plot_image, F_avg

; Load darks
ff_df = list()
for i=0,4 do begin
    filename = 'ff_df_' + strtrim(i+1,2) + '.bmp'

    image = double(read_bmp(filename))
    ;ff_df.add, image[226:526, 90:390]
    ff_df.add, image
endfor
D_avg = (ff_df[0]+ff_df[1]+ff_df[2] + ff_df[3] + ff_df[4])/5.0
window, 2
plot_image, D_avg


; Load darks
df_raw = list()
for i=0,4 do begin
    filename = 'df' + strtrim(i+1,2) + '_1.bmp'

    image = double(read_bmp(filename))
    ;df_raw.add, image[226:526, 90:390]
    df_raw.add, image
endfor
D_irawavg = (df_raw[0]+df_raw[1]+df_raw[2] + df_raw[3] + df_raw[4])/5.0
window, 3
plot_image, D_irawavg

F_master = F_avg - D_avg

window, 4
plot_image, F_master

F_normmaster = F_master/mean(F_master)

```

```
window, 5
plot_image, F_normmaster


;I_raw = diff[226:526, 90:390]
I_raw = diff

I_corr = (I_raw - D_irawavg) / F_normmaster

window, 6
plot_image, I_corr

save_img, I_corr, 'corrected_image', type='eps', color_table = 2, title='Corrected diffraction image'

end
```