Imaging Instrumentation Laboratory #1

Basic Camera Characterization

Katie Ceraso, Xihan Liu, Aurelius Jing

Introduction

Since the 21st century, generating film then digital images has been possible thanks to image sensors (York, 2011). The most commonly used image sensors are those using Charge Coupled Device (CCD) technology and Complementary Metal Oxide Semiconductor (CMOS) sensors. Compared with CCD, CMOS has many advantages, such as lower power consumption, voltage operation, on-chip functionality, and lower cost (Bigas, 2006). However, the main limitation to CMOS sensors is the poor spatial resolution compared to CCD sensors. This report aims to characterize primary camera responses, using a CMOS sensor, sensitivity to exposure times, and strategies for optimal acquisition (Stayman, 2023). There are four parts to this experiment. The first part of the experimental procedure focuses on characterizing detector offset as a function of exposure time, and determining readout noise as a function of exposure time for dark images. Also, the camera linearity is investigated by using the uniform scene and varying exposure time. The second part focuses on the mean offset values, noise and signal-to-noise ratio (SNR) as a function of exposure time for various light field images, along with determining the optimal imaging settings. The third part focuses on a design to create an auto-exposure tool. Finally, the last part of the report shows histograms of the flat-signal data for different exposures (including very low exposures) and corresponding discussion.

Method

The materials used for this lab are as follows:

- x1 CMOS Camera and Lens
- x1 Forked Base Clamp
- x1 Imaging Plate/Backboard
- x2 Vertical Holders
- x3 Tie down screws/washers
- x1 Optical Bench
- x1 Laptop with USB 3.0 port, Python, ThorLabs

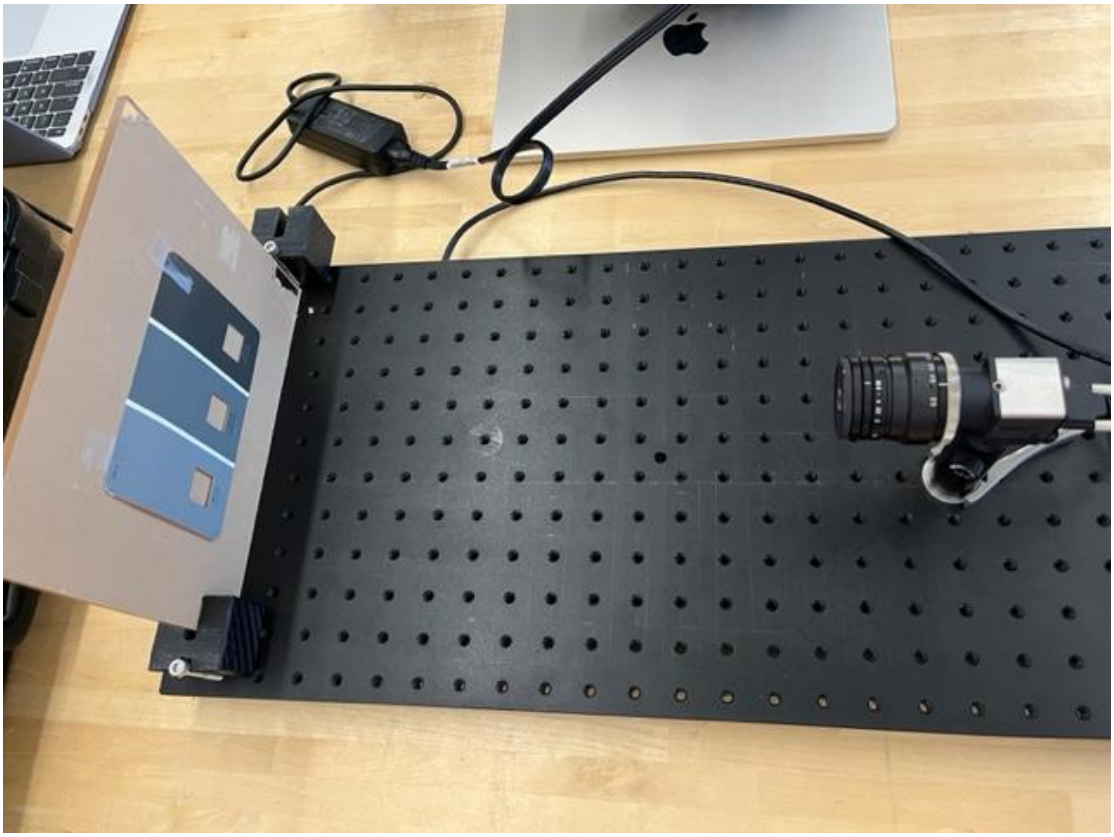


Figure 1: Optical Bench Set-Up for Experiments 1-4

For the three parts of this experiment, the optical bench has been set up as shown in Figure 1.

Experiment 1: Characterizing Detector Mean Offset and Noise for No-Signal Images

For experiment 1, the lens cap was kept on the CMOS camera allowing it to collect dark no-signal images. The camera settings are: aperature/focus settings=N/A, pixel clock = 7 MHz.

First, 50 frames of full-field frame-averaged offset images were taken at the relatively high exposure time of 120 ms. From these 50 frames, a noise image, calculated as the image variance, was also produced. The purpose of the averaged frames is to reduce the effect of system noise between the images collected in the experiment. Second, 100 frames of 100 x 100 pixel AOIs were taken at 17 different exposure times across the achievable range (10, 15, 20, 35, 40, 50, 60, 75, 80, 90, 100, 110, 115, 120, 125, 130, 140 ms). The 100 frames were again averaged to reduce the effect of system noise between images taken. These exposures were deemed achievable from the *cam.get_exposure_range()* function in python. These AOI images were used to determine the relationship between exposure time and mean offset value, along with the relationship between exposure time and noise, for dark field images. In other words, they represent the noise inherent to the camera system from the dark current.

Experiment 2: Characterizing Detector Mean Offset, Noise, and Signal-Noise Ratio for Light Signal Images

For experiment 2, the lens cap is taken off, and signals are received by the CMOS sensor of the graded uniform paint swatch taped to the blackboard. The aperture = 6 and the focus = 0.37 m. These values are determined through live-imaging a section of the paint swatch with small details and choosing aperture and focus values that return a clear image with no blur that capture the full spectrum of pixel values (0-255). The pixel clock has been set up to 7 megahertz. The exposure is once again set to the relatively high value of 120 ms. A full-field image of the three uniform regions of the paint swatch is visible in the field of view. First, the image averaging process from Experiment 1 is repeated with the cap off, with 50 frames taken and averaged to produce an average offset value image and a noise image. A flat region is defined here as visually

constant pixel light values across the AOI of an image. Second, for each of three different shades of gray on the paint swatch, the mean and variance in the flat region will be calculated with 100 frames with 100 x 100 pixel AOIs are taken at 17 exposure times [10, 15, 20, 35, 40, 50, 60, 75, 80, 90, 100, 110, 115, 120, 125, 130, 140] ms. These images were used to calculate the mean offset value, the noise, and the signal-to-noise ratio (SNR) as functions of exposure time.

With the aperture set to 6, focus setting of 0.37m and exposure at 120 ms, the best image quality possible has been found. The SNR is known as the metric of the best image quality. A strategy to determine that all three regions have a maximum SNR involved first manually setting the aperture and focus values while live imaging the paint swatches so they returned clear images with no blur and enough light to distinguish between the three shades of gray. Then, the frames from each of the three AOIs were imaged at various exposures, and an SNR was calculated for each AOI. The exposure with the highest relative SNR across all three AOIs was chosen as part of the camera set up for the largest SNR. This strategy was the inspiration for our auto-exposure tool. A similar strategy can be utilized to determine the highest SNR for only the darkest region of interest. All the same steps were followed, except only the dark AOI has images taken at various exposure times, and the largest SNR for that region only is selected.

Experiment 3: Development of Auto-Exposure Tool in Python

For experiment 3, a tool that can quickly find the optimal exposure for the scene shown in experiment 2 has been created. The general functionality of this tool involves calculating the SNR for the image at various exposure times. As will be discussed later, the graph result of SNR in three AOIs in part 2 demonstrate that the SNR would increase exponentially and saturate. Thus,

we will calculate the first derivative, and when the first derivative approaches or is equal to 0, it will signify that the exposure level fits the current scene.

We did two trials with different settings. In trial 1, data were collected using the following camera settings: aperture = 6, focus = 0.37m, frame rate = 7Hz, exposure range = 0ms -142ms, Pixel Clock = 7 MHz, Gain = 1.92, Image Size = 1024x1280. In trial 2: aperture = 7, focus = 0.35m, frame rate = 5Hz, exposure range = 0ms - 199ms, Pixel Clock = 7MHz, Gain = 1.92, Image Size = 768x1024. These values were chosen because at varied exposures, they had yielded high spatial resolution images.

Experiment 4: Creating Histograms for flat-signal data at different exposures

For experiment 4, we will plot the histograms of images at 13 different exposure times. We count the number of pixels of 1024x1280-sized images with intensities varying from 0-255. The images used in this experiment had the following settings: frame rate = 7Hz, exposure range = 0ms -142ms, Pixel Clock = 7 MHz, Gain = 1.92, Image Size = 1024x1280.

The Results and Discussion section will show the data and design for all three parts.

Results

Detailed below are the findings from the experiments outlined above. Key findings include the positive correlation between noise and exposure time for both no-signal and signal images. One exception to this finding was the relationship between the mean offset value and exposure time for the dark images in Experiment 1. Additionally, the noise in the dark images is significantly lower than in the light images. Furthermore, the functionality of the auto-exposure tool is detailed below and proved effective at finding an ideal exposure for a given imaging setup.

Lastly, the histograms from experiment 4 demonstrated that as exposure time increased, there was a larger number of pixels at higher intensities.

Experiment 1: Characterizing Detector Mean Offset and Noise for No-Signal Images

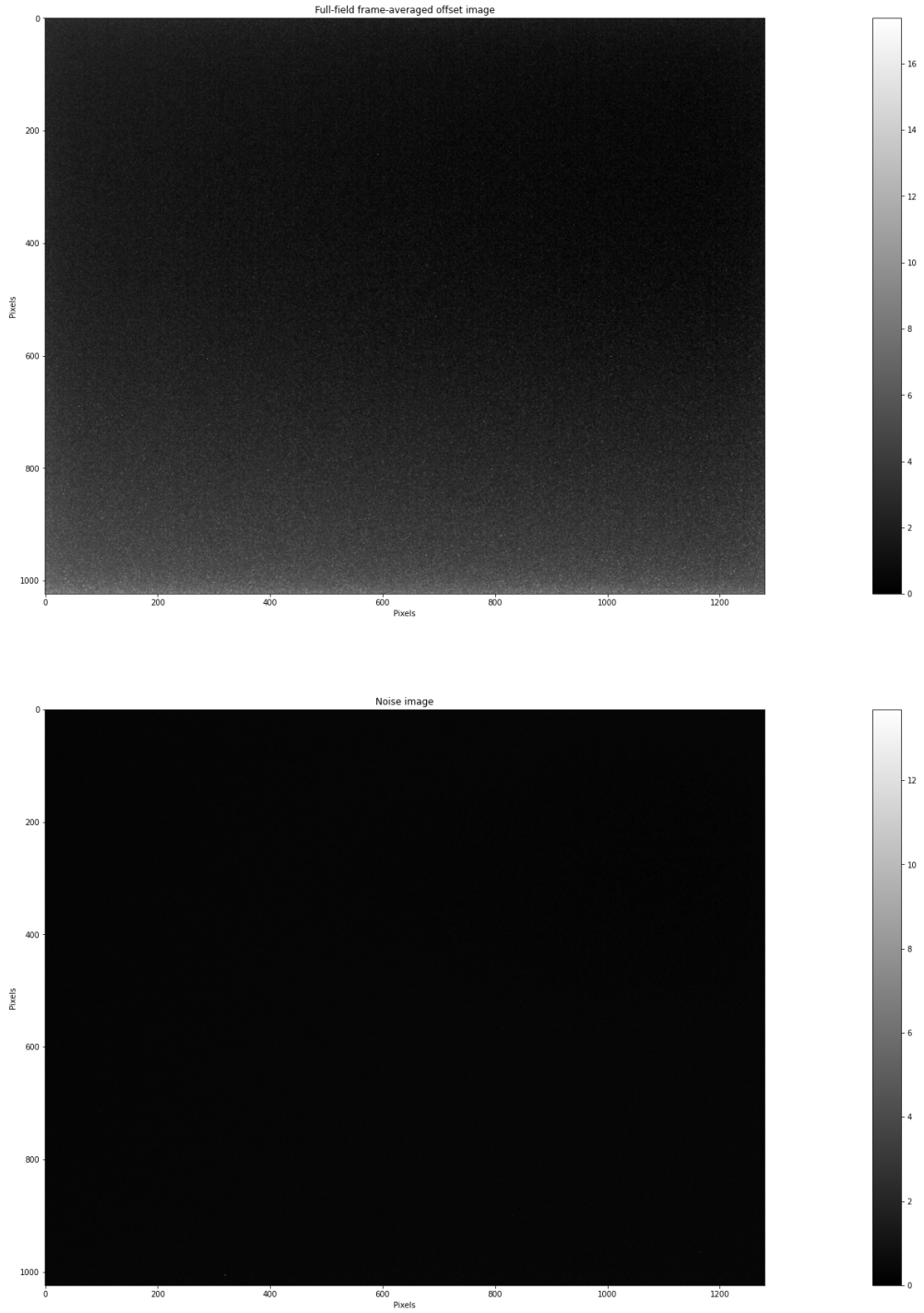


Figure 2A. Full-field frame-averaged and noise-no-light images for 50 frames. (Images were taken with a DCC3240M CMOS camera.)

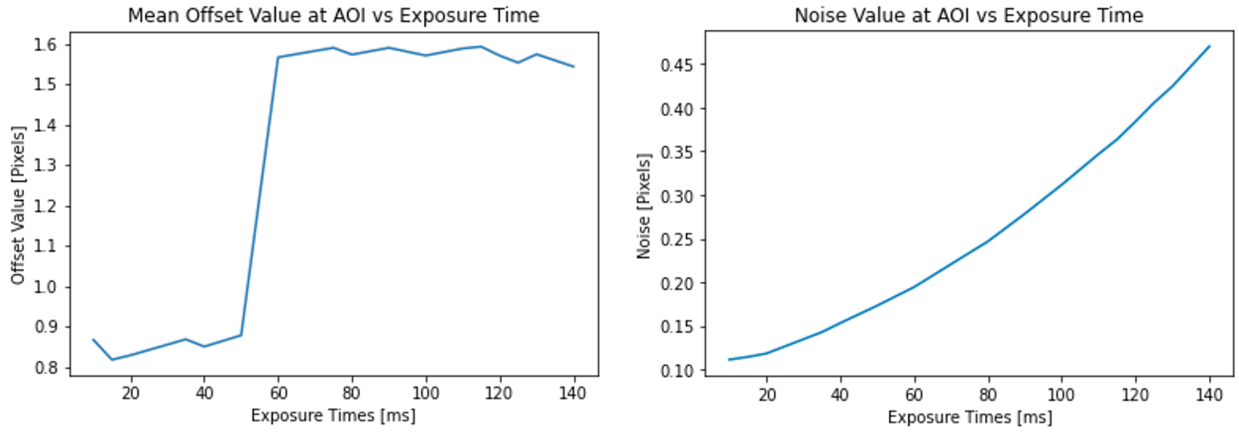


Figure 2B. Mean Offset Values and Noise for Dark Images

Left: Line graph of mean offset values versus exposure time from 100 frames of dark images taken at 17 different exposures. Right: Line graph of noise values, represented by variance, versus exposure time from 100 frames of dark images taken at 17 different exposures.

Figures 2A and 2B show the results from Experiment 1. The full-field frame-averaged and noise images from the 50 frames with no signal can be seen in Figure 2A. The averaged photo is mainly dark, with some lighter pixels along the bottom of the frame. The noise image is largely uniform with mostly dark values, representing a noise variance of 0. There are some "bad" pixels that are difficult to see with the naked eye but are evidenced by the color bar going up to 12 for white values. The left image in Figure 2B depicts the relationship between exposure time and mean offset value for the dark images. This relationship begins positively linear, then shows a sharp increase in offset value around an exposure of 50. The mean offset value then remains relatively constant. The relationship between the mean offset value and exposure times for the dark image should be linearly increased theoretically, but in our data is difficult to define: the longer the exposure time, the larger the offset values. The right image in Figure 2B represents the noise versus exposure time, and an approximately positive linear relationship between the two variables can be seen in the graph.

Experiment 2: Characterizing Detector Mean Offset, Noise, and Signal-Noise Ratio for Light Signal Images

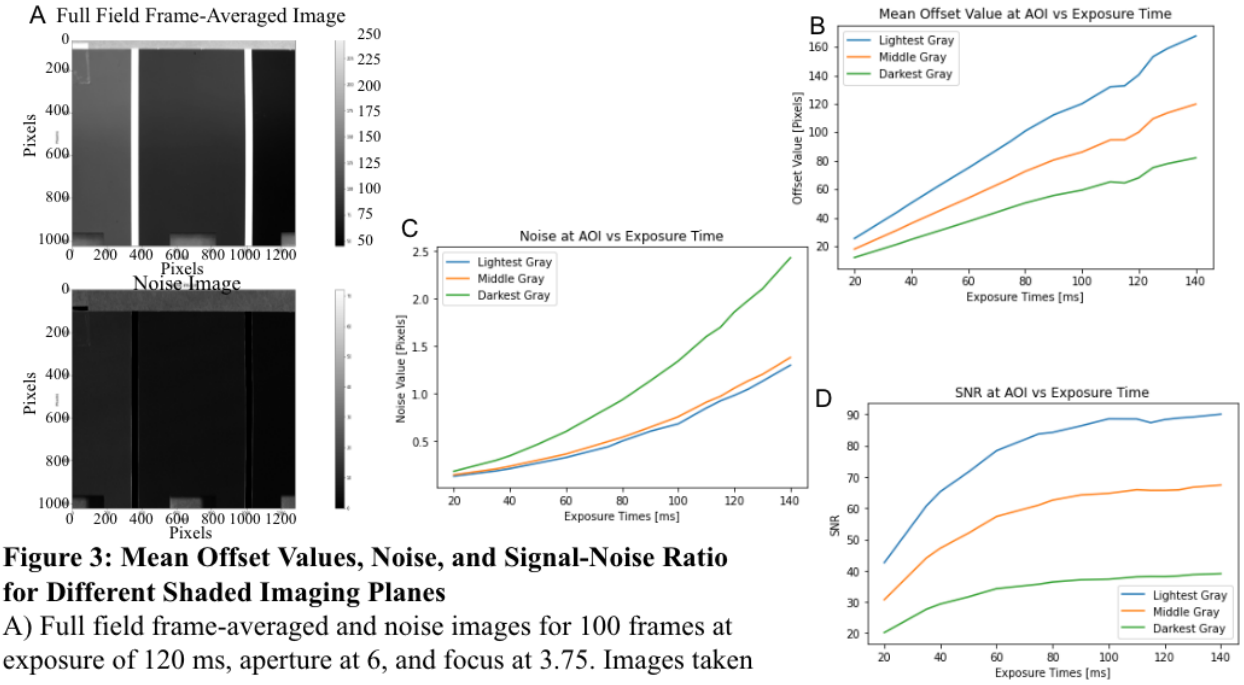


Figure 3: Mean Offset Values, Noise, and Signal-Noise Ratio for Different Shaded Imaging Planes
A) Full field frame-averaged and noise images for 100 frames at exposure of 120 ms, aperture at 6, and focus at 3.75. Images taken with a DCC3240M CMOS camera. B) Line graph of mean offset values versus exposure time taken at 17 different exposures. Images include three Areas of Interest (AOIs) of three uniform shades. C) Line graph of noise values, represented by variance, versus exposure time taken at 17 different exposures. Images include three AOIs of three uniform shades. D) Line graph of Signal-Noise Ratio (SNR) values versus exposure time taken at 17 different exposures. Images include three AOIs of three uniform shades.

Figure 3 illustrates the results from Experiment 2. The full-field frame-averaged and noise images of the paint swatch from the 50 frames can be seen in Figure 3A. Here, the full-field image has a high spatial resolution, and the three shades of gray regions are distinct. Additionally, the 100x100 pixel AOIs chosen were at (200, 200), (450, 200), and (1100, 200) and visually can be identified as flat regions with consistent pixel values. Figure 3B shows the positive linear relationship between mean offset values and exposure time for the light images. Additionally, the lightest shade of gray AOI has the highest mean offset values, with the values decreasing as the AOIs are darker shades. Figure 3C shows a positive correlation between exposure time noise values, and the highest noise values are at the lightest gray AOI, with values dropping for the

darker grays. Figure 3D depicts a positive correlation between exposure time and SNR, with the SNR values plateauing at an exposure time of around 75 ms. Again, the highest SNR values belong to the lightest gray AOI.

Experiment 3: Development of Auto-Exposure Tool in Python

Trial 1:

The auto-exposure tool coded in python takes 20 frames of 1024x1280 pixels image at the following exposures [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130] ms. These 20 frames at each exposure time are averaged, and the SNR is calculated. Then we calculated the first derivative of the SNR line. The derivative of the highest SNR should be close to or equal to 0 as a local maximum. We record the exposure level correlated with the maximum SNR value, and send the value to the camera.

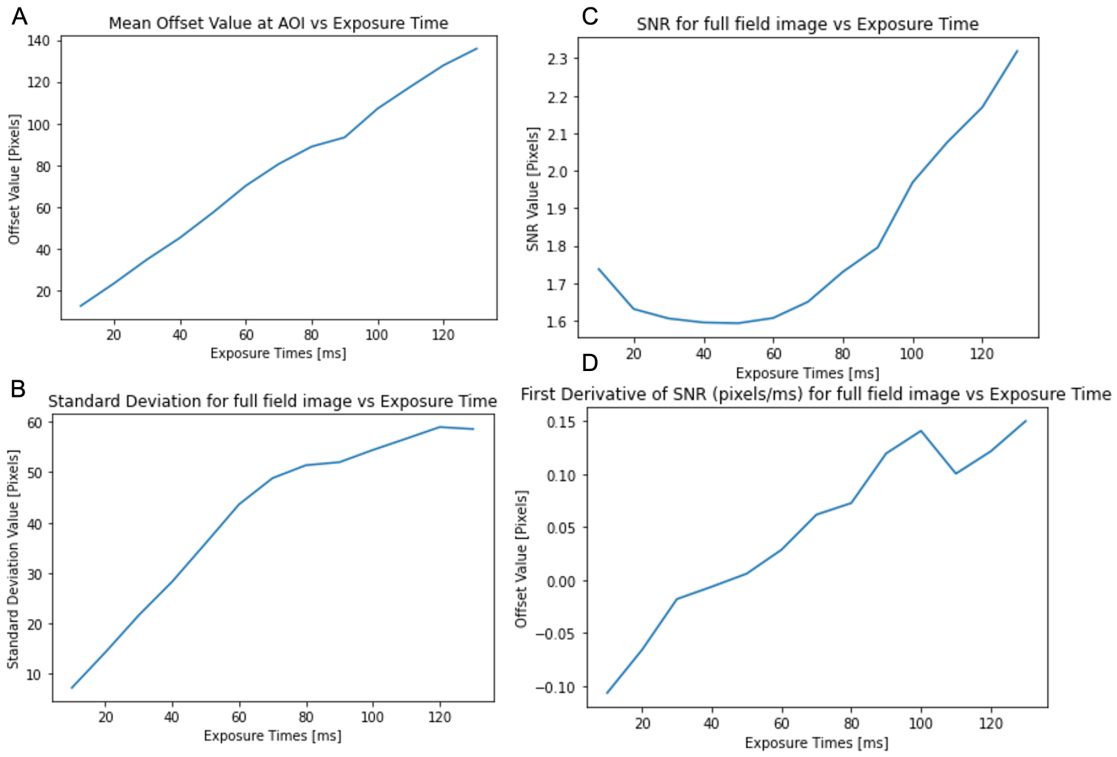


Figure 4. Mean offset values, noise, SNR, and First Derivative of SNR Line Graphs with averaged 20 frames of real-scene full-field images taken at 13 different exposures.

A) Line graph of mean offset values versus different exposure times. B) Line graph of noise values, represented by variance versus different exposure times. C) Line graph of SNR values versus different exposure times. D) Line graph of First Derivative of SNR values versus different exposure times.

From Figures 4A and 4B, we analyzed that the mean offset values and noise have linear relationships with exposure times, consistent with the results in Part 2 Experiments, which demonstrates that our mean offset values and noise are collected correctly. Figure 4C shows a decreasing trend from 10ms to 60ms and an increasing trend from 60ms to 130ms, with Figure 4D showing negative values from 10ms to 60ms and positive values after 60 ms. We did not detect a local maximum from the graph, so we performed the experiment again as Trial 2.

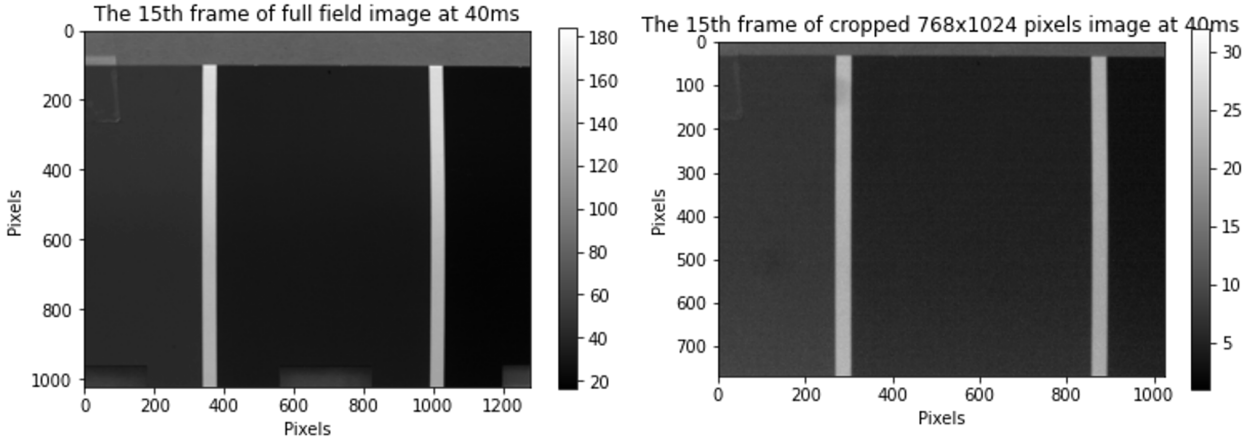


Figure 5. Full-field Image and Cropped-size image

Left: Full-field image (1024x1280) used in Trial 1. Right: Cropped-size image (768x1024) used in Trial 2.

Trial 2:

In trial 2, the tool takes 20 frames of cropped centered images (768x1024 pixels) from full-field images at exposures of [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190] ms due to limited memory storage. As Figure 5 demonstrated, we cropped the

image with coordinates [100:868,100:1124], which loses some boundary information due to memory limitations. Still, we captured the significant features in the cropped image compared to the full-field image. These 20 frames at each exposure time are averaged similarly to trial 1, and the SNR is calculated. Then, we calculated the first derivative of SNR to get the local maximum.

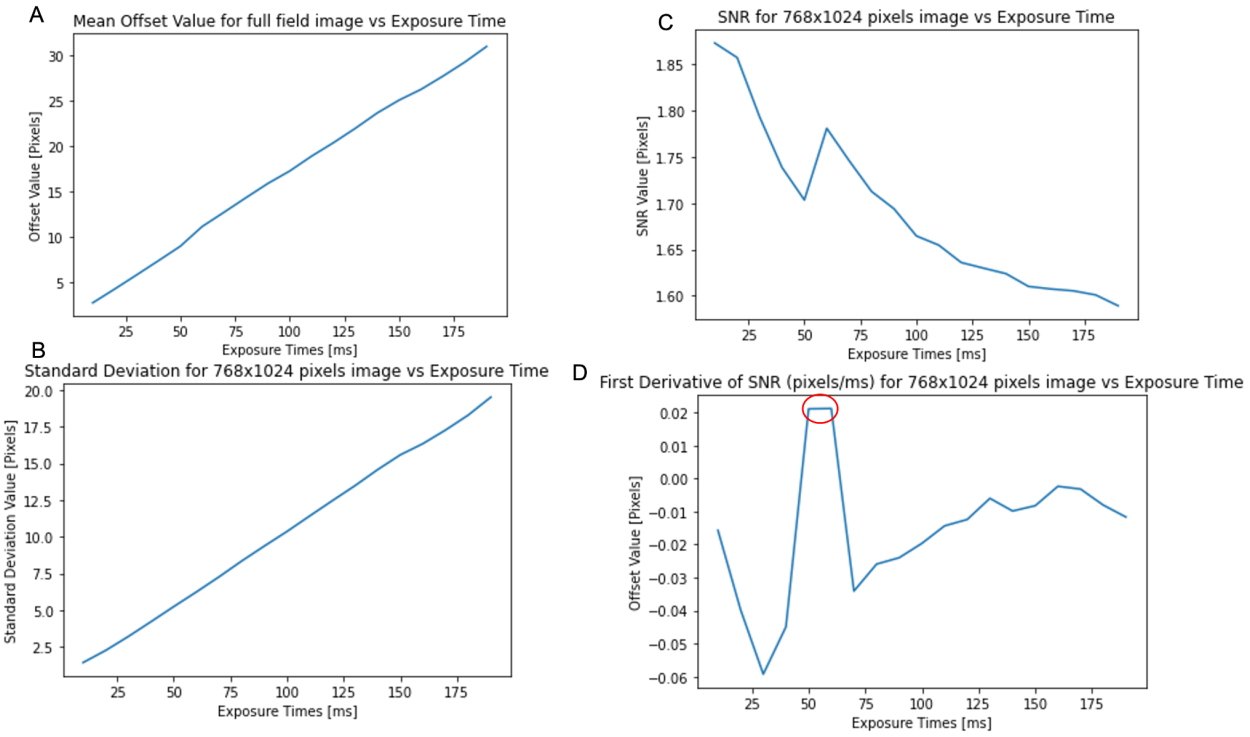


Figure 6. Mean offset values, noise, SNR, and First Derivative of SNR Line Graphs from averaged 20 frames of real-scene cropped size images taken at 13 different exposures.

A) Line graph of mean offset values versus different exposure times. B) Line graph of noise values, represented by variance, versus different exposure times. C) Line graph of SNR values versus different exposure times. D) Line graph of First Derivative of SNR values versus different exposure times. The ideal exposure time is circled in red.

Similarly, from Figures 6A and 6B, we plotted the line graph of the mean offset values and noise versus different exposure times to prove that our mean offset values and noise were collected correctly. Figure 4C shows a major decreasing trend from 10ms to 190ms and a peak at about 55ms, with Figure 5D showing gradients of SNR in Figure 4. An important thing to be

noticed in Figure 6D is an increasing trend from 25ms to 50ms and a decreasing trend from 70ms to 75ms, with a flat region marked in red from 50ms to 60 ms, demonstrating that we found a local maximum in SNR correctly. All in all, the ideal exposure time from the auto-exposure tool is around 60ms.

Experiment 4: Creating Histograms for flat-signal data at different exposures

We analyzed the histograms of our full-field images collected in Part 3, Trial 1. Figure 7 illustrates 9 of 13 different exposure times over the achievable range, from very low exposures to longer exposures. When the exposure is at 10ms, 20ms, and 30ms, most pixel values are at 10, 15, and 20, respectively; when the exposure is at 70ms, 80ms, and 90ms, most pixel values are at 55, 60, and 65, respectively; when the exposure is at 110ms, 120ms, and 130ms, the distribution of pixel values are more uniform compared to the previous bins, and most pixel values are at 75, 80, and 95, respectively. We also observed some extremely high pixel values(outliers/noise) at 255 starting from 70ms and ranging to 130ms. As a result, we can conclude that at low exposure times, the majority of the pixels have low intensity values, and as the exposure time increases, the pixel values will shift from lower to higher intensities.

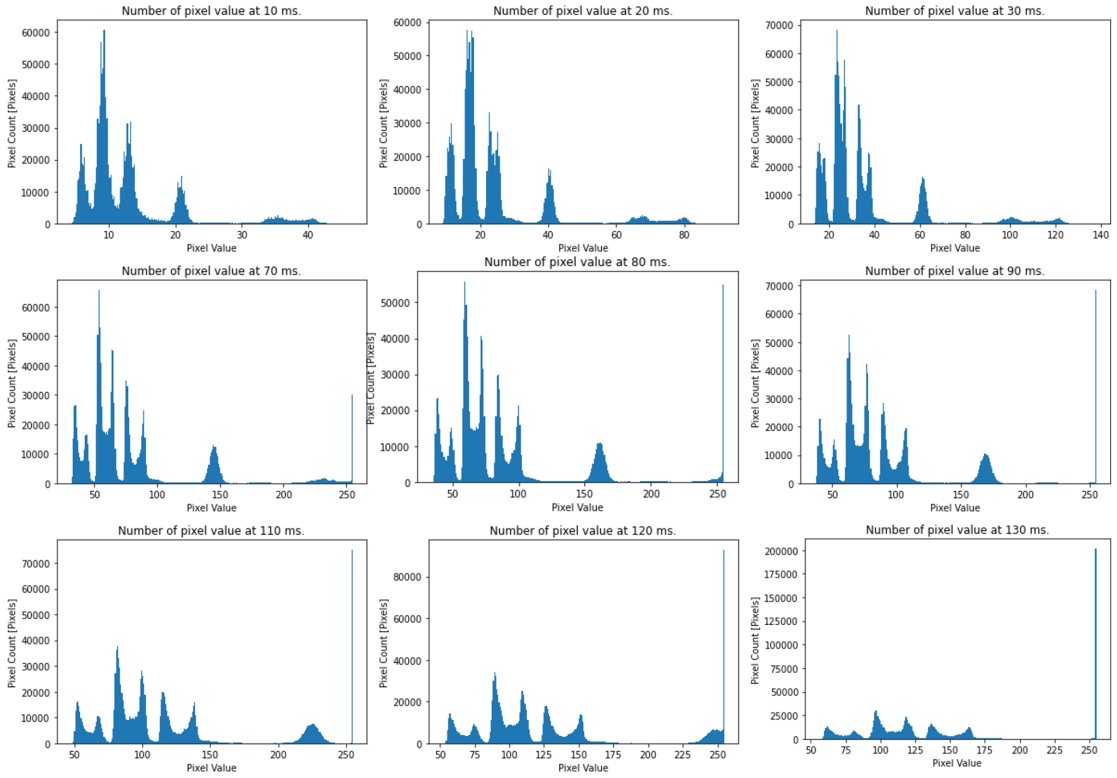


Figure 7. Group of Histograms for pixel count versus grayscale image intensity(0-255) at [10, 20,30,70,80,90,110, 120, 130]ms.

Discussion

CMOS sensors exhibit non-zero measurements even when no signal is presented to the system. This non-zero "offset" value may come from multiple sources and depend on a particular camera setting. The leading cause is dark current caused by rogue electrons flowing across the photodiode even when no voltage is applied, which is caused by photons contacting the sensor. Similarly, a single measurement from a sensor may produce noise - some of which depends on the signal you receive and some of which depends on the detector. Noise can seriously affect the quality of the picture, so it is essential to know the source of the noise and how much noise to try to account for when imaging.

Dark No-Signal Images

For the dark images, mean offset and noise images can be seen in Figure 1A. The offset values are relatively uniform and dark, with a collection of lighter pixels towards the bottom of the image. Additionally, there are no visible “bad” pixels or very light pixels in regions that should be dark in the mean offset image, but there are some in the noise image, as shown by the light pixels on the color bar. These are both crucial factors to consider while imaging. The lighter region of the mean offset value image implies that the dark current in the sensor leads to false signals and that they are more concentrated along the bottom of the image. This is important to consider if the analysis is done in an AOI on the bottom of a captured image. Additionally, the bad pixels in the noise imply that without any light signal, there is noise from the dark current inherent in any image taken.

Additionally, Figure 2B shows the relationship between mean offset value in an AOI versus exposure time to be initially linear and somewhat positively correlated, with a significant jump in mean offset values at an exposure time around 50 ms. Although we took 50 frames for averaging to eliminate noise from single measurements, this big jump also demonstrates the crucial effects of dark current and how increasing exposure time generally increases the mean offset value. This aligns with the expected results that when taking an image for a longer time, the amount of dark current that’s imaged will be larger, with more time for more electrons to move across the diode. These CMOS cameras and sensors have shown an unexpected jump at the 50 ms exposure time. This could be due to hardware issues or that images taken at low exposure times below 60 ms can be unreliable since there is not enough time to take a proper image. All in all, this implies images taken at around 50 ms should be carefully analyzed in case the steep jump in mean offset values affects the image quality.

Similarly, Figure 2C shows the relationship between noise and exposure time also to be positively linear. This is logical, considering the dark current causing the signal with the cap on would have a larger effect on the image when collected over a longer time. This is an important characteristic to remember when imaging, since the tradeoff of a longer exposure time can take in more photons to produce larger noise from the dark current.

Light Signal Images

For the light images, the mean offset values and noise images are shown in Figure 3A. The mean offset image has high spatial resolution and accurately represents the paint swatch. The noise image has the highest values along the edges of the white stripes between the shades of gray. The stripes with the highest noise values, where the color changes from the gray shade to the white stripe also make sense. The imaging system will be noisier when there is a sharp change between nearby pixel values, since the sensor taking in the jump between colors is more difficult, varied, and prone to errors.

An important discovery surrounding the noise versus exposure time graph in Figure 3C was that the no-signal noise effects were non-significant compared to the signal noise, since the no-signal noise values ranged from 0.1 to 0.45, and the signal noise ranged from 0 to 10. This means the signal variance from experiment 2 can be discussed exclusively as signal noise, without having to remove the no-signal noise since it can be disregarded manually. The important trends of the noise versus exposure time graph are that exposure time is positively correlated with noise, as the highest noise values are at the highest exposure times, and these variables have a roughly linear relationship. It is logical that the more time the camera is imaging the signal, the more incident light will hit the sensor, and the noisier the signal will be. This is exacerbated by the poor

resolution of a CMOS sensor due to the pixel not being entirely photosensitive to circuitry on the surface. An unexpected trend was that the darkest gray shade had the highest noise, and the noise decreased with lighter shades. This is discussed in depth below, but is thought to be because of the physical variety on the darkest shade of the paint swatch. The most significant impacts on noise are the amount of light entering the camera, and the difference in the amount of light entering the camera across adjacent pixels. The biggest impacts on image quality are aperture, focus, exposure time, and minimizing the above factors that lead to more noise.

Figure 3D shows the SNR at the AOI as a function of exposure time. The SNR increases with exposure time, and plateaus at around 75 ms. The SNR increases with increasing exposure time because the signal captured is greater. It begins to plateau because the noise in the denominator of the ratio becomes larger with increasing exposure time, as seen in graph 3C. Therefore, an important conclusion is that increasing the exposure time initially allows for a higher quality image, but the tradeoff is increased noise, so an ideal exposure time for the highest SNR is likely in the middle of the achievable exposure range.

The optimal strategies for the overall imaging had the aperture set to 6, the focus=0.37m, and exposure = 120 ms. The optimal strategy for the darkest AOI only had the same aperture and focus, with a slightly higher exposure of 130 ms. We arrived at this strategy by first visually assessing the focus and lack of blur in the image when adjusting the manual focus and aperture settings. The other virtually set camera settings were determined to have a much smaller effect on the final image quality, so the values stated in the lab report and initialized from lab 0 were kept. The ideal exposure time was then selected based on the SNR calculated from the array of tested exposure times. The darkest AOI required a slightly higher exposure time since there were fewer

photons reflected off the dark AOI, so a high-quality image required collecting the small amount of signal for longer.

Auto-exposure Algorithm

As shown in the Result Section, we utilized techniques learned from Part 1 and Part 2 of the lab. Specifically, that the best SNR value would help us find the ideal exposure time. As Figure 5 showed, we used the settled scene to test our auto-exposure tool. We are choosing to take a derivative of an SNR ratio because we observed that in Figure 3D, the SNR improved at the first 50ms, then plateaued due to increased noise. Our first trial gave us an unusual convex curve and it shows that the longer exposure time should have a higher SNR ratio which conflicted with our conclusion in Part 2. Therefore, we retested by changing the frame rate = "5MHz", and the exposure time range is inversely increased. We then found a peak in the SNR line graph shown in Figure 6B. And for the derivative close to 0, we decided this was the best exposure level with the highest SNR.

During the experiment, another concern emerged: our tool took a very long time and large memory spaces for data collection. Since our three AOIs have the same intensity value when they test separately due to hardware issues, we took the whole 1024x1280 image to test the tool. Since averaging 20 frames took us about 10GB of memory, we cropped the full field image into a smaller one which loses some boundary information. This method could be utilized when the object is uniform or when boundary information is less critical.

Another potential solution is to test and record the best SNR for different grayscale intensities. Then, we build a 'dictionary' to pair intensity with exposure time. As we have

collected data in the histogram part, the majority pixel intensity refers to a stored exposure value such that we only use O(1) time to find the best exposure level.

This tool was also tested with the cap on to assess auto exposure on no-signal images. The results were generally a lower exposure time than the results from the tool with the cap off. Considering the dark field images have increased noise with increased exposure, the highest SNR would be at a lower exposure. However, our results were inconsistent, and determining a no-signal SNR proved to be difficult.

Histogram

During the experiment, we found that drawing the histograms to count the number of pixels with different intensities is an efficient and quicker way to obtain a better exposure time from where most pixels lie on the bins. From Figure 7, we analyzed that at very low exposures, the pixels are lying close to 0 and with a shaper distribution. When the exposure time becomes longer, the pixel values are shifted to a higher pixel intensity with more uniform distribution. And for the longer exposure time, there are extremely high intensity values on the histogram due to the light noise from photons. With longer exposure time, more light noise is detected as more pixels lie on the maximum value shown in the histogram.

Unusual findings

As from Figure 3C, we noticed that the noise at the darkest region has higher noise than the lightest gray. Theoretically speaking, the lightest gray should have higher noise due to the fact that there is more light noise in the lightest gray. We initially thought the problem came from our dataset. So we recollected data by reperforming the Part 2 experiment entirely. The same

calculation and different experiment data produced similar results. We also thought about the differences that came from the variance calculation, so we tried to code the variance method by calculating the square of the differences between the mean offset images and the mean from the ensemble data. This also did not affect the result. As shown in Figure 8, based on the specified frames drawn from the raw data, we could see that the lightest gray has a more uniform result compared to the dark gray. On the other hand, there are more white pixels in captured dark gray images which cause the noise in the darker gray is higher than the lightest one.

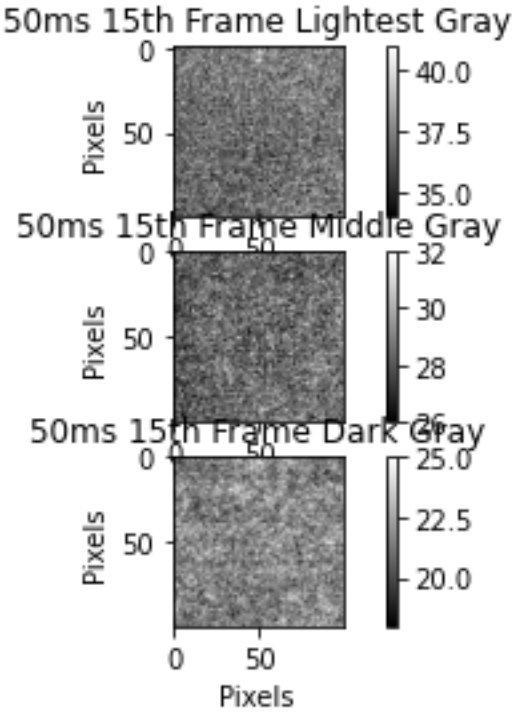


Figure 8. The 15th frame of 100x100 pixels AOIs for Lightest, Middle, Dark Gray at 50ms

Main Points Learned in this Lab

- We found that image averaging is critical to reduce the effect of noise.
- We found that the relationship between the mean offset value in an AOI vs exposure time in dark no-signal images is linearly proportional, and that this relationship is important

because it demonstrates the important effects of dark current. In short, it shows how increasing exposure time generally increases the mean offset value.

- We found the relationship between the SNR at the light image's AOIs vs the exposure time, and this relationship is important because it tells us how to find the best quality of an image. Specifically, increasing the exposure time initially allows for a higher quality image, but the tradeoff is increased noise. Therefore, an ideal exposure time for highest SNR is likely in the middle of the achievable exposure range.
- We found the relationship between noise and exposure time is important because the tradeoff of a larger exposure time that can take in more photons is larger noise from the light and dark current.
- We found that the clock time has never changed except when we set the pixel clock. This is because the pixel clock determines the speed of the sensor cells that can be read out, which means it does not change with other settings unless we set the pixel clock.
- We found that SNR is a crucial parameter for image quality evaluation. If either the image signal is larger or the noise signal is smaller, we would consider as good image quality and it is essential for achieving proper exposure time.

Appendix: Python Code(Next page)

Lab 1:

The main purpose of Lab 0 is to familiarize yourselves with the camera. You should be able to capture an image, establish a basic understanding of camera parameters and adjust them in live image display to obatin the best image possible

Import basic Python modules

```
# Required imports
import sys
import os
from PIL import Image
import numpy as np
import time
import matplotlib.pyplot as plt
import cv2
# from instrumental.drivers.cameras import uc480
import scipy.io as sio
```

```
# Use when camera is unmounted and dataset is stored in the drive.
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Camera object instantiation

```
# initialize camera
# lists the cameras that are detected.
instruments = uc480.list_instruments()
# initiliaze the camera using the detected cameras.
cam = uc480.UC480_Camera(instruments[0])
# Open Camera. DONT FORGET TO CLOSE!
cam.open()
```

Check camera properties, change Camera settings and grab an image.

```
# Connect to the camera and set up the image memory
cam.open()
# view current camera params.
cam.get_parameters()
```

```
Exposure Time: 66.62091228070176ms
Exposure Time Range: [0.008982456140350877 66.62091228070176]
Framerate: 14.997947649269046Hz
Pixel Clock: 24MHz
Gain: 1.0
```

Set Framerate

```
# set frame rate
cam.set_framerate(framerate = "7Hz")
# get exposure range, you can see that the frame rate will change the maximum exposure time.
print(cam._get_exposure_range())
# print current frame rate
print(cam.framerate)
```

```
(<Quantity(0.00898245614, 'millisecond')>, <Quantity(142.790456, 'millisecond')>)
7.000578714507065 hertz
```

Set Exposure Time

```
# get exposure range
cam._get_exposure_range()
"""
(0.008982456140350877 <Unit('millisecond')>,
 142.79045614035087 <Unit('millisecond')>)
"""
```

```
# set cam exposure (ms)
cam._set_exposure('120 ms')
cam._get_exposure()
```

```
119.99981718398561 millisecond
```

Setting Pixel Clock

```
# Set pixelclock (Use the default)
cam.set_pixelclock(pixel_clock='7MHz')

# show the current pixel clock.
cam.pixelclock
```

```
7 megahertz
```

Setting Gain

```
# The master gain factor; 1.0 is the lowest gain, max gain is 4.0
# Use the default gain.
cam._set_gain(30)

# get max gain
cam.max_master_gain

# get current gain.
cam.master_gain
```

```
1.92
```

```
cam.get_parameters()
```

```
Exposure Time: 410.35267486080465ms
Exposure Time Range: [0.008982456140350877 488.26301754385963]
Framerate: 2.0472896614601814Hz
Pixel Clock: 7MHz
Gain: 1.92
```

```
print(cam._width)
```

```
1280
```

```
print(cam._height)
```

```
1024
```

```
x0, y0 = 0, 0
# this sets the width and height of the rectangle.
width, height = 1280, 1024
cam._set_AOI(x0, y0, width, height)
cam._get_AOI()
```

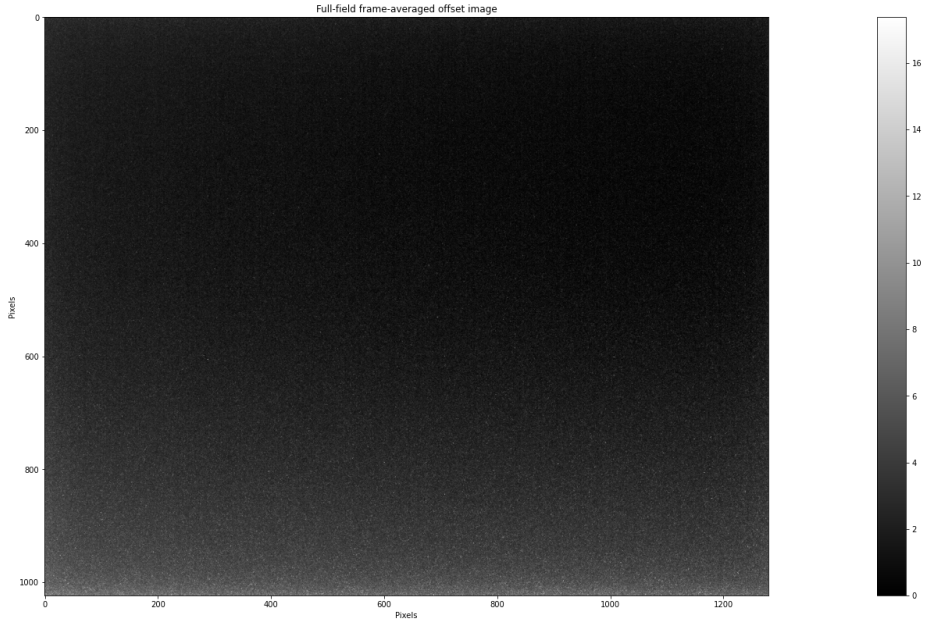
```
(0, 0, 1280, 1024)
```

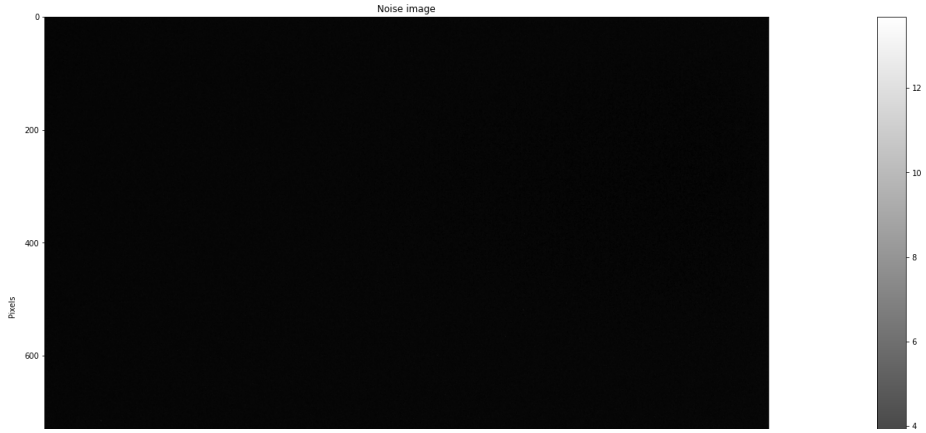
Take an image

```
## Part 1
# Taking the average of 50 frames image and the variance image
img_total = 0
avg = np.zeros((50,1024,1280))
for i in range(50):
    # set cam exposure (ms)
    cam._set_exposure('120 ms')
    cam._get_exposure()
    #take an image
    img = cam.grab_image().copy()
    avg[i,:,:] = img

# average = np.mean(avg, axis = 0)
# variance = np.var(avg, axis = 0)
```

```
float_img = avg.astype('float')
average = np.mean(float_img, axis = 0)
```

```
variance = np.var(float_img, axis = 0)

# Save Full-field frame-averaged offset image and noise image as .mat file
sio.savemat("50framesMean_part1.mat", {"average":average})
sio.savemat("50framesVariance_part1.mat", {"variance":variance})


# Check images are saved by loadmat()
average = sio.loadmat("50framesMean_part1.mat")
variance =  sio.loadmat("50framesVariance_part1.mat")


## Part 1
# Plot full-field frame-averaged offset image and noise image with colorbar
f = plt.figure()
f.set_figwidth(50)
f.set_figheight(30)

f.add_subplot(2,1,1)
ax1 = plt.subplot(2,1,1)
ax1.set_xlabel("Pixels")
ax1.set_ylabel("Pixels")
plt.imshow(average, cmap = 'gray')
plt.title('Full-field frame-averaged offset image')
plt.colorbar()

f.add_subplot(2,1,2)
ax2 = plt.subplot(2,1,2)
ax2.set_xlabel("Pixels")
ax2.set_ylabel("Pixels")
plt.imshow(variance, cmap = 'gray')
plt.title('Noise image')
plt.colorbar()

# Save two images as .png
f.savefig("50FramesAvgVar.png")
```





Set Image ROI

You don't always have to capture the full-field image, feel free to narow down to a specific ROI and focus on your target of interest

Attention: Set the image ROI based on coordinates. This sets the starting point (x_0, y_0) of the rectangular ROI. Then you set the width (X-direction Left to Right) and height of the rectangle (Y-direction top to bottom).

```
# get size of max image.
cam.open()
cam._get_max_img_size() # Width, Height
```

```
(1280, 1024)
```

```
# set the image ROI based on coordinates.
# This sets the starting point of the rectangle in the X-direction west to east. and Y-direction north to south
x0, y0 = 600, 0
# this sets the width and height of the rectangle.
width, height = 100, 100
cam._set_AOI(x0, y0, width, height)
cam._get_AOI()
```

```
(600, 0, 100, 100)
```

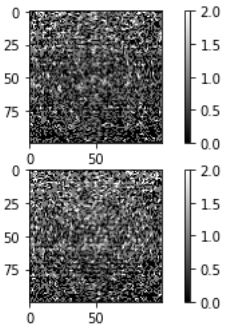
```
## Part 1
# 100 frames at 15 exposures
x = np.zeros((17, 100,100,100))

exp = [10, 15, 20, 35, 40, 50, 60, 75, 80, 90, 100, 110, 115, 120, 125, 130, 140]
j = 0
for val in exp:
    print("Current exp:", val)
    for i in range(100):
        # set cam exposure (ms)
        cam._set_exposure(str(val)+ ' ms')
        #take an image
```

```
            img = cam.grab_image().copy()
            x[j,i,:,:] = img
        j += 1
```

```
Current exp: 10
Current exp: 15
Current exp: 20
Current exp: 35
Current exp: 40
Current exp: 50
Current exp: 60
Current exp: 75
Current exp: 80
Current exp: 90
Current exp: 100
Current exp: 110
Current exp: 115
Current exp: 120
Current exp: 125
Current exp: 130
Current exp: 140
```

```
## Part 1
# For randomly picking two frames with varied exposure from exposure list
x = x.astype('float')
f = plt.figure()
f.add_subplot(2,1,1)
ax1 = plt.subplot(2,1,1)
plt.imshow(x[0,0,:,:], cmap = 'gray')
plt.colorbar()
f.add_subplot(2,1,2)
plt.imshow(x[0,49,:,:], cmap = 'gray')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x21e2473afd0>
```

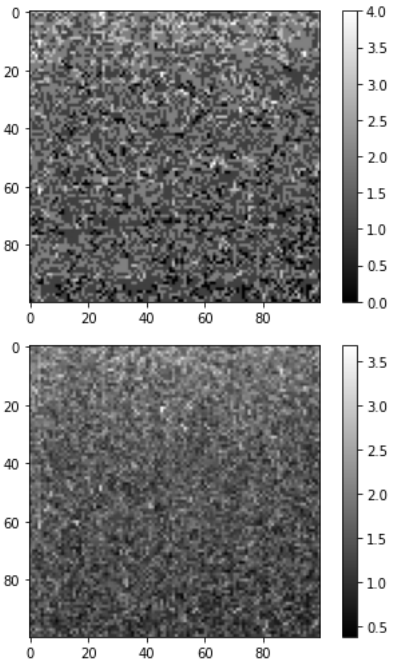



```
sio.savemat("100frames15exp_part1.mat", {"images":x})
```

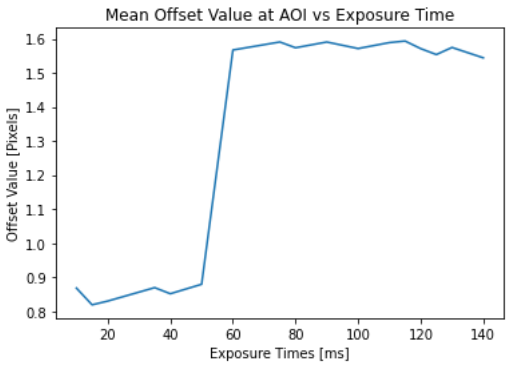
```
x = sio.loadmat("/content/drive/MyDrive/Imaging Instrumentation Dataset/100frames15exp_part1.mat")["images"]
x = x.astype("float")
```

```
## Part 1
# Graph of exposure time vs mean offest value in AOIs
offset = np.zeros((17,100,100))
for i in range(17):
    offset[i,:,:] = np.mean(x[i,:,:,:],axis = 0)

offset_vals = np.zeros((17,1))
for j in range(17):
    offset_vals[j] = np.mean(offset[j,:,:])

#Either just do it about axis 2 since it's a square, or do np.mean twice over axis 1 and 2,
# or fo np.mean over axis 1 (--> (17,100) matrix of avg column values) then divide by 100
# I don't think averaging over columns will work for variance and stdev

# offset_vals = np.zeros((17,1))
# for i in range(17):
#   xx = x[i,:,:,:]
#   xx_offset_by_frames = np.mean(xx,axis = 0)
#   offset_vals[i] = np.mean(xx_offset_by_frames,axis=(0,1))


plt.imshow(x[6,49,:,:], cmap = 'gray')
plt.colorbar()
plt.figure()
plt.imshow(offset[6,:,:], cmap = 'gray')
plt.colorbar()
```

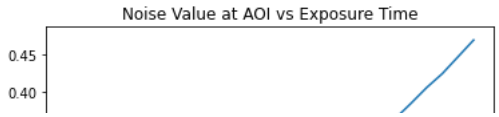
```
<matplotlib.colorbar.Colorbar at 0x7f67e0a685e0>
```



```
## Part 1
exp = [10, 15, 20, 35, 40, 50, 60, 75, 80, 90, 100, 110, 115, 120, 125, 130, 140]
plt.plot(exp, offset_vals)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Offset Value [Pixels]")
plt.title("Mean Offset Value at AOI vs Exposure Time")
plt.savefig("MeanOffsetVsExposure_part1.png")
```




```
## Part 1
# Graph of exposure time vs variance value in AOIs
variance = np.zeros((17,100,100))
for i in range(17):
    variance[i,:,:] = np.mean(x[i,:,:,:],axis = 0)

variance_vals = np.zeros((17,1))
for j in range(17):
    variance_vals[j] = np.var(variance[j,:,:])



plt.plot(exp, variance_vals)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Noise [Pixels]")
plt.title("Noise Value at AOI vs Exposure Time")
plt.savefig("NoiseVarianceVsExposure_part1.png")
```

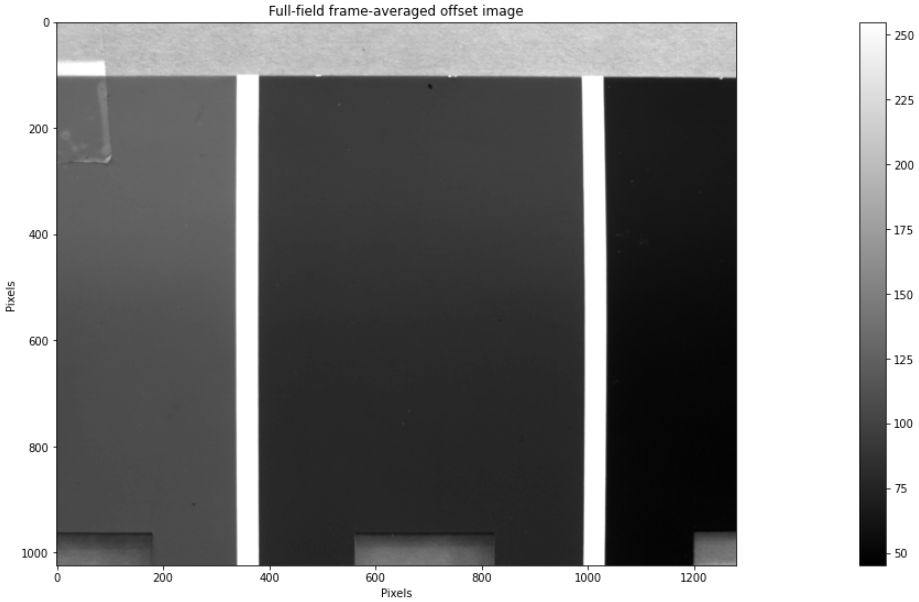


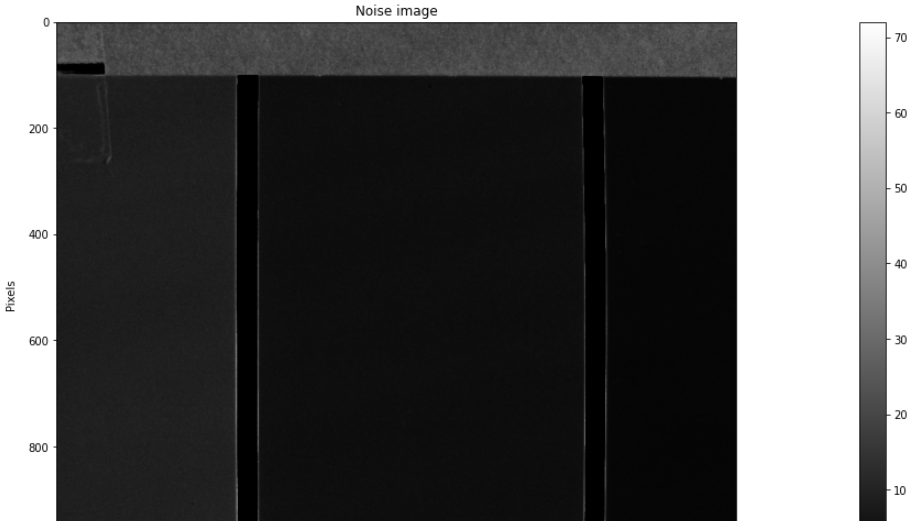
Take some images and vary exposure time and ROI to get a sense of the camera.

```
## Part 2
# Full-field mean and variance images with good exposure
# This sets the starting point of the rectangle in the X-direction west to east. and Y-direction north to south
x0, y0 = 0, 0
# this sets the width and height of the rectangle.
width, height = 1280, 1024
cam._set_AOI(x0, y0, width, height)
cam._get_AOI()

# Aperature: 6
# Focus: 0.37 m
# Exposure: 120 ms
```

```
(0, 0, 1280, 1024)
```

```
## Part 2
img_total = 0
avg2 = np.zeros((50,1024,1280))
for i in range(50):
    # set cam exposure (ms)
    cam._set_exposure('120 ms')
    cam._get_exposure()
    #take an image
    img = cam.grab_image().copy()
    avg2[i,:,:] = img

avg2 = avg2.astype('float')
average2 = np.mean(avg2, axis = 0)
variance2 = np.var(avg2, axis = 0)


sio.savemat("50framesMean_part2.mat", {"average2":average2})
sio.savemat("50framesVariance_part2.mat", {"variance2":variance2})


## Part 2
f = plt.figure()
f.set_figwidth(40)
f.set_figheight(20)

f.add_subplot(2,1,1)
ax1 = plt.subplot(2,1,1)
ax1.set_xlabel("Pixels")
ax1.set_ylabel("Pixels")
plt.imshow(average2, cmap = 'gray')
plt.title('Full-field frame-averaged offset image')
plt.colorbar()


f.add_subplot(2,1,2)
ax2 = plt.subplot(2,1,2)
ax2.set_xlabel("Pixels")
ax2.set_ylabel("Pixels")
plt.imshow(variance2, cmap = 'gray')
plt.title('Noise image')
plt.colorbar()

f.savefig("50FramesAvgVar_part2.png")
```



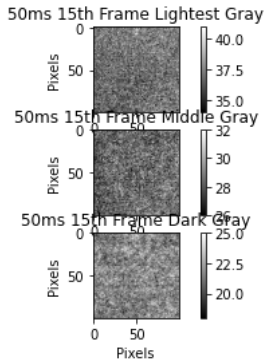



```
## Full-field image and cropped to 3 AOIs
x0, y0 = 0,0
width, height = 1280, 1024
cam._set_AOI(x0, y0, width, height)
cam._get_AOI()

full_images_x = np.zeros((15,100,100,100))
full_images_x2 = np.zeros((15,100,100,100))
full_images_x3 = np.zeros((15,100,100,100))
exp = [20, 35, 40, 50, 60, 75, 80, 90, 100, 110, 115, 120, 125, 130, 140]
j = 0
for val in exp:

    for i in range(100):
        print(val,i)
        # set cam exposure (ms)
        cam._set_exposure(str(val)+ ' ms')
        #take an image
        img = cam.grab_image().copy()

        full_images_x[j,i,:,:] = img[200:300, 200:300]
        full_images_x2[j,i,:,:] = img[200:300, 450:550]
        full_images_x3[j,i,:,:] = img[200:300, 1100:1200]
    j += 1

sio.savemat("ThreeAOIs100Frames_part2_second_time_try.mat", {"lightest_gray":full_images_x, "middle_gray":full_images_x2,"dark

part2_images = sio.loadmat("/content/drive/MyDrive/Imaging Instrumentation Dataset/ThreeAOIs100Frames_part2_second_time_try.ma


full_images_x = part2_images["lightest_gray"]
full_images_x2 = part2_images["middle_gray"]
full_images_x3 = part2_images["darkest_gray"]
x = full_images_x.astype('float')
```

```
x2 = full_images_x2.astype('float')
x3 = full_images_x3.astype('float')
```

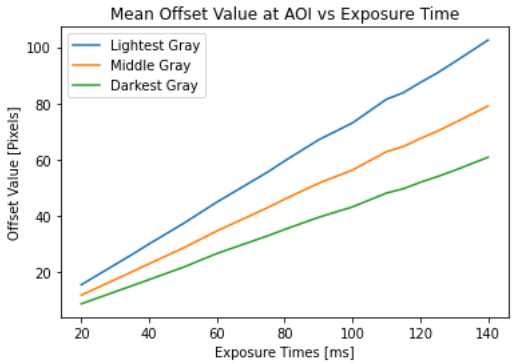
```
f = plt.figure()

f.add_subplot(3,1,1)
ax1 = plt.subplot(3,1,1)
ax1.set_xlabel("Pixels")
ax1.set_ylabel("Pixels")
plt.imshow(x[3,50,:,:], cmap = 'gray')
plt.title('50ms 15th Frame Lightest Gray')
plt.colorbar()

f.add_subplot(3,1,2)
ax1 = plt.subplot(3,1,2)
ax1.set_xlabel("Pixels")
ax1.set_ylabel("Pixels")
plt.imshow(x2[3,50,:,:], cmap = 'gray')
plt.title('50ms 15th Frame Middle Gray')
plt.colorbar()

f.add_subplot(3,1,3)
ax1 = plt.subplot(3,1,3)
ax1.set_xlabel("Pixels")
ax1.set_ylabel("Pixels")
plt.imshow(x3[3,50,:,:], cmap = 'gray')
plt.title('50ms 15th Frame Dark Gray')
plt.colorbar()
```

```
<ipython-input-5-701dbe99b23b>:4: MatplotlibDeprecationWarning: Adding an axes usi
  ax1 = plt.subplot(3,1,1)
<ipython-input-5-701dbe99b23b>:12: MatplotlibDeprecationWarning: Adding an axes us
  ax1 = plt.subplot(3,1,2)
<ipython-input-5-701dbe99b23b>:20: MatplotlibDeprecationWarning: Adding an axes us
  ax1 = plt.subplot(3,1,3)
<matplotlib.colorbar.Colorbar at 0x7fdeda695220>
```

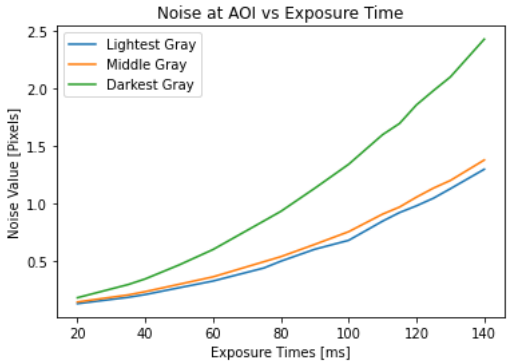



```
## Part 2
# Graph of exposure time vs mean offest value in AOI: Three grays
exp = [20, 35, 40, 50, 60, 75, 80, 90, 100, 110, 115, 120, 125, 130, 140]
# Lightest gray
offset1 = np.zeros((15,100,100))
for i in range(15):
    offset1[i,:,:] = np.mean(x[i,:,:,:],axis = 0)
offset1_vals = np.zeros((15,1))
for j in range(15):
    offset1_vals[j] = np.mean(offset1[j,:,:])


# Middle gray
offset2 = np.zeros((15,100,100))
for i in range(15):
    offset2[i,:,:] = np.mean(x2[i,:,:,:],axis = 0)
offset2_vals = np.zeros((15,1))
for j in range(15):
    offset2_vals[j] = np.mean(offset2[j,:,:])


# Darkest gray
offset3 = np.zeros((15,100,100))
for i in range(15):
    offset3[i,:,:] = np.mean(x3[i,:,:,:],axis = 0)
offset3_vals = np.zeros((15,1))
for j in range(15):
    offset3_vals[j] = np.mean(offset3[j,:,:])
```

```
plt.plot(exp, offset1_vals, label = "Lightest Gray")
plt.plot(exp, offset2_vals, label = "Middle Gray")
plt.plot(exp, offset3_vals, label = "Darkest Gray")
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Offset Value [Pixels]")
plt.title("Mean Offset Value at AOI vs Exposure Time")
plt.legend()
plt.savefig("MeanOffsetVsExposure_part2.png")
```

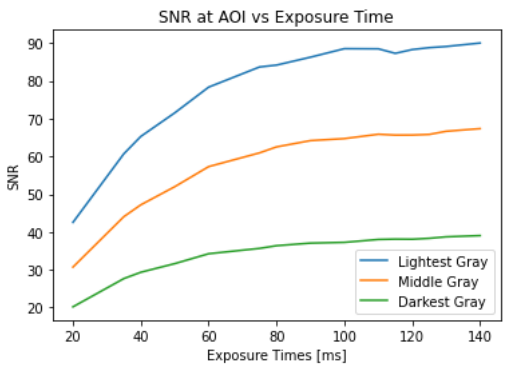



```
## Part 2
# Graph of exposure time vs. noise/ variance value in AOI: three grays
# More photons, more light noise.

# Lightest gray
variance = np.zeros((15,100,100))
variance_vals = np.zeros((15,1))
for i in range(15):
    variance_vals[i] = np.var(offset1[i,:,:])


# Middle Gray
variance2 = np.zeros((15,100,100))
variance2_vals = np.zeros((15,1))
for i in range(15):
    variance2_vals[i] = np.var(offset2[i,:,:])


# Darkest Gray
variance3 = np.zeros((15,100,100))
variance3_vals = np.zeros((15,1))
for i in range(15):
    variance3_vals[i] = np.var(offset3[i,:,:])


plt.plot(exp, variance_vals, label = "Lightest Gray")
plt.plot(exp, variance2_vals, label = "Middle Gray")
plt.plot(exp, variance3_vals, label = "Darkest Gray")
plt.legend()
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Noise Value [Pixels]")
plt.title("Noise at AOI vs Exposure Time")
plt.savefig("NoiseVarianceVsExposure_part2.png")

## We initially thought the problem came from our dataset. So we recollect data by reperforming the part 2 experiement.
## with the same calculation method. But different experiemnt produces similar result. We also thought about the differences
## came from the variance calculation, so we tried hard code the variance method. It did not affect the result.
## Based on the specified several frames drawn from the colloected data, we could see that the darker gray has more uniform re
```




```
## Part 2
#SNR graph
```

```
# Lightest Gray
stdev = np.zeros((15,100,100))
for i in range(15):
    stdev[i,:,:] = np.mean(x[i,:,:,:],axis = 0)
stdev_vals = np.zeros((15,1))
for j in range(15):
    stdev_vals[j] = np.std(stdev[j,:,:])

snr = np.zeros((15,1))
for k in range(15):
    val = offset1_vals[k] / stdev_vals[k]
    snr[k] = val

# Middle Gray
stdev2 = np.zeros((15,100,100))
for i in range(15):
    stdev2[i,:,:] = np.mean(x2[i,:,:,:],axis = 0)
stdev2_vals = np.zeros((15,1))
for j in range(15):
    stdev2_vals[j] = np.std(stdev2[j,:,:])

snr2 = np.zeros((15,1))
for k in range(15):
    val = offset2_vals[k] / stdev2_vals[k]
    snr2[k] = val

# Darkest Gray
stdev3 = np.zeros((15,100,100))
for i in range(15):
    stdev3[i,:,:] = np.mean(x3[i,:,:,:],axis = 0)
stdev3_vals = np.zeros((15,1))
for j in range(15):
    stdev3_vals[j] = np.std(stdev3[j,:,:])

snr3 = np.zeros((15,1))
for k in range(15):
    val = offset3_vals[k] / stdev3_vals[k]
    snr3[k] = val


plt.plot(exp, snr, label = "Lightest Gray")
plt.plot(exp, snr2, label = "Middle Gray")
plt.plot(exp, snr3, label = "Darkest Gray")
plt.xlabel("Exposure Times [ms]")
plt.ylabel("SNR")
plt.title("SNR at AOI vs Exposure Time")
plt.legend()
plt.savefig("SNRVsExposure_part2.png")
```



```
# Part 3: auto-exposure
# Take 20 frames at various exposures over exposure range
# print(cam._get_exposure_range())
# Gives min and max, decide on step based on size of range/15
exposure = list(range(10, 140, 10))
x4 = np.zeros((len(exposure),20,1024,1280))
j = 0
for val in exposure:
    for i in range(20):
        print(val, i)
        # set cam exposure (ms)
        cam._set_exposure(str(val)+ ' ms')
        #take an image
        img = cam.grab_image().copy()
        x4[j,i,:,:] = img
    j += 1
sio.savemat("AutoExposureImageData_Part3", {"Images": x4})
```

```
x4 = sio.loadmat("/content/drive/MyDrive/Imaging Instrumentation Dataset/AutoExposureImageData_Part3")["Images"]
```

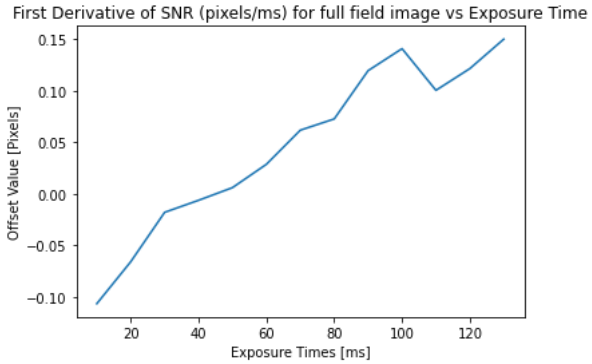
```
# Calculate SNR for each exposure (mean and standard deviation), put in a list (same method as above, but fixed)
x4 = x4.astype("float")
exposure = list(range(10, 140, 10))
offset4 = np.zeros((len(exposure),1024,1280))
offset4_vals = np.zeros((len(exposure),1))
stdev4_vals = np.zeros((len(exposure),1))
snr4 = np.zeros((len(exposure)))

for i in range(len(exposure)):
    offset4[i,:,:] = np.mean(x4[i,:,:,:],axis = 0)
    offset4_vals[i] = np.mean(offset4[i,:,:])


for j in range(len(exposure)):
    stdev4_vals[j] = np.std(offset4[j,:,:])


for k in range(len(exposure)):
    val = offset4_vals[k] / stdev4_vals[k]
    snr4[k] = val

print("SNR Lists:", snr4)
# If new SNR < old SNR, set exposure to that assoicated with old SNR;
# Also pick set of middle exposure values (cut out extremes we know won't work)

# Find max value in that list, then its location, then set exposure to value in list at same location
best = max(snr4)
print(best)
# loc_exp = snr.index(max)
# auto_exp = exposure[loc_exp]

# cam._set_exposure(auto_exp)
# cam._get_exposure()
```

```
SNR Lists: [1.73741299 1.63088075 1.60564952 1.59488963 1.59297964 1.60679859
 1.65010347 1.7300621  1.79520242 1.96879868 2.07656575 2.16939508
 2.31929688]
2.3192968757110974
```

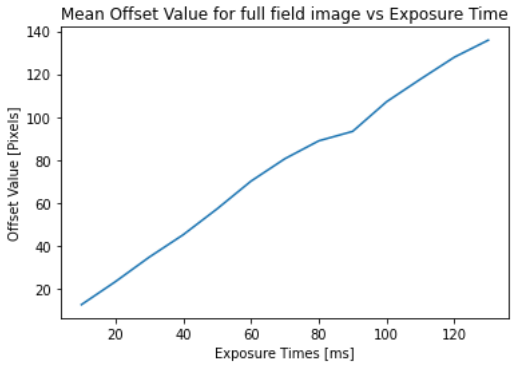
```
plt.imshow(x4[3,15,:,:], cmap = 'gray')
plt.colorbar()
plt.title("The 15th frame of full field image at 40ms")
plt.xlabel("Pixels")
plt.ylabel("Pixels")
```

```
Text(0, 0.5, 'Pixels')
```

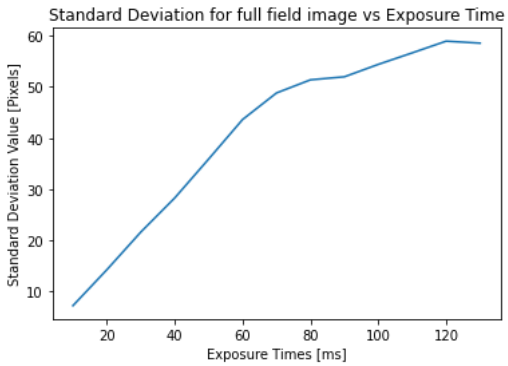
```
plt.plot(exposure, np.gradient(snr4))
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Offset Value [Pixels]")
plt.title("First Derivative of SNR (pixels/ms) for full field image vs Exposure Time")
plt.savefig("Derivative_MeanOffsetVsExposure_part1.png")
```




```
plt.plot(exposure, offset4_vals)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Offset Value [Pixels]")
plt.title("Mean Offset Value for full field image vs Exposure Time")
plt.savefig("MeanOffsetVsExposure_part1.png")
```

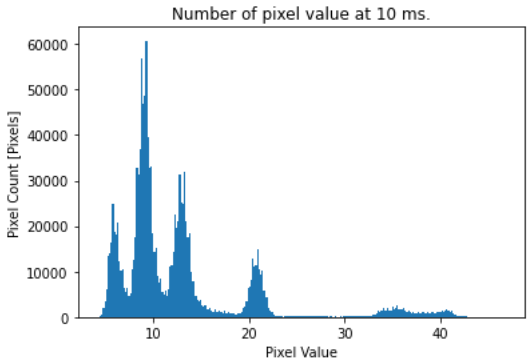


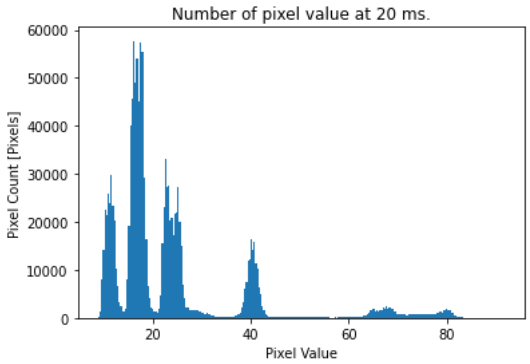

```
plt.plot(exposure, stdev4_vals)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Standard Deviation Value [Pixels]")
plt.title("Standard Deviation for full field image vs Exposure Time")
plt.savefig("Standard DeviationVsExposure_part1.png")
```

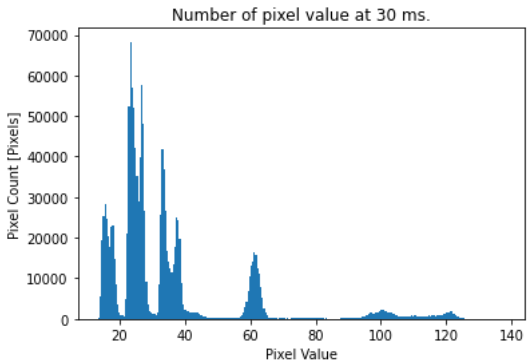


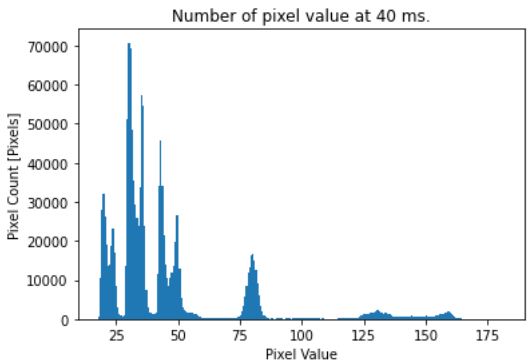

```
plt.plot(exposure, snr4)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("SNR Value [Pixels]")
plt.title("SNR for full field image vs Exposure Time")
plt.savefig("SNRVsExposure_part1.png")
```

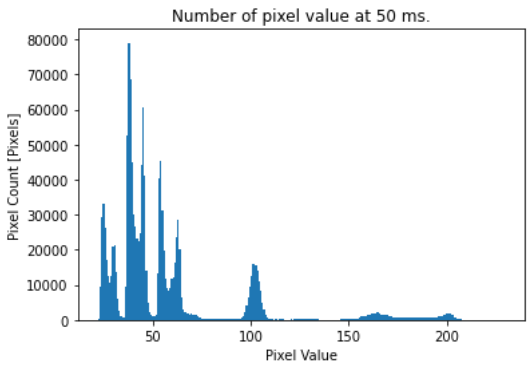

```
for i in range(len(exposure)):
  plt.figure()
  image1 = offset4[i,:,:]
  plt.hist(image1.flatten(),bins=256)
  plt.xlabel("Pixel Value")
  plt.ylabel("Pixel Count [Pixels]")
  plt.title("Number of pixel value at "+ str(exposure[i]) + " ms.")
```

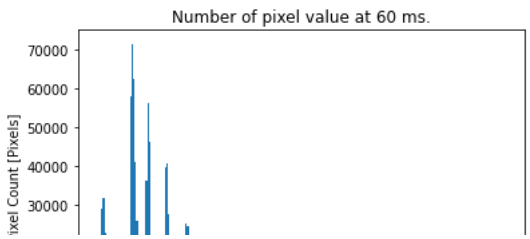

Number of pixel value at 10 ms.
Pixel Count [Pixels]
Pixel Value

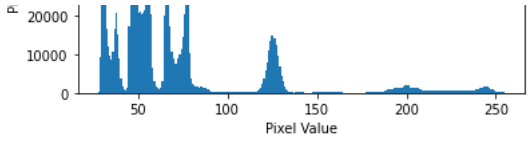

Number of pixel value at 20 ms.
Pixel Count [Pixels]
Pixel Value

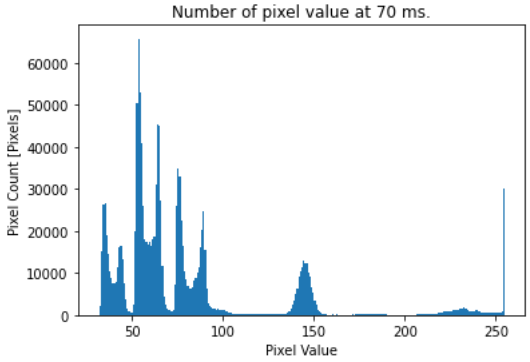

Number of pixel value at 30 ms.
Pixel Count [Pixels]
Pixel Value

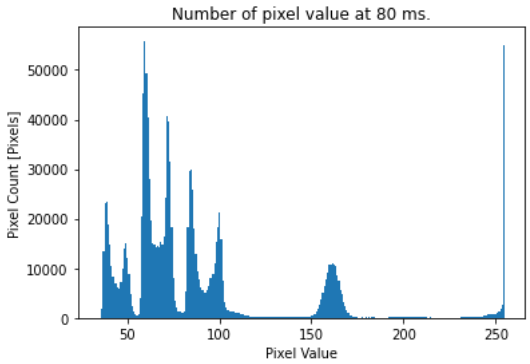

Number of pixel value at 40 ms.
Pixel Count [Pixels]
Pixel Value

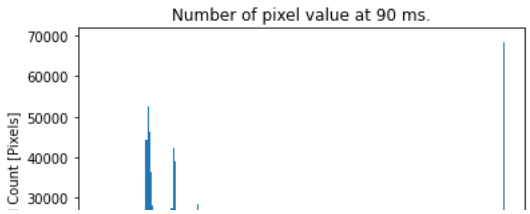

Number of pixel value at 50 ms.
Pixel Count [Pixels]
Pixel Value


Number of pixel value at 60 ms.
Pixel Count [Pixels]










```
## Part 3 Trial 2
```

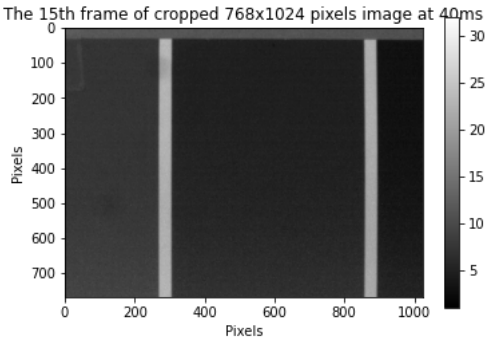

```
x4 = sio.loadmat("/content/drive/MyDrive/Imaging Instrumentation Dataset/AutoExposureImageData_Part3_Third")["Images"]

# Calculate SNR for each exposure (mean and standard deviation), put in a list (same method as above, but fixed)
x4 = x4.astype("float")
frames = 20
max_expo = 195
exposure = list(range(10, max_expo, 10))
print(list(exposure))
offset4 = np.zeros((len(exposure),128*6,128*8))
offset4_vals = np.zeros((len(exposure),1))
stdev4_vals = np.zeros((len(exposure),1))
snr4 = np.zeros((len(exposure)))

for i in range(len(exposure)):
    offset4[i,:,:] = np.mean(x4[i,:,:,:],axis = 0)
    offset4_vals[i] = np.mean(offset4[i,:,:])
#     stdev4_vals[i] = np.std(offset4[i,:,:])
#     val = offset4_vals[i] / stdev4_vals[i]
#     snr4[i] = val
#     print()

for j in range(len(exposure)):
    stdev4_vals[j] = np.std(offset4[j,:,:])


for k in range(len(exposure)):
    val = offset4_vals[k] / stdev4_vals[k]
    snr4[k] = val

print("SNR Lists:", snr4)
# If new SNR < old SNR, set exposure to that assoicated with old SNR;
# Also pick set of middle exposure values (cut out extremes we know won't work)

# Find max value in that list, then its location, then set exposure to value in list at same location
best = max(snr4)
```

```
print(best)
# loc_exp = snr.index(max)
# auto_exp = exposure[loc_exp]

# cam._set_exposure(auto_exp)
# cam._get_exposure()
```

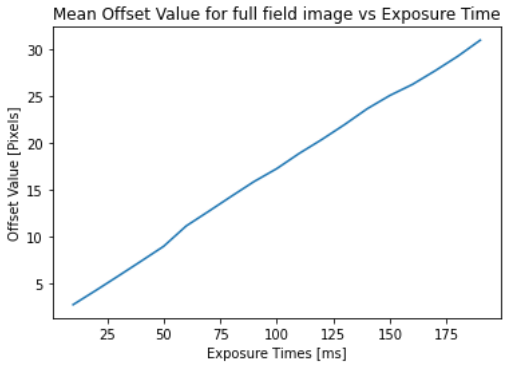
```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190]
SNR Lists: [1.87279709 1.85707604 1.79324743 1.73859333 1.70340803 1.78072608
 1.74575661 1.71256613 1.69391389 1.66459413 1.65463173 1.63588948
 1.62980337 1.62383981 1.61010198 1.60733079 1.60534511 1.60090666
 1.58923593]
1.8727970897217439
```

```
plt.imshow(x4[3,15,:,:], cmap = 'gray')
plt.colorbar()
plt.title("The 15th frame of cropped 768x1024 pixels image at 40ms")
plt.xlabel("Pixels")
plt.ylabel("Pixels")
```

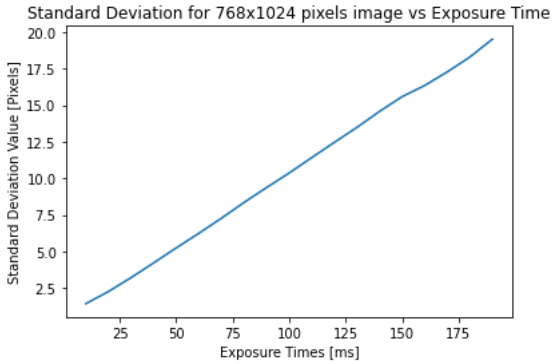
```
Text(0, 0.5, 'Pixels')
```




```
plt.plot(exposure, offset4_vals)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Offset Value [Pixels]")
plt.title("Mean Offset Value for 768x1024 pixels image vs Exposure Time")
plt.savefig("MeanOffsetVsExposure_part1.png")
```

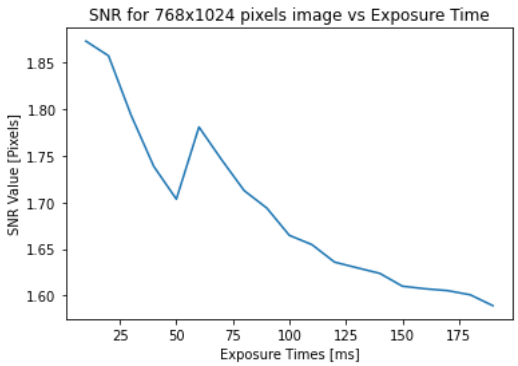



```
plt.plot(exposure, stdev4_vals)
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Standard Deviation Value [Pixels]")
plt.title("Standard Deviation for 768x1024 pixels image vs Exposure Time")
plt.savefig("Standard DeviationVsExposure_part1.png")
```

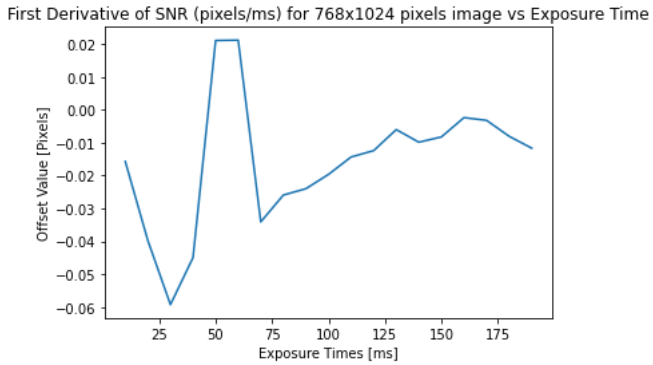



```
plt.plot(exposure, snr4)
plt.xlabel("Exposure Times [ms]")
```

```
plt.ylabel("SNR Value [Pixels]")
plt.title("SNR for 768x1024 pixels image vs Exposure Time")
plt.savefig("SNRVsExposure_part1.png")
```




```
plt.plot(exposure, np.gradient(snr4))
plt.xlabel("Exposure Times [ms]")
plt.ylabel("Offset Value [Pixels]")
plt.title("First Derivative of SNR (pixels/ms) for 768x1024 pixels image vs Exposure Time")
plt.savefig("Derivative_MeanOffsetVsExposure_part1.png")
```



Live Video Feed for tuning focal length and aperture.

```
cam._get_AOI() # look at the Area of Interest.
```

```
(0, 0, 1280, 1024)
```

```
# set frame rate
cam.set_framerate(framerate = "10Hz")
# get exposure range, you can see that the frame rate will change the maximum exposure time.
print(cam._get_exposure_range())
# print current frame rate
print(cam.framerate)
```

```
(<Quantity(0.00898245614, 'millisecond')>, <Quantity(197.594526, 'millisecond')>)
5.05606735178449 hertz
```

```
# to quit press live feed press "q"
cam.start_live_video()
while cam.is_open:

    frame = cam.grab_image(timeout='100s', copy=True)
    frame1 = np.stack((frame,) * 3,-1) #make frame as 1 channel image
    frame1 = frame1.astype(np.uint8)
    gray = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    #now u can apply opencv features
    cv2.imshow('Camera', gray)
    if cv2.waitKey(30) & 0xFF == ord('q'):
       break
cam.stop_live_video()
cv2.destroyAllWindows()
```

```
5.05606735178449 hertz
```

```
cam.close()
```

```
del cam
```