EECS 395/495: Introduction to Computational Photography

Homework 2: Measuring the sensor noise of your Tegra tablet

Student Name: Haikun Liu

Student Number: 2903021

NetID: hlg483

1 Write android code to capture a sequence of images

Noticing that the capacity of buffer for storing raw pictures is preset to 51 2584 × 1936 pictures, we can configure the for-loop to run 49 times to capture 49 RAW pictures for each of different sensitivity settings.

Using following code, we can obtain the maximum and minimum sensitivity values of the tablet camera and store separately into higher_sstt and lower_sstt:

```
Range<Integer> sstt =
characteristics.get(CameraCharacteristics.SENSOR_INFO_SENSITIVITY_RANGE);
Integer lower_sstt = sstt.getLower();
Integer higher_sstt = sstt.getUpper();
```

To turn off the to auto-exposure of the tablet:

```
requester.set(CaptureRequest.CONTROL_AE_MODE, CaptureRequest.CONTROL_AE_MODE_OFF);
```

To configure the camera using the highest, lowest or middle sensitivity values, the following request set function can be deployed:

```
requester.set(CaptureRequest.SENSOR_SENSITIVITY, (higher_sstt + lower_sstt)/2);
```

By changing the last parameter in the requester.set() function, we can capture the picture in highest sensitivity (= 1600), lowest sensitivity (= 100) or middle sensitivity (= 850).

Figure 1. Captured High Sensitivity Picture

Figure 2. Captured Low Sensitivity Picture

Figure 3. Captured Middle Sensitivity Picture

2 Process the image sequence to gather noise statistics

2.1 Load the raw images into MATLAB

Using the code below, we can load 49 pictures of a single sensitivity setting into MATLAB. After reading each picture, it is cropped into a 950*700 image with the upper-left corner located at (1275,635) of the original image. The cropped image is just a gradient picture in the scene. By arranging the image matrix, we can store the image into a column vector. In that way, we can more easily manipulate the images for following steps. For the total 49 images, we can build a 665000*49 container matrix to store them.

```
% Middle sensitivity = (High sensitivity + Low sensiticity)/2
figureNo = 0;
NumOfIms = 49;%There are totally 49 RAW pictures captured for each sensitivity
value
```

```
str_Path = '/Users/HKLHK/Desktop/2015 Fall Quarter (NU)/EECS495 Introduction to
Computational Photography/HW/HW2/Middle Sensitivity 850 (2)/';
for i = 1: NumOfIms
str_Load = strcat(str_Path, num2str(i-1), '.dng');%set image path for high
sensiticity pictures
t = Tiff(str_Load, 'r');
I = read(t); %read RAW images
Im = imcrop(I,[1275,635,949 699]);%crop picture into smaller size image 950*700
centered at (1275,635)
ImageMS(:,i) = double(reshape(Im, [ ], 1));%for each image, rearrange its pixels
into a column vector. (make it easy for future manipulation)
end
```

2.2 Plot a histogram of these pixel values for a given pixel

As shown in Figure 4, the histograms of 4 different pixels through 49 pictures of the high sensitivity setting are plotted. The shapes of the histogram are more close to Poisson distributions, where the middle of the histogram biases to the lower tail. When the camera is configured with a high sensitivity, there are more photons captured by the sensor for one shot. As the result, photon noise (signal-dependent) dominates the total noise distribution over the read noise and ADC noise (signal-independent).

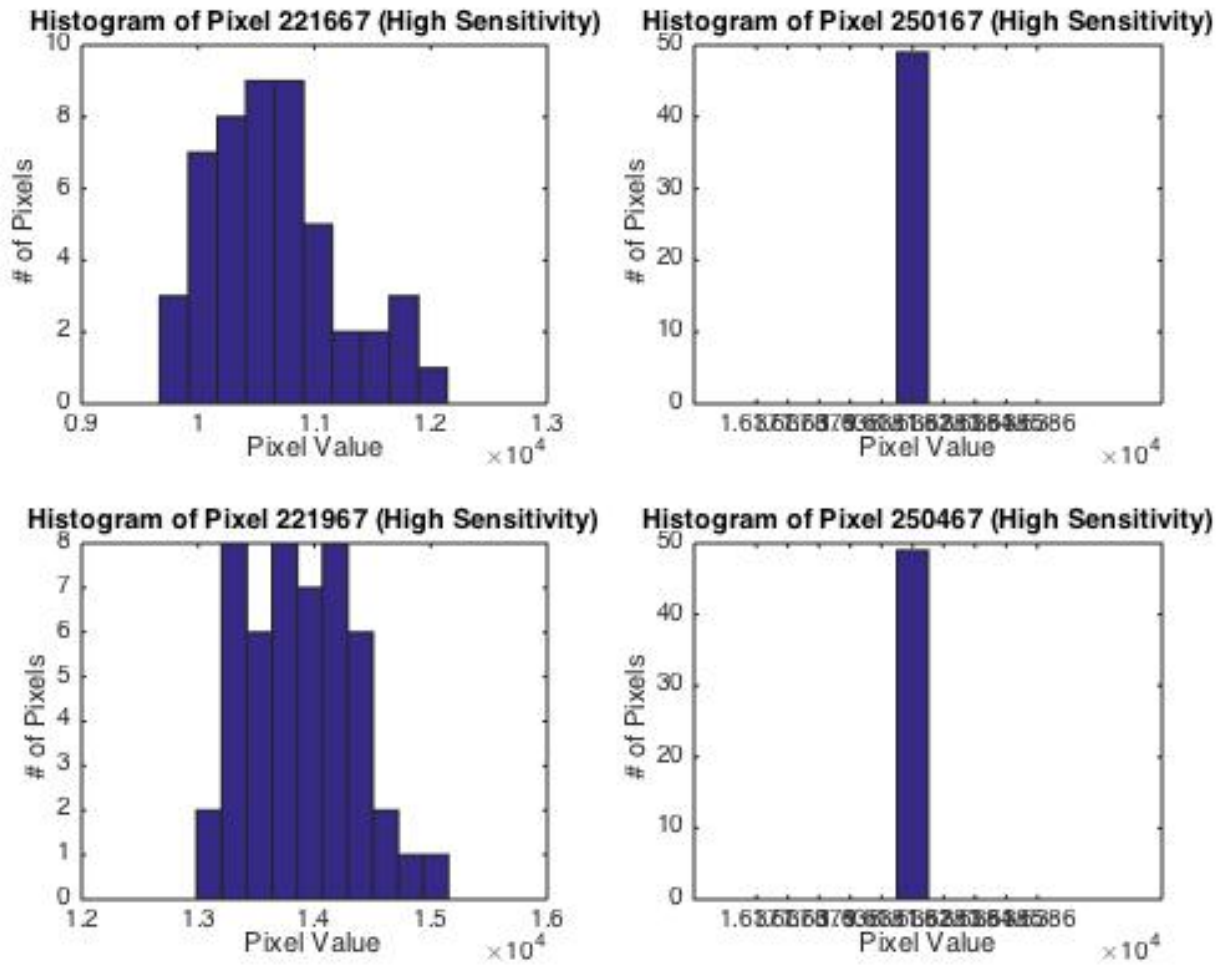


Figure 4. Histogram of pixel values: Pixel 221667, 250167, 221967, 250467 (High Sensitivity)

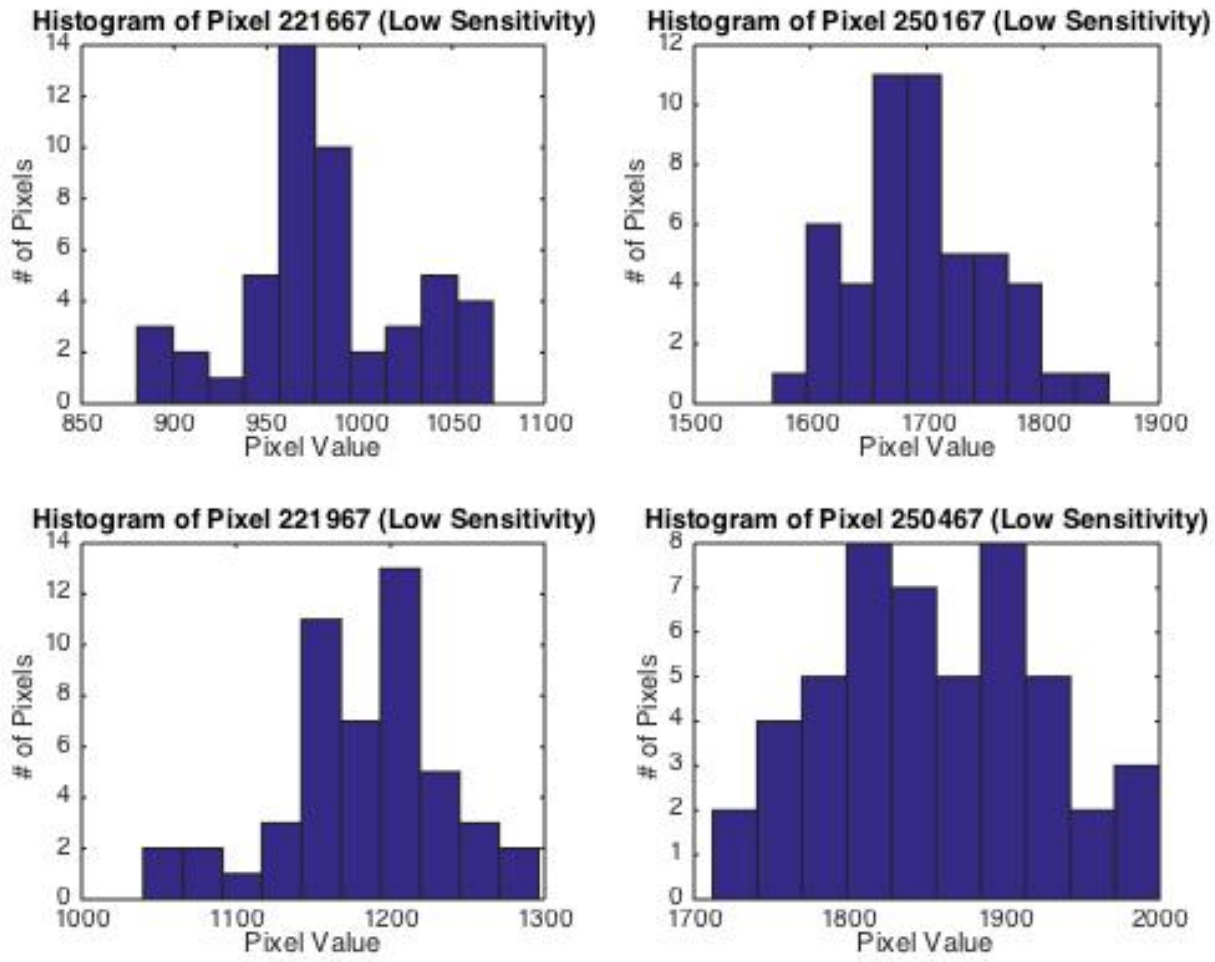


Figure 5. Histogram of pixel values: Pixel 221667, 250167, 221967, 250467 (Low Sensitivity)

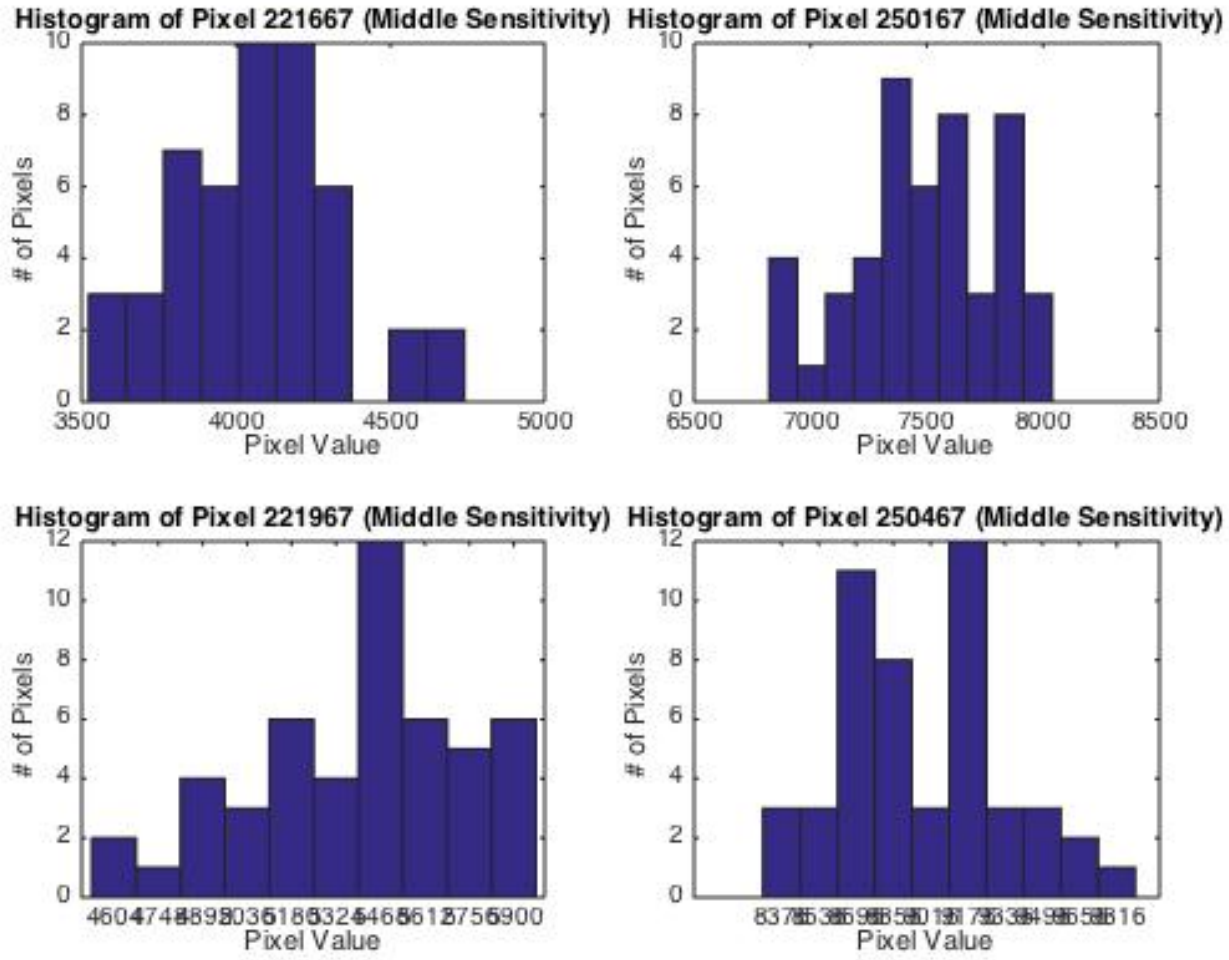


Figure 6. Histogram of pixel values: Pixel 221667, 250167, 221967, 250467 (Middle Sensitivity)

For the low sensitivity setting, the histograms of pixel value are like Gaussian distributions, which are shown in Figure 5. When using low sensitivity configured camera to capture picture, there will be less photon capture by the sensor for a shot. Therefore, ADC (quantization) noise and read noise (signal-independent) leads the noise distribution of the pixel, which can be an approximate Gaussian distribution.

2.3 Calculate the mean μ_i and variance σ_i^2 for each pixel in the image

The mean image can be calculated using the formula:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} I_i^{(j)} \approx (\phi_i \cdot t)g$$

The variance image can be calculated using the formula:

$$\sigma_i^2 = \frac{1}{N}\sum_{j=1}^{N} \left(I_i^{(j)} - \mu_i\right)^2$$

The MATLAB code below calculates the mean and variance images for a single sensitivity setting. After computing, the pixel values are arranged into a 950*700 matrix to form an image block.

```
for j = 1 : size(ImageMS(:,1))
    AvgImMS(j,1) = mean(ImageMS(j,:)); % Calculate mean image for 49 cropped images
    VarImMS(j,1) = var(ImageMS(j,:)); % Calculate variance image for 49 cropped
images
end
AvgImageMS = uint16(reshape(AvgImMS, [ ], 950)); % Rearrange the mean image into
1280*1024 picture
VarImageMS = uint16(reshape(VarImMS, [ ], 950)); % Rearrange the variance image
into 1280*1024 picture
```

The mean image, variance image and 2 original raw images of different sensitivity settings are shown in Figure 7, Figure 8 and Figure 9.

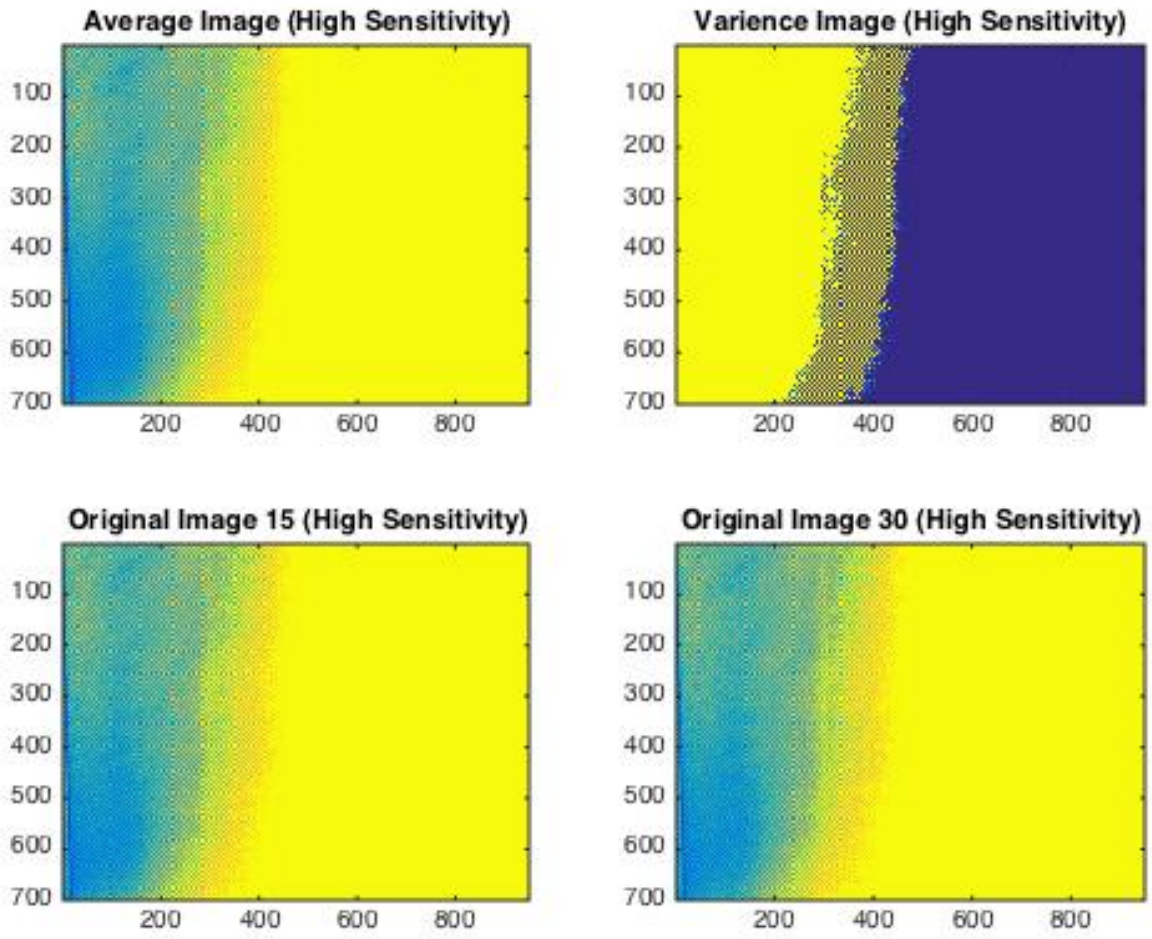


Figure 7. Mean image, variance image, #15 original image and #30 original image (High Sensitivity)

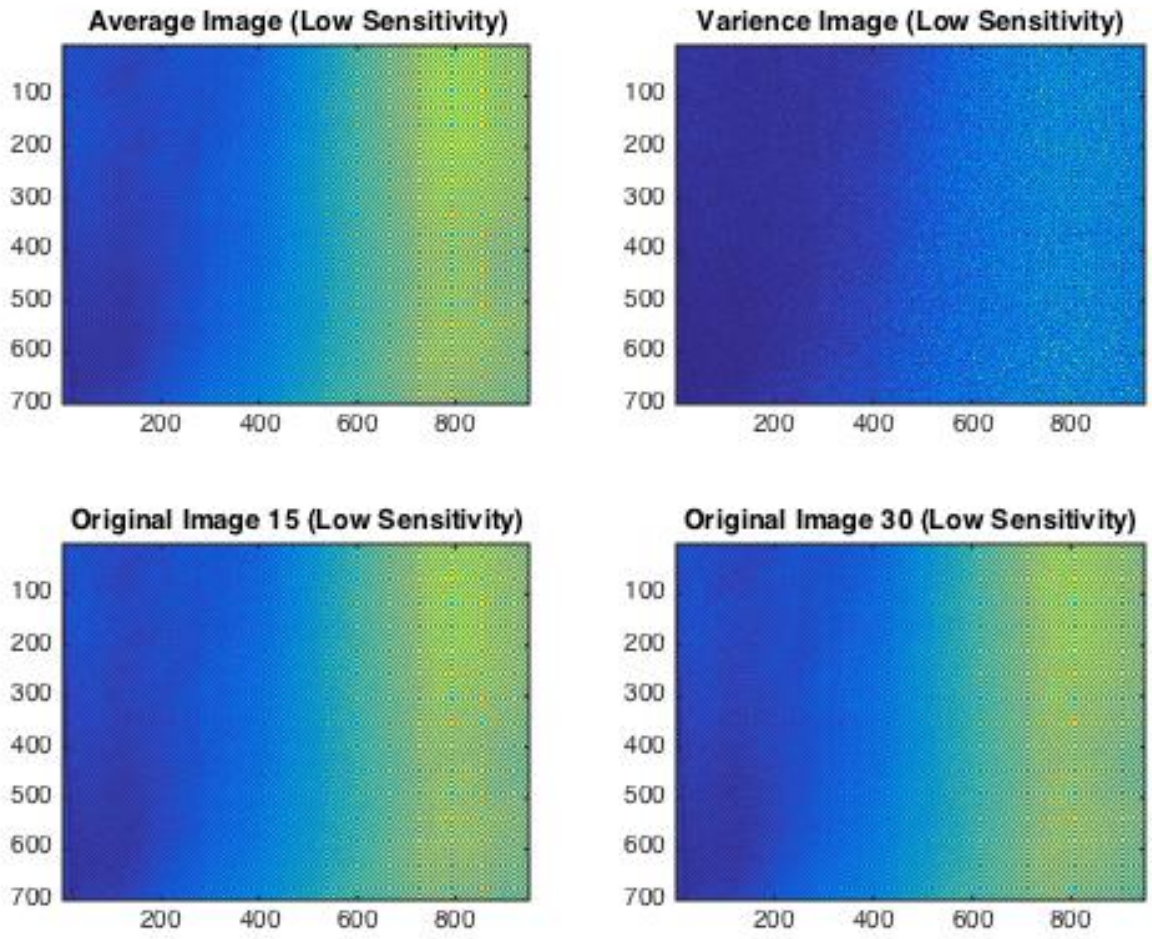


Figure 8. Mean image, variance image, #15 original image and #30 original image (Low Sensitivity)

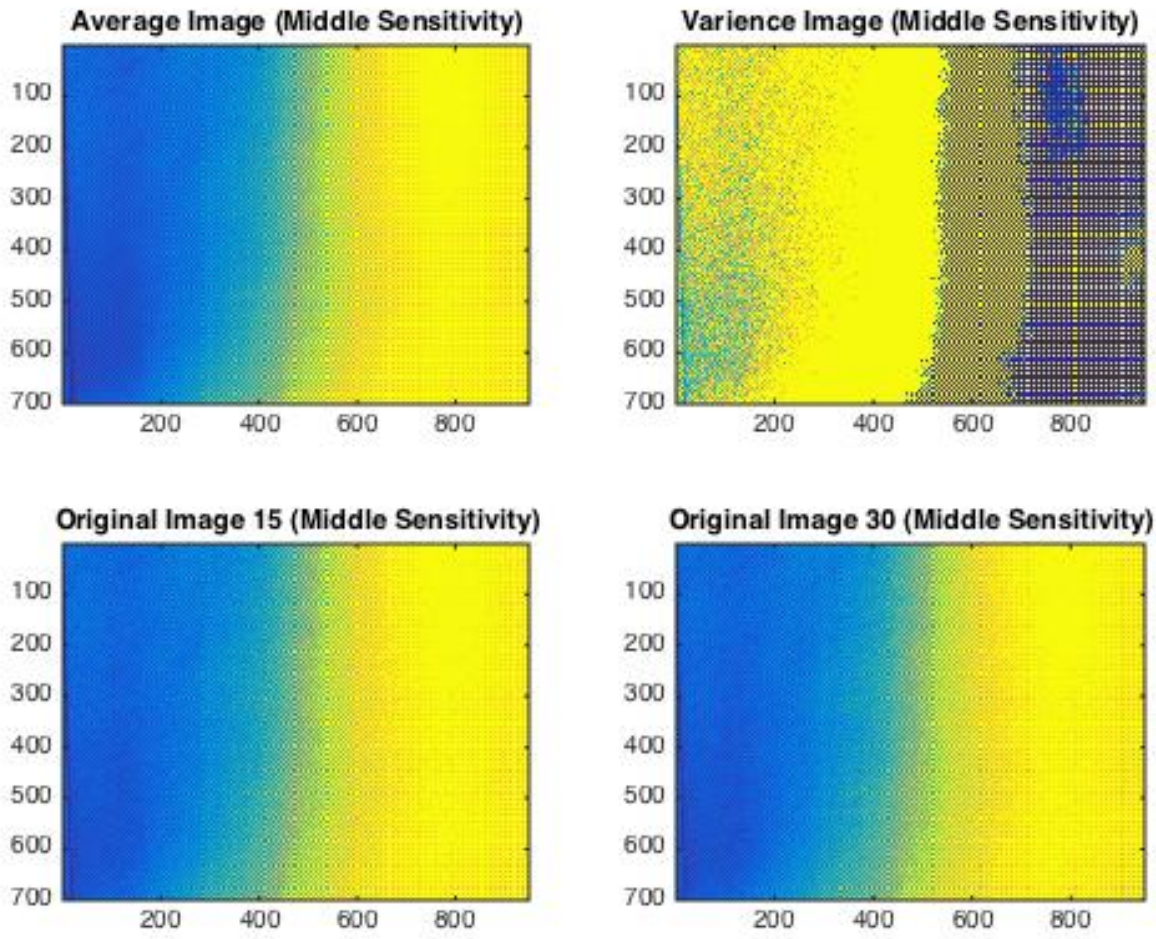


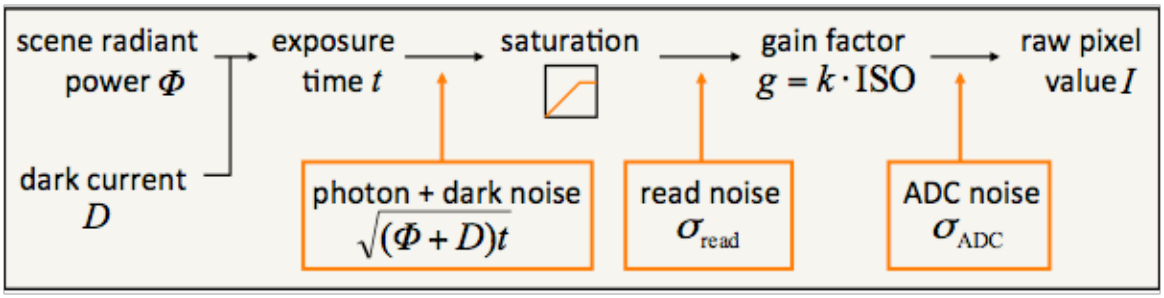
Figure 9. Mean image, variance image, #15 original image and #30 original image (Middle Sensitivity)

2.4 Plot the variance as a function of the mean

In order to plot variance as a function of the mean, we firstly round the mean pixel values to the nearest integers. Now, there may be a lot of repeated mean values. Then, we can discard the repeated mean values and average the variance values for each unique mean pixel value. Next, we discard the (mean, variance) pairs where the pixels are situated (different thresholds for different sensitivity settings: discard mean_high_sensitivity > 14000, mean_low_sensitivity >6000 and mean_middle_sensitivity > 12500), because

our Sensor Noise Model is built under the situation where there are no saturated pixels. By plotting (mean, variance) pairs, we can have the graphic representations for the variance as a function of the mean of different sensitivity settings. The (mean, variance) plots are shown in Figure 10 (High sensitivity), Figure 11 (Low sensitivity) and Figure 12 (Middle sensitivity), where the variance is approximately linearly related to mean value. As we can see in the figures, the linearity reduces as sensitivity increases. Recall that the Sensor Noise Model includes photon, read, and ADC noise:

$$\sigma_i^2 = (\phi_i \cdot t)g^2 + \sigma_{read}^2 \cdot g^2 + \sigma_{ADC}$$
$$= \mu_i \cdot g + \sigma_{read}^2 \cdot g^2 + \sigma_{ADC}^2$$



Sensor noise model

The MATLAB code is shown as below:

```
AvgvarMS = [round(AvgImMS),VarImMS]; % Round the mean values to the nearest integer
AvgvarMS = sortrows(AvgvarMS,1); % Sort the (mean,variance) pairs accroding to
ascending mean values
AvguniMS = unique(round(AvgImMS)); % Discard repeated mean values

for k = 1:length(AvguniMS)
    S = find(AvgvarMS(:,1)==AvguniMS(k));
    VaruniMS(k,1) = mean(AvgvarMS(S,2)); % Calculate the average variance for each
mean value
```

```
end
AvgvaruniMS(:,1) = AvguniMS;
AvgvaruniMS(:,2) = VaruniMS;


for t = 1:(sum((AvgvaruniMS(:,1)<=12500))) % Discard saturated pixels, where pixel
value > 12500)
    tAvgvaruniMS(t,:) = AvgvaruniMS(t,:);
end


%tAvgvaruniMS = AvgvaruniMS;

figureNo = figureNo+1;
figure(figureNo);

plot(tAvgvaruniMS(:,1),tAvgvaruniMS(:,2)); % Plot the variance as a function of
the mean
title('Mean vs. Variance (Middle Sensitivity)');
xlabel('Mean Pixel Value');
ylabel ('Variance Value');
```

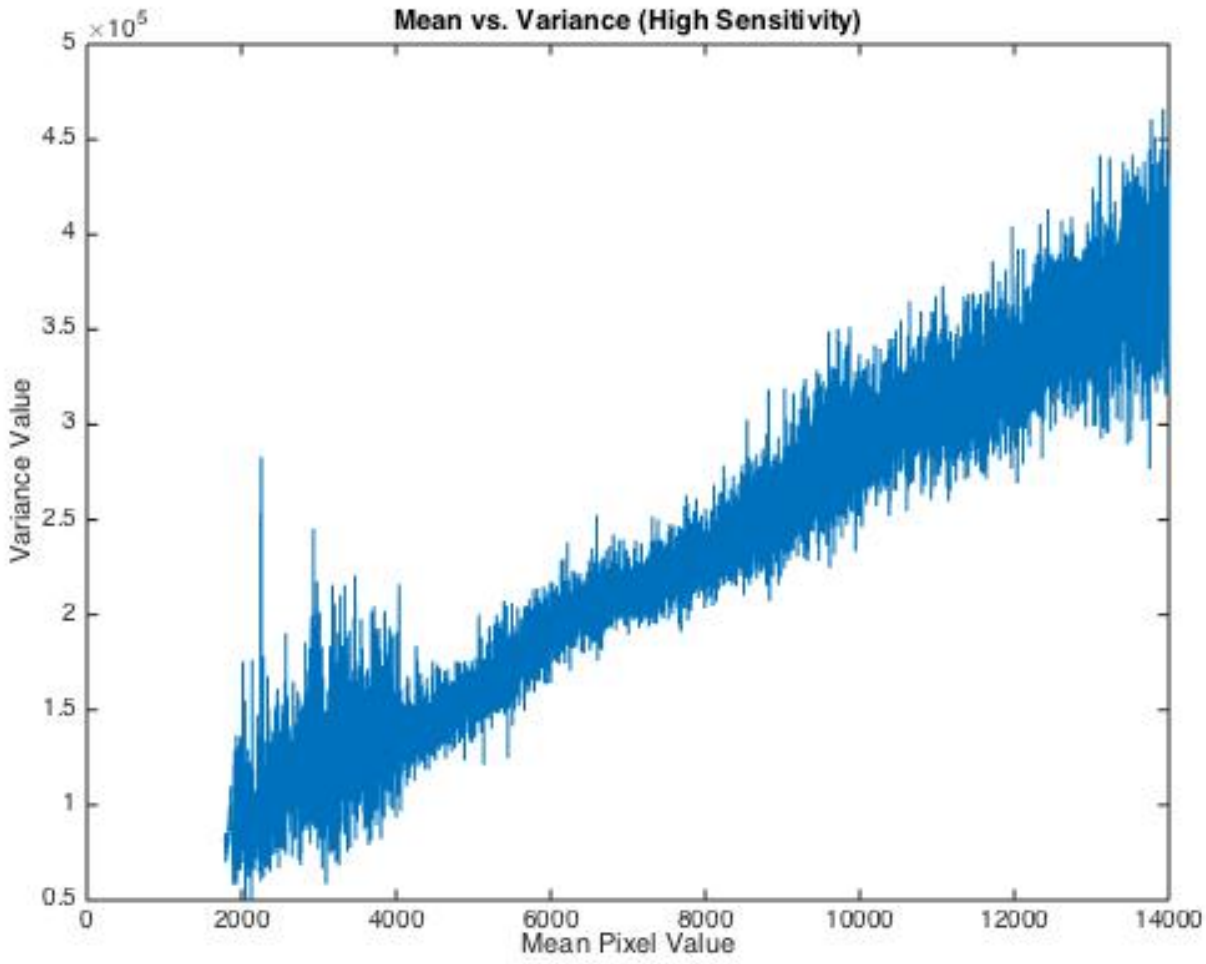


Figure 10. Mean vs. variance plot (High Sensitivity)

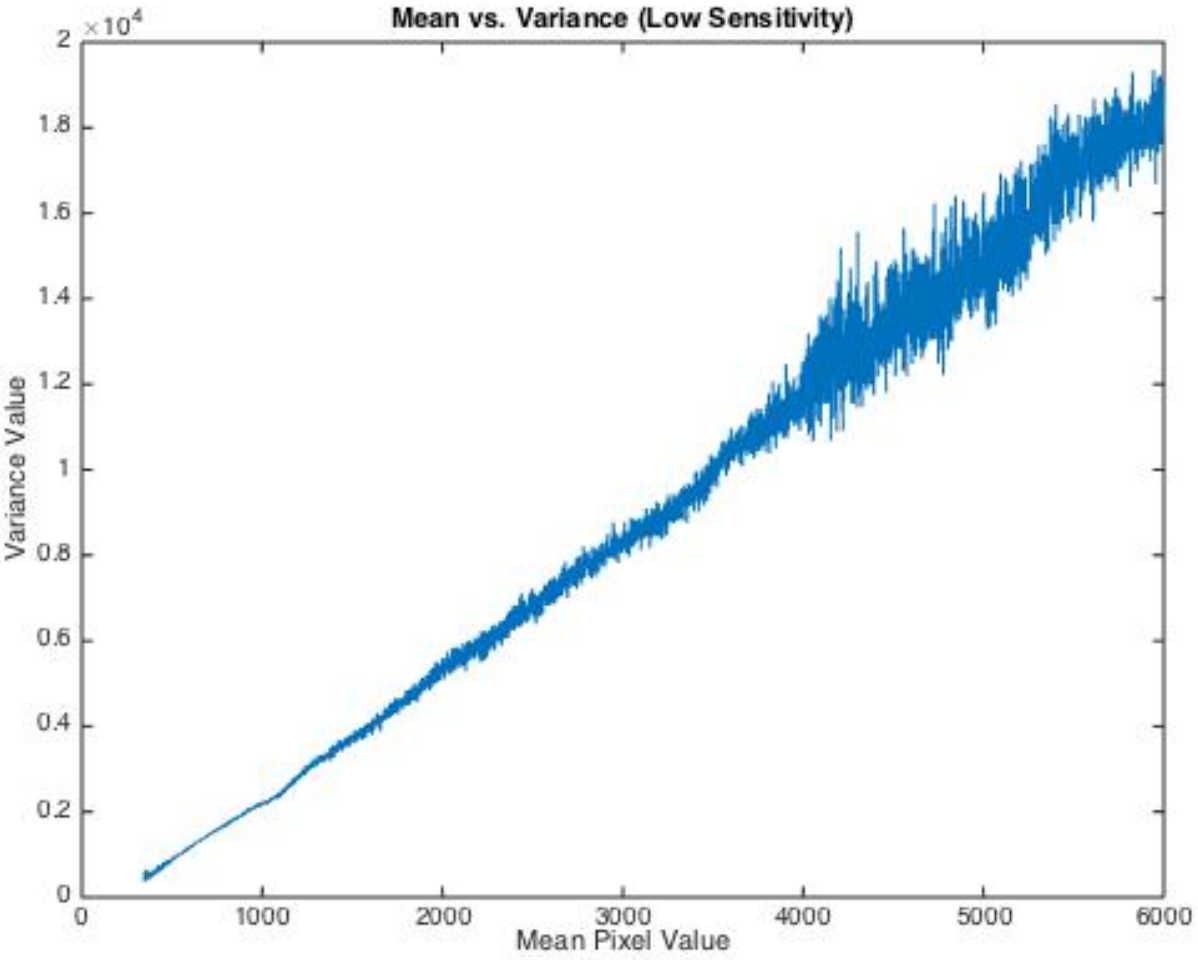


Figure 11. Mean vs. variance plot (Low Sensitivity)

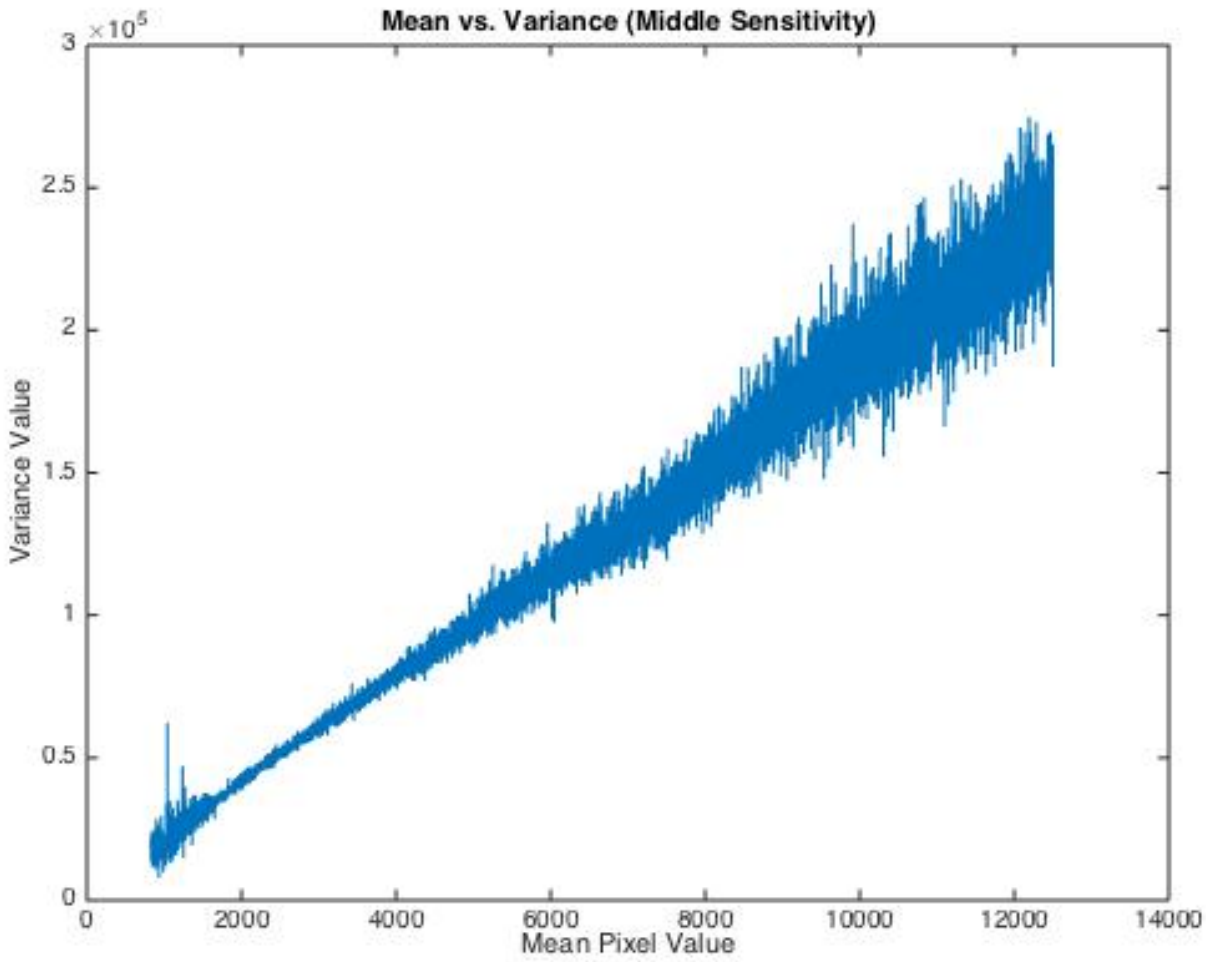


Figure 12. Mean vs. variance plot (Middle Sensitivity)

2.5 Fit a line to the plotted data

We can used the built-in MATLAB function polyfit to conduct linear regression (1st order) for (mean, variance) pairs. The fitting lines are the red lines shown in the Figure 13 (High Sensitivity), Figure 14 (Low Sensitivity) and Figure 15 (Middle Sensitivity), where the blue circles are (mean, variance) pairs.

```
figureNo = figureNo+1;
figure(figureNo);
scatter(tAvgvaruniMS(:,1),tAvgvaruniMS(:,2),'b'); % scatter plot (mean,
variance) pairs
hold on;
grid on;
LineMS = polyfit(tAvgvaruniMS(:,1),tAvgvaruniMS(:,2),1); % Linear Line fitting
```

```
x = (0:12500);
plot(x,LineMS(1)*x + LineMS(2),'r'); % plot linear regression line
title('Mean vs. Variance (Middle Sensitivity)');
xlabel('Mean Pixel Value');
ylabel('Variance Value');
legend('Mean vs. Variance Data','Linear Regression');
```

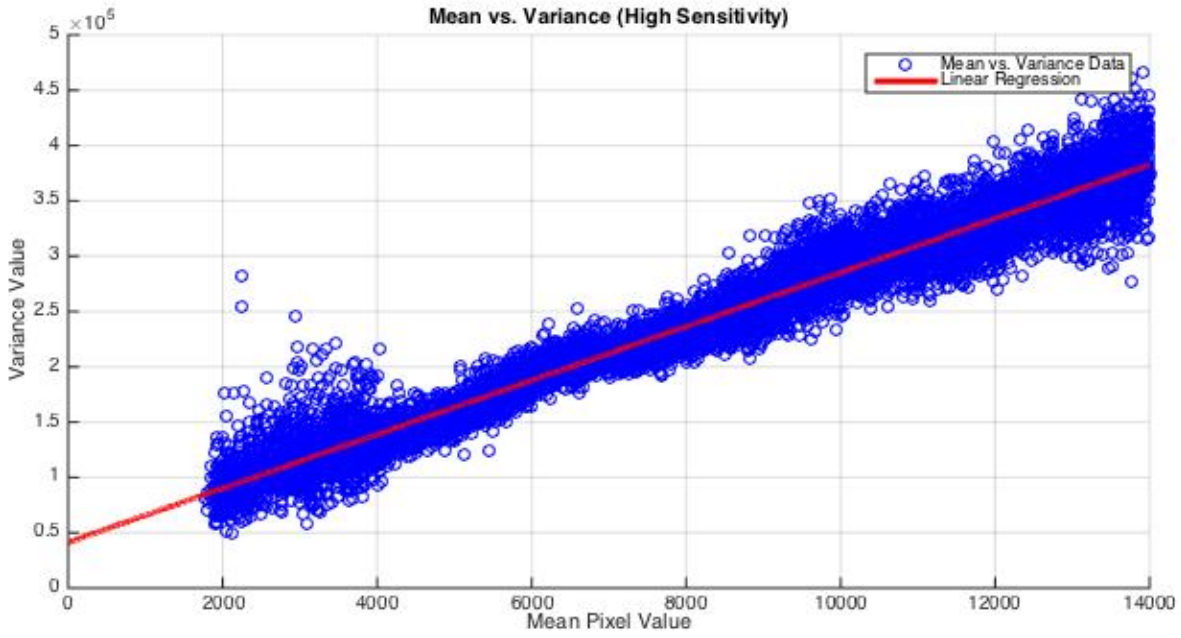


Figure 13. Linear line fitting for (mean, variance) pairs (High Sensitivity)

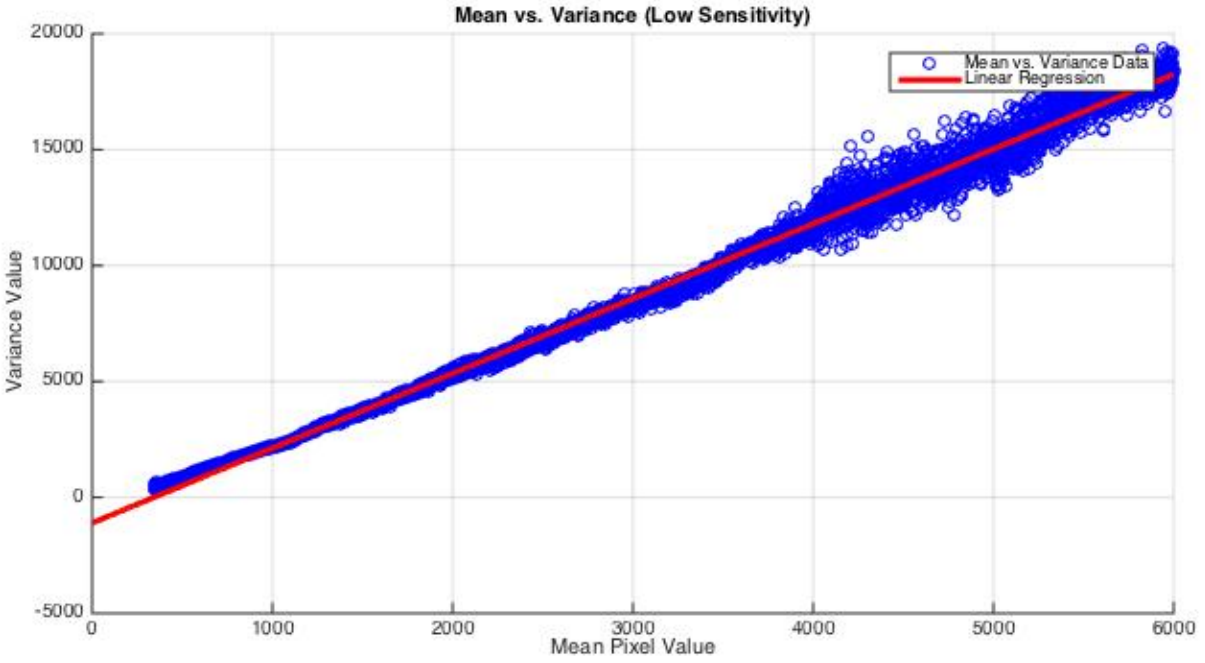


Figure 14. Linear line fitting for (mean, variance) pairs (Low Sensitivity)

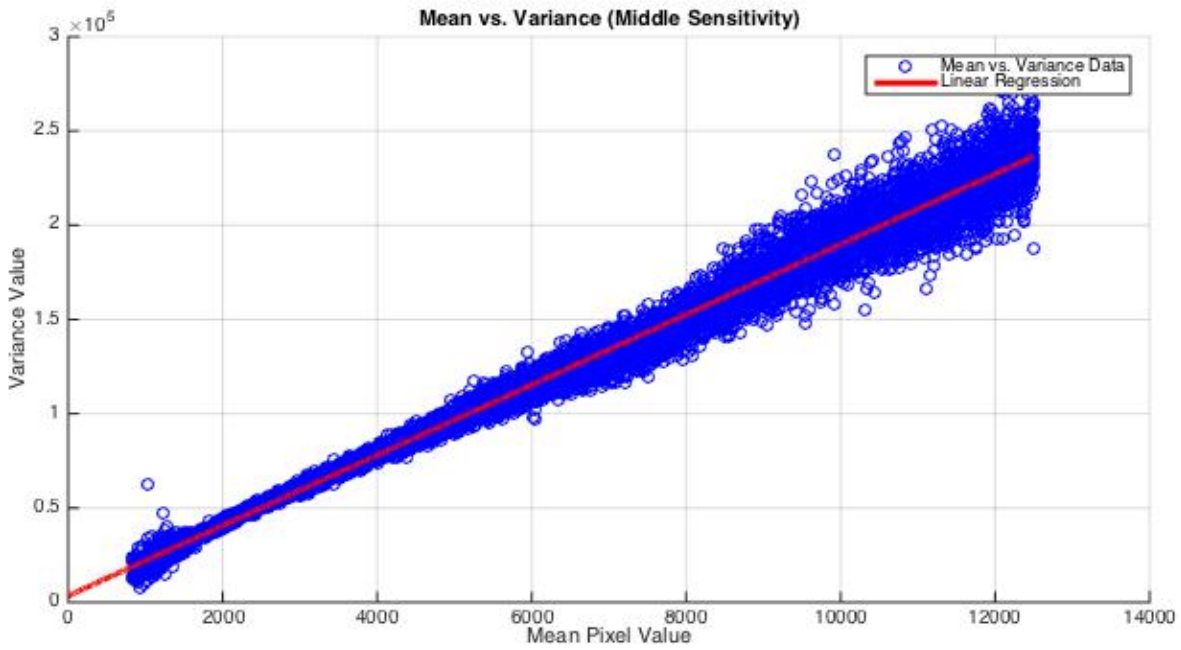


Figure 15. Linear line fitting for (mean, variance) pairs (Middle Sensitivity)

2.6 Use your fitted line data to calculate the camera gain *g*, the read noise variance σ_{read}^2 (measured in photo-electrons), and the ADC noise variance σ_{ADC}^2 (measured in gray levels).

Referring to:

$$\sigma_i^2 = (\phi_i \cdot t)g^2 + \sigma_{read}^2 \cdot g^2 + \sigma_{ADC}$$
$$= \mu_i \cdot g + \sigma_{read}^2 \cdot g^2 + \sigma_{ADC}^2$$

High Sensitivity Polyfit(linear) = [24.3868110570157 41366.8764342445]

Low Sensitivity Polyfit(linear) = [3.22595671775642 -1103.73022963516]

Middle Sensitivity Polyfit(linear) = [18.6526979912926 3398.17075294288]

Gain_high = 24.3868110570157 , K_high = 0.01524175691

Gain_low = 3.22595671775642 , K_low = 0.03225956718

Gain_middle = 18.6526979912926 , K_middle = 0.02194435058

σ_{read}^2 = 13.3383, σ_{ADC}^2 = -1242.53941354732

Problem: the linear regression line for high sensitivity setting yields a very high cost, which indicates it is not reliable to calculate σ_{read}^2 and σ_{ADC}^2 using the result of high sensitivity setting. However, if we using the results of low and middle sensitivity settings to solve the group of equations, the ADC noise is negative and that is impossible to have a negative power. So, that is a problem.

2.7 Signal-to-Noise Ratio (SNR)

$$SNR = \frac{\mu}{\sigma}$$

We can calculate Signal-to-Noise Ratio (SNR) using the MATLAB code for each sensitivity setting, where the SNR is obtained in logarithmic domain (dB):

```
SNRMS = 20*log10(tAvgvaruniMS(:,1)./sqrt(tAvgvaruniMS(:,2))); % Calculate SNR
in logarithmic domain
```

Figure 16, Figure 17 and Figure 18 are the plots of SNR as a function of mean pixel value using different sensitivity settings. We can find that, in plots, SNR increases as mean pixel value goes high; however, the incremental rate reduces. The mean pixel value closely associates with signal power. A high signal power indicates

that tons of photons are captures by the sensor, companying with a high photon noise (signal-dependent). When the pixel value is low, the noise is mainly consisted of read noise and ADC noise. Although the SNR is relative low at that point, the gradient of the curve is high.

For the high sensitivity setting, sensors are sensitive to photons and more susceptible to signal-dependent noise (photon noise). With the low sensitivity setting, the camera sensors are influenced by signal-independent noise (read noise and ADC noise). Note that, the signal-independent noises also contribute to the total noise in high sensitivity setting. As the result, the mean-SNR curve of high sensitivity setting is below that of middle sensitivity setting, which is below the curve of low sensitivity setting. Refer to the noise equation:

$$\sigma_i^2 = (\phi_i \cdot t)g^2 + \sigma_{read}^2 \cdot g^2 + \sigma_{ADC}$$
$$= \mu_i \cdot g + \sigma_{read}^2 \cdot g^2 + \sigma_{ADC}^2$$

Maximum SNR value for low sensitivity setting is 33.2750 dB

Maximum SNR value for middle sensitivity setting is 29.2145 dB

Maximum SNR value for high sensitivity setting is 28.3489 dB

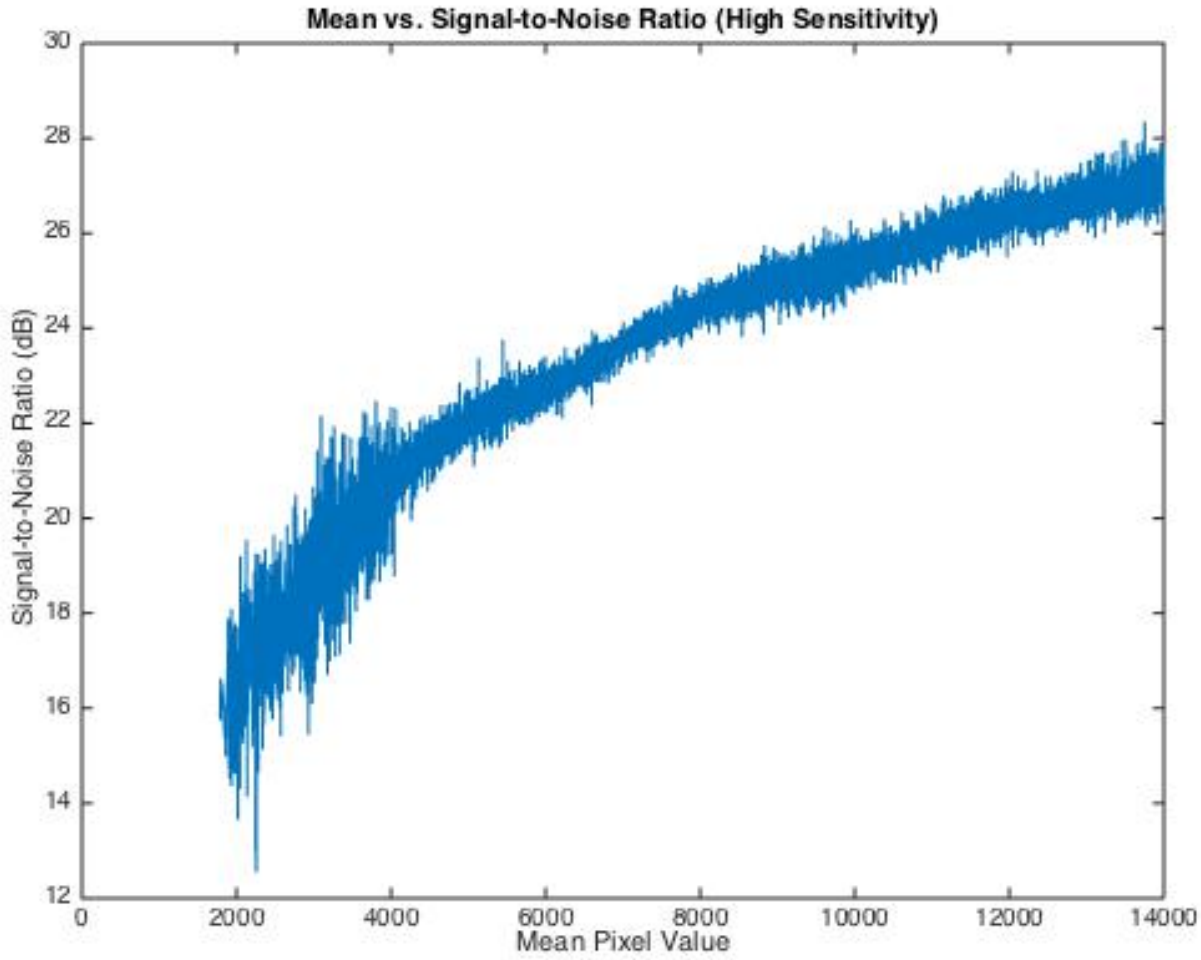


Figure 16. Mean vs. SNR (High Sensitivity), where max SNR = 28.3489 dB

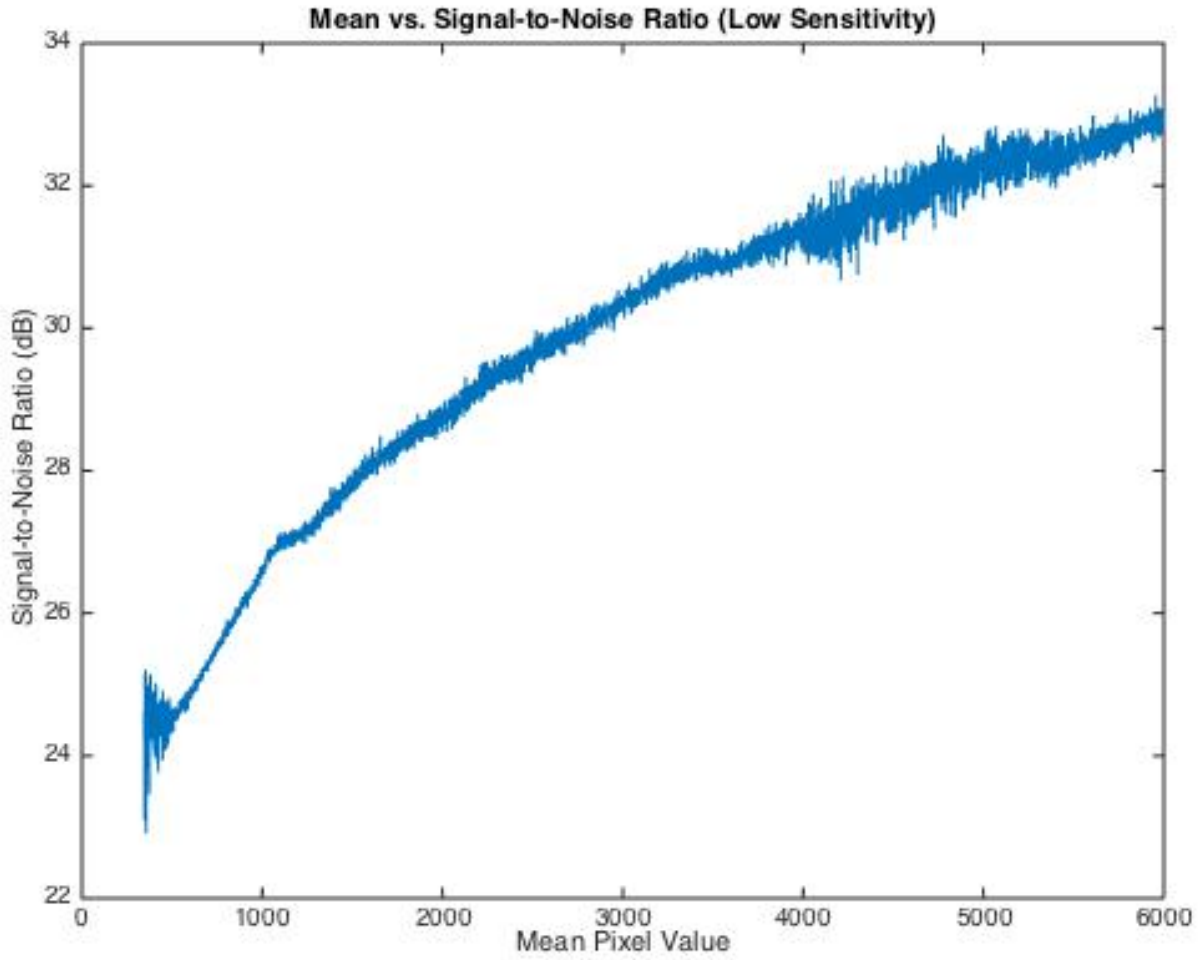


Figure 17. Mean vs. SNR (Low Sensitivity), where max SNR = 33.2750 dB

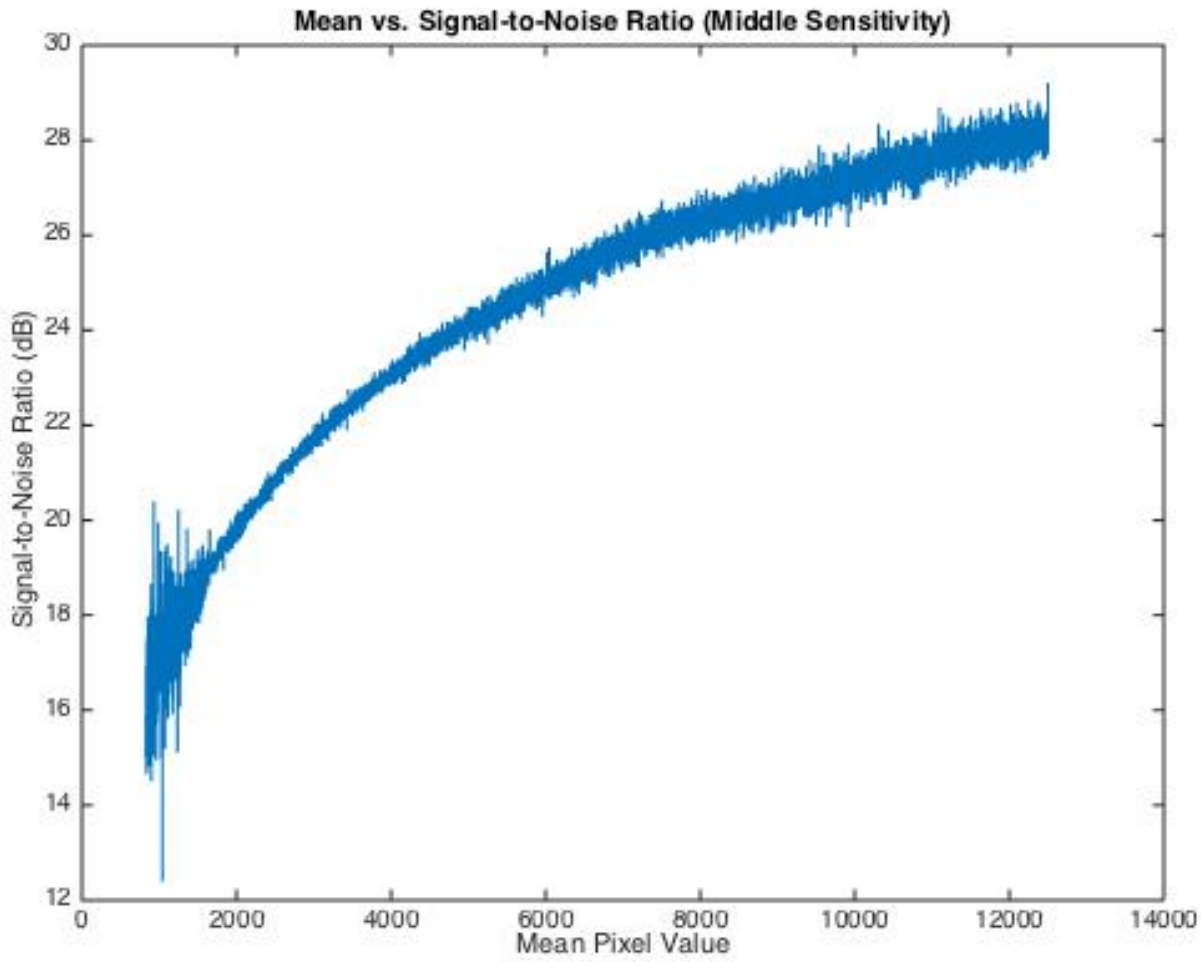


Figure 18. Mean vs. SNR (Middle Sensitivity), where max SNR = 29.2145 dB