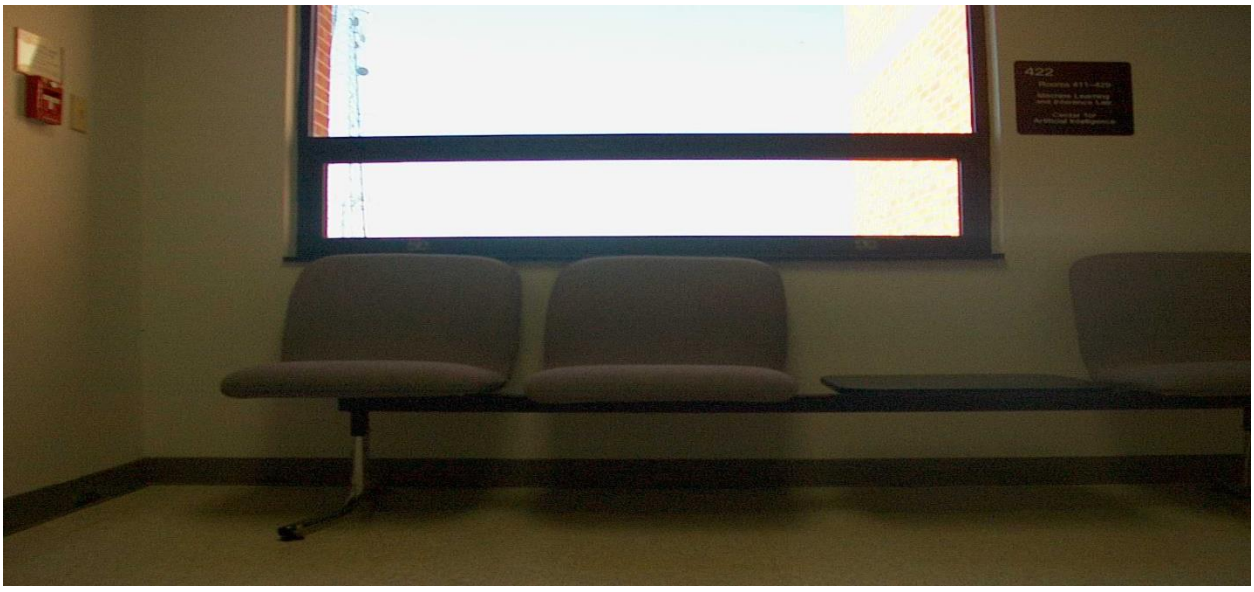
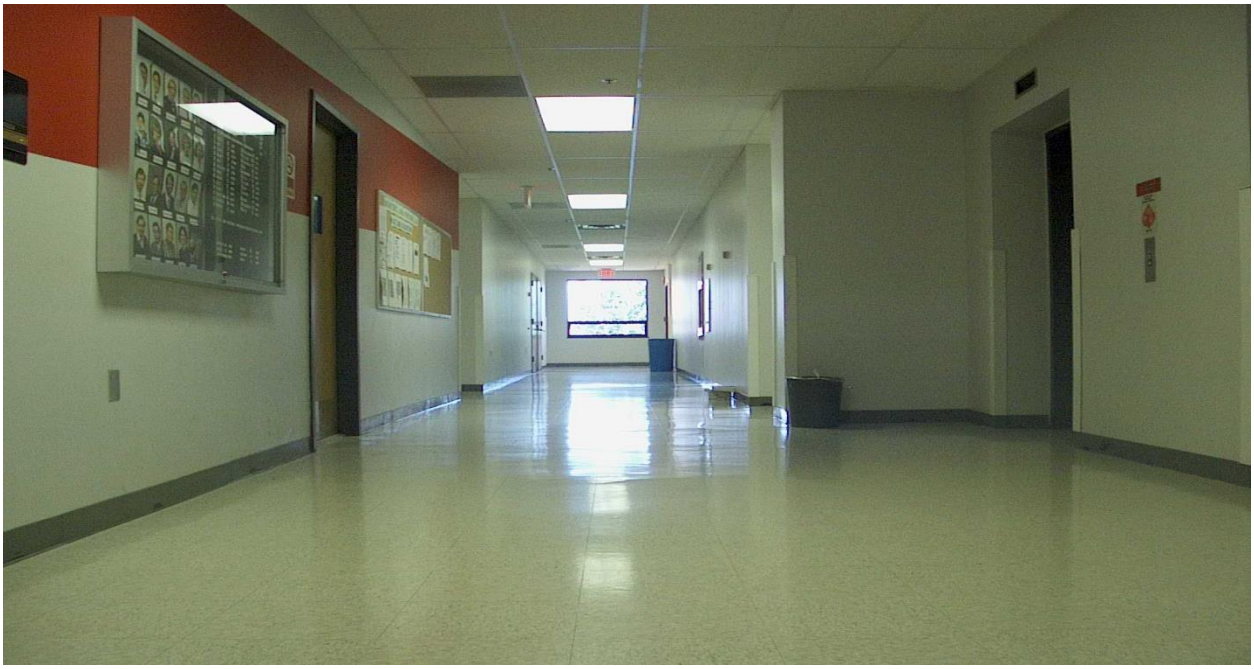
CS 682 Home Work III
KRISHNA SINDHURI VEMURI
G01024066

I have used the 4001 and 4050 images for illustrating the results of the program.



ST2MainHall4001.jpg



ST2MainHall4050.jpg

To compute the gray level edges for the images, first we need to convert the original image into gray.

Gray4001.jpg

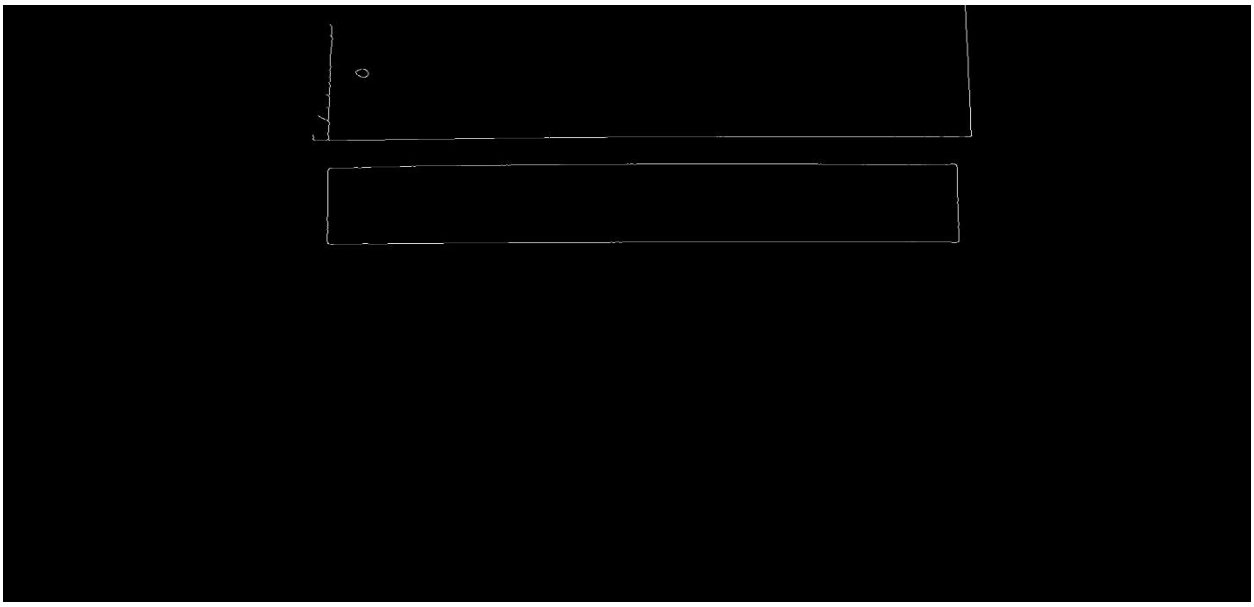
Gray4050.jpg

Then I have blurred all the images using the Gaussian Filter with kernel width and height as 7 pixels and sigma as 2.

Blur4001.jpg

Blur4050.jpg

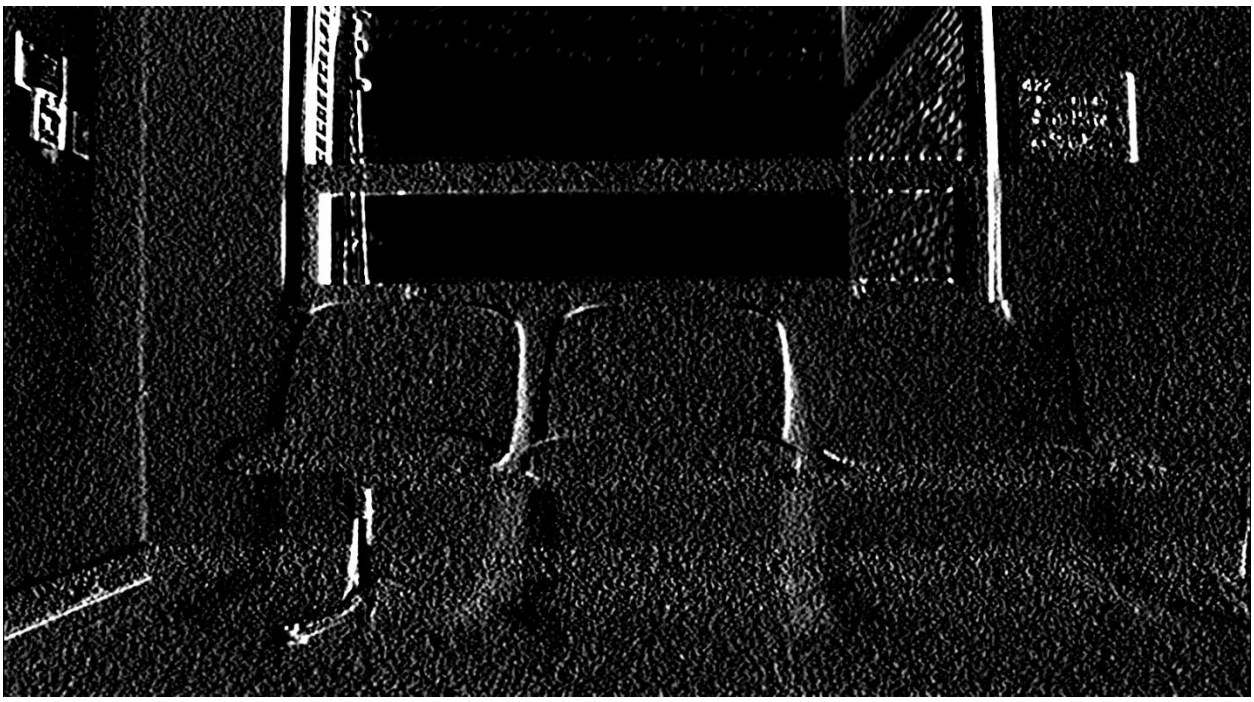
After blurring the image most of the noise is removed and using the canny edge detection algorithm with low threshold as 100 and high threshold as 200, we detect the edges in all the images.



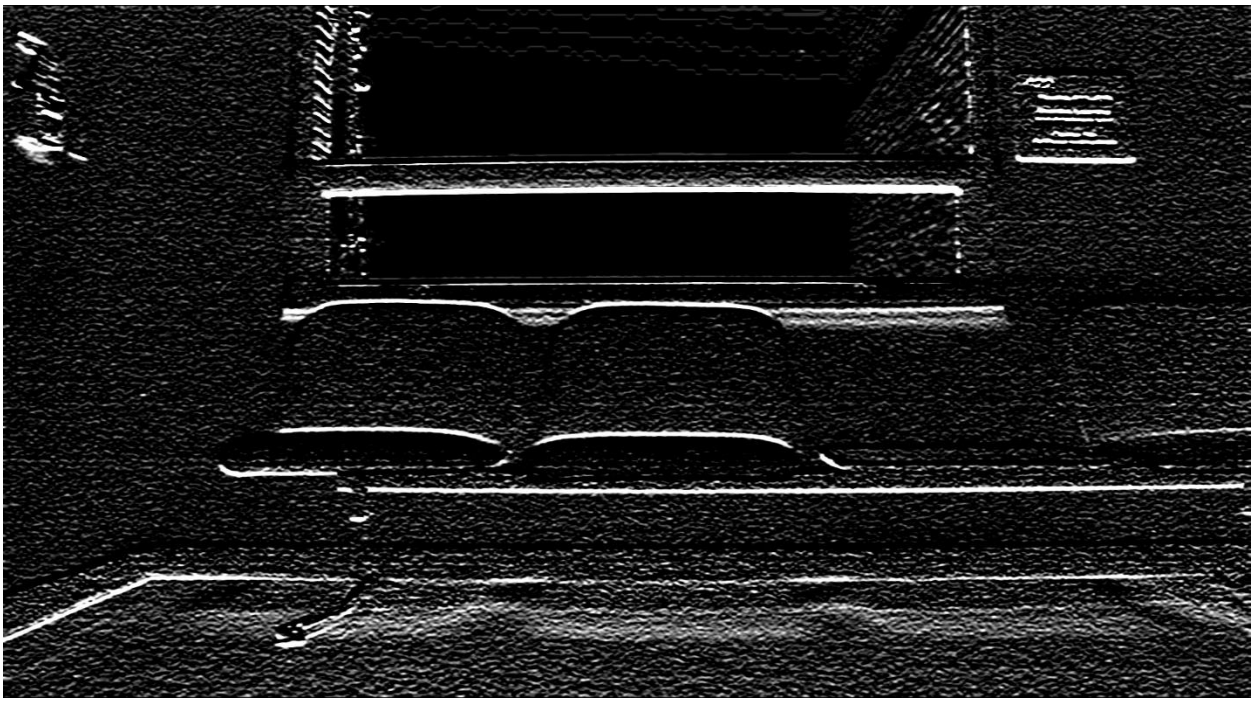
Canny4001.jpg

Canny4050.jpg

Also, we find the X and Y gradients of the blurred images.

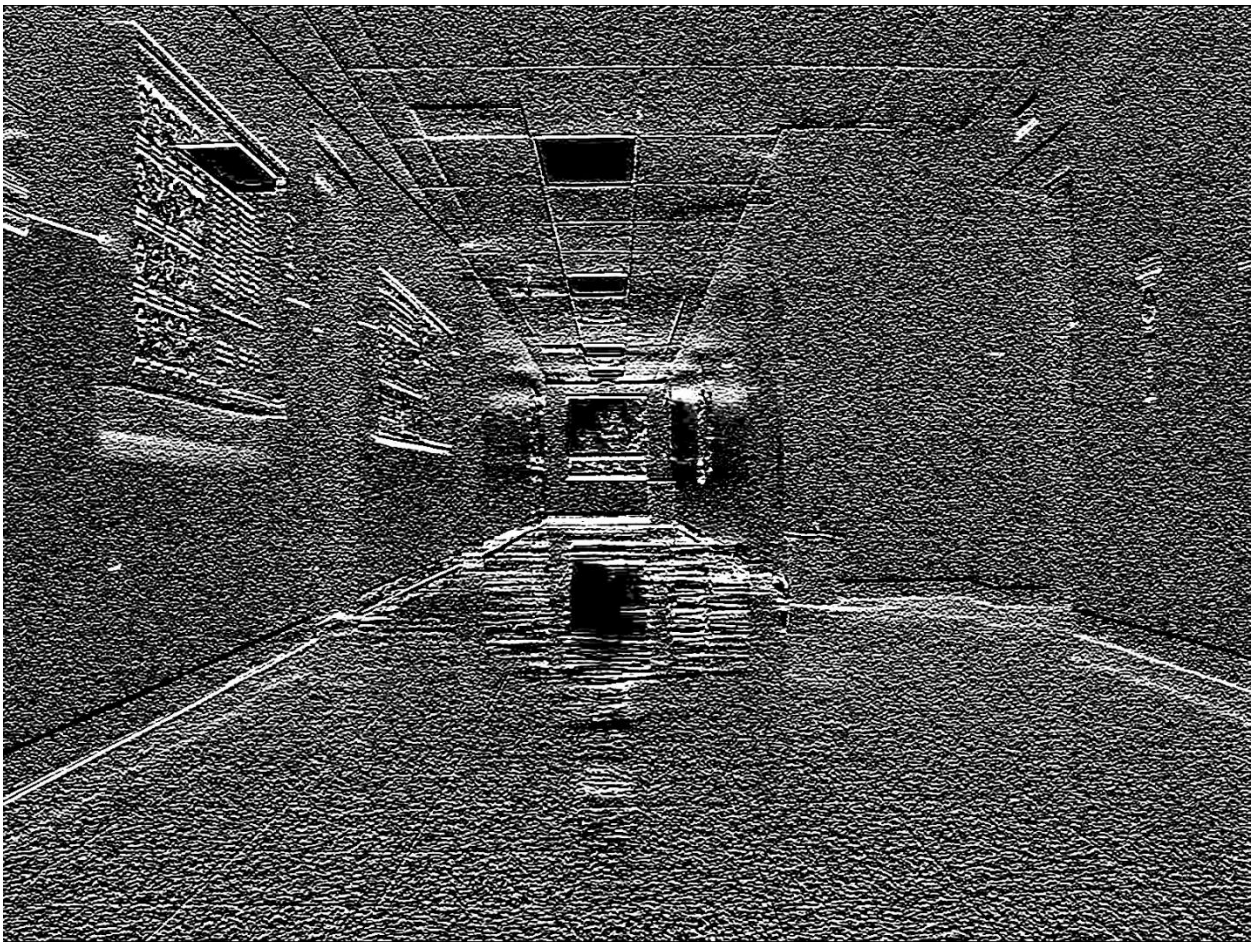


Gx4001.jpg



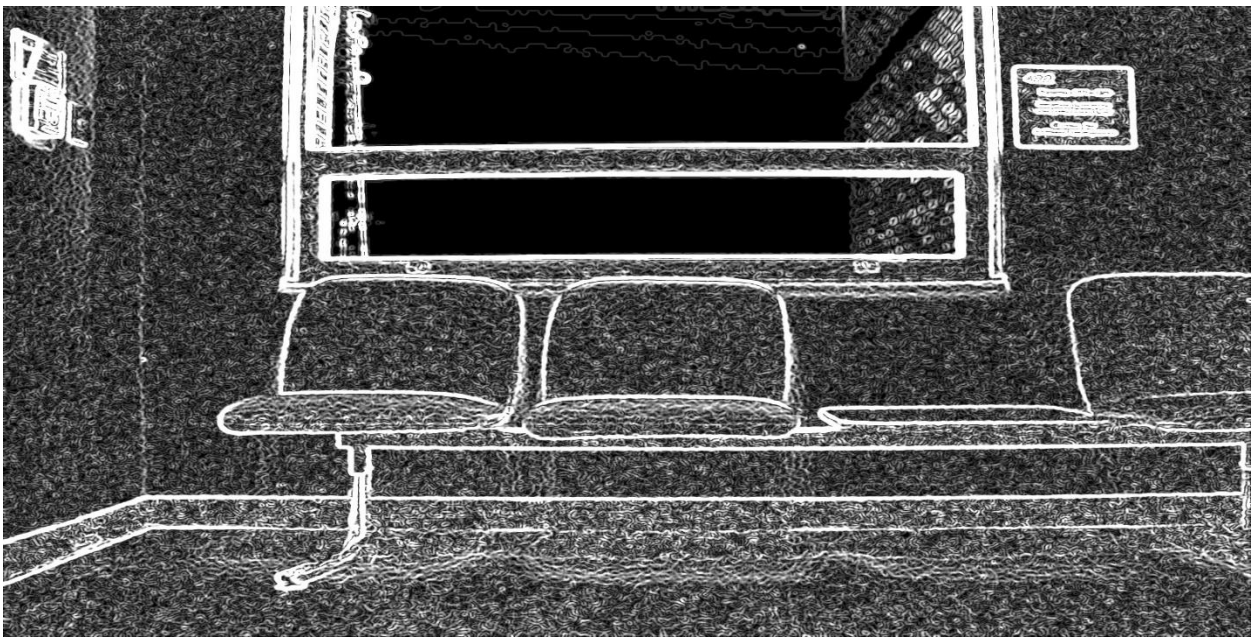
Gy4001.jpg

Gx4050.jpg

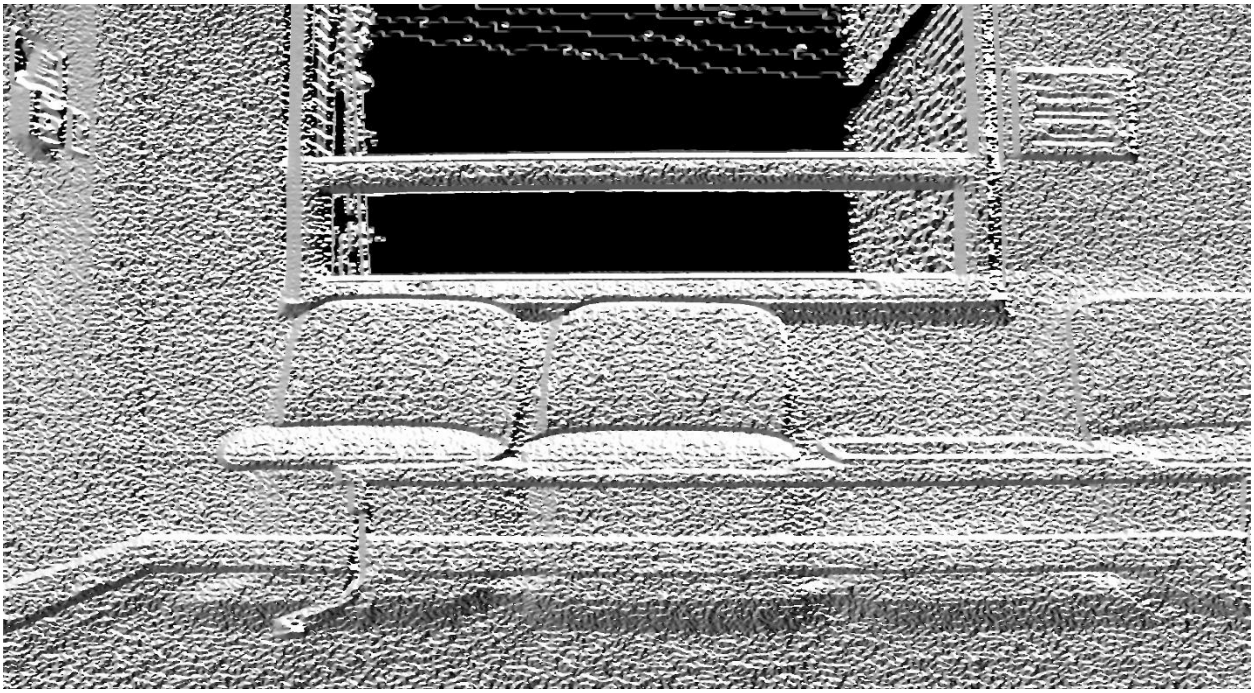


Gy4050.jpg

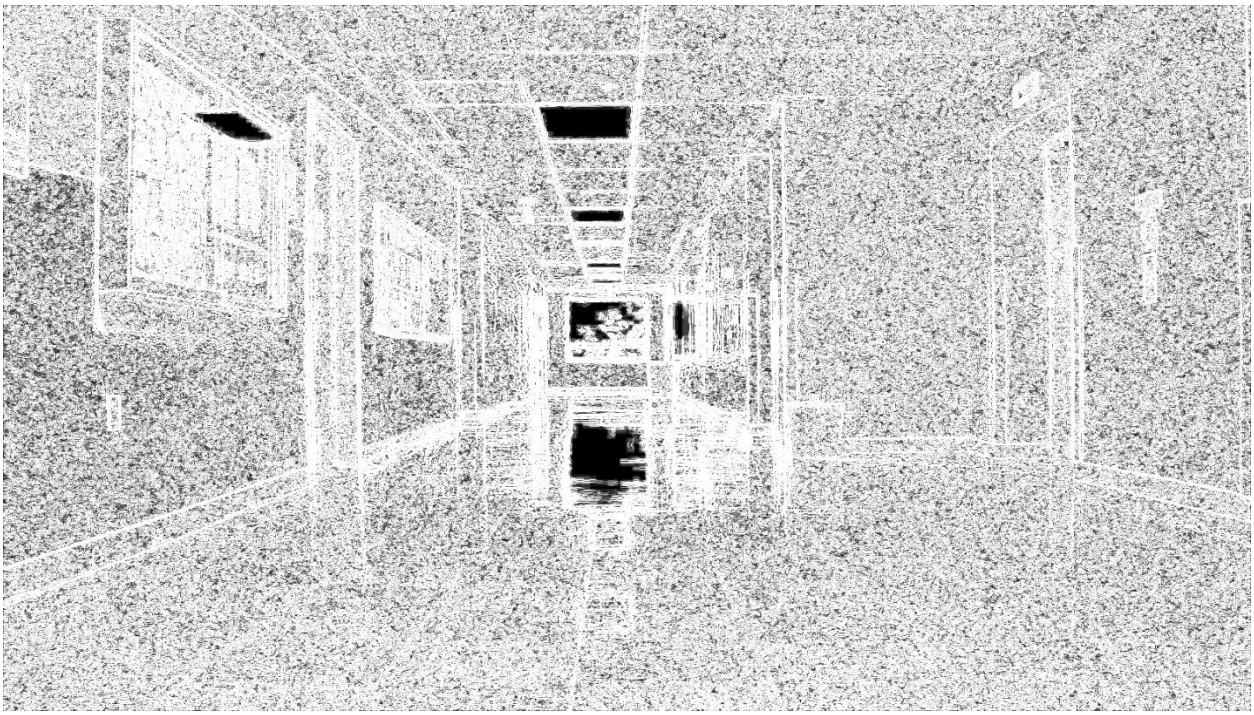
Using the cartToPolar built in function from OpenCV, we calculate the magnitude and phase of the gradients.



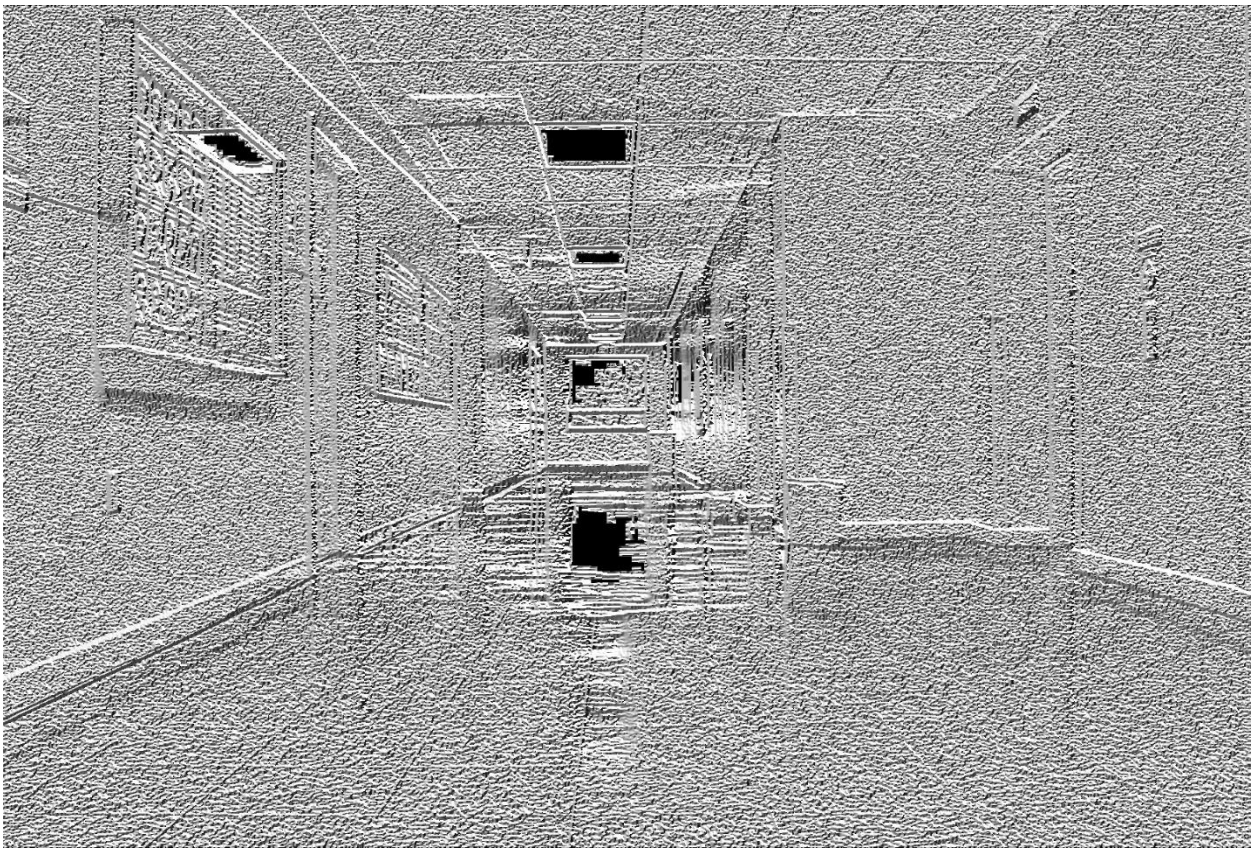
Magnitude4001.jpg



Phase4001.jpg

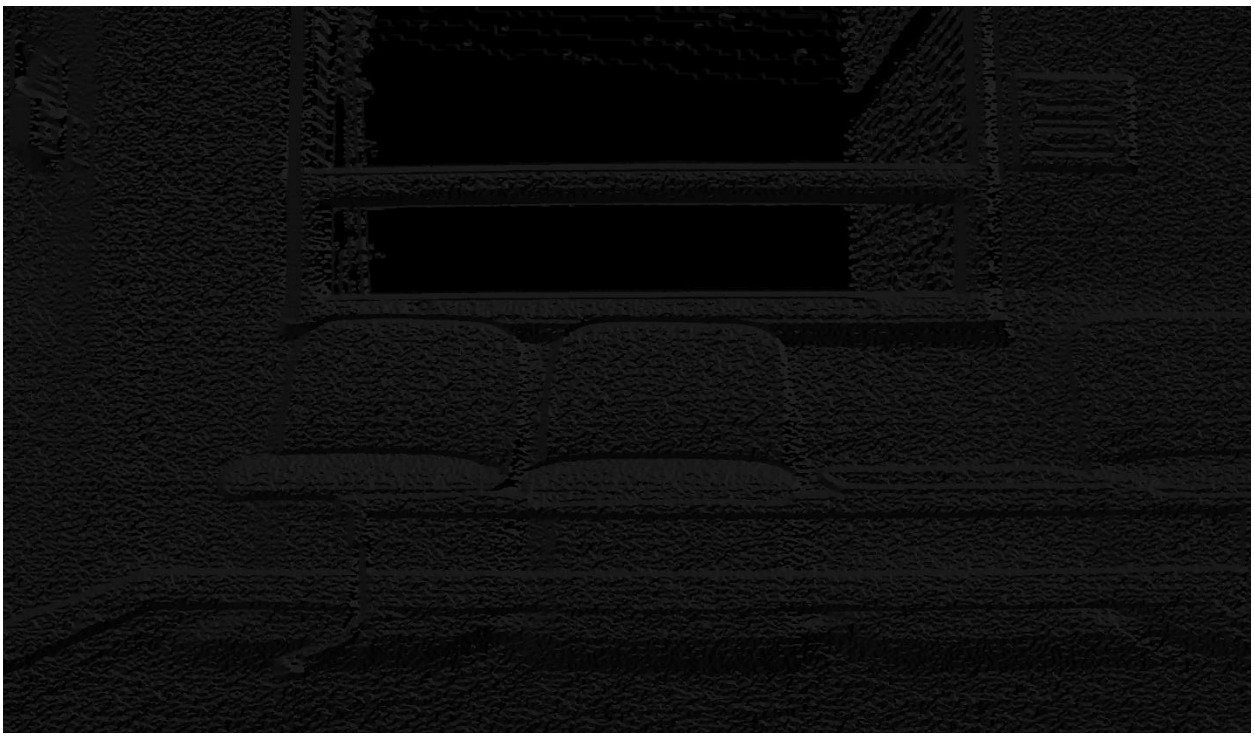


Magnitude4050.jpg

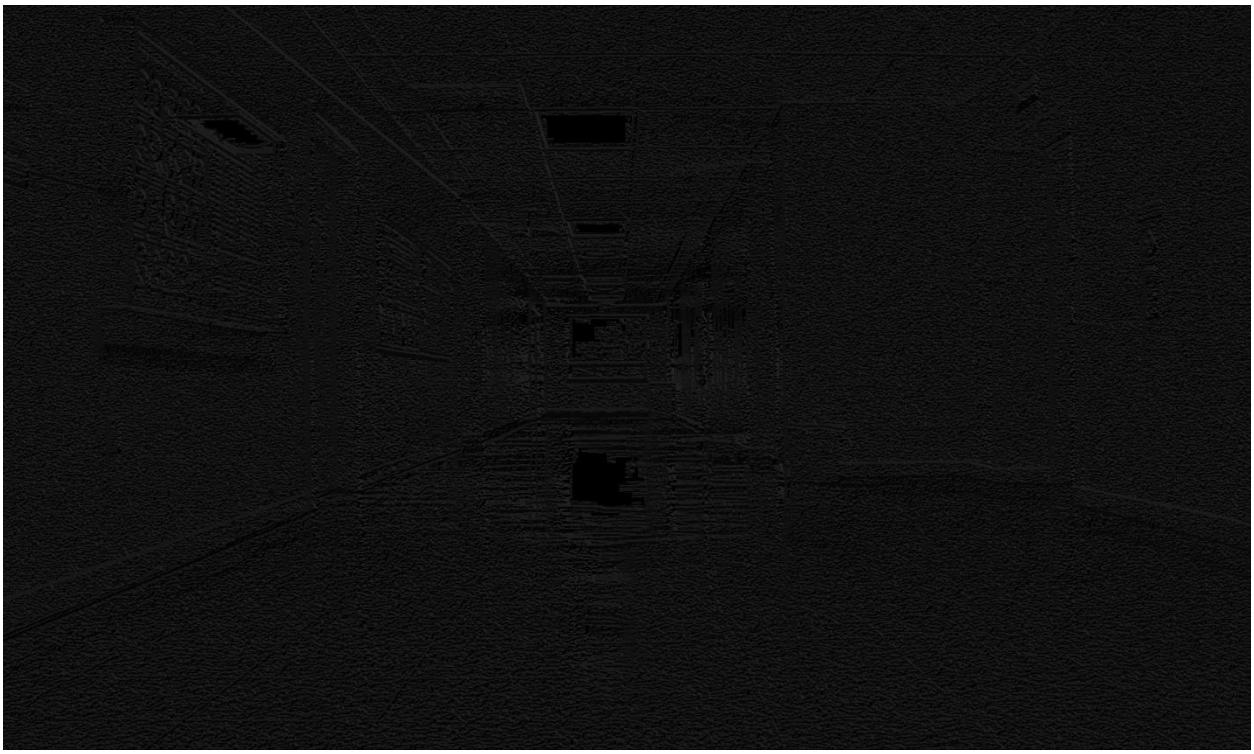


Phase4050.jpg

Also, we divide and round the phase to make it to fall in 36-bin range.

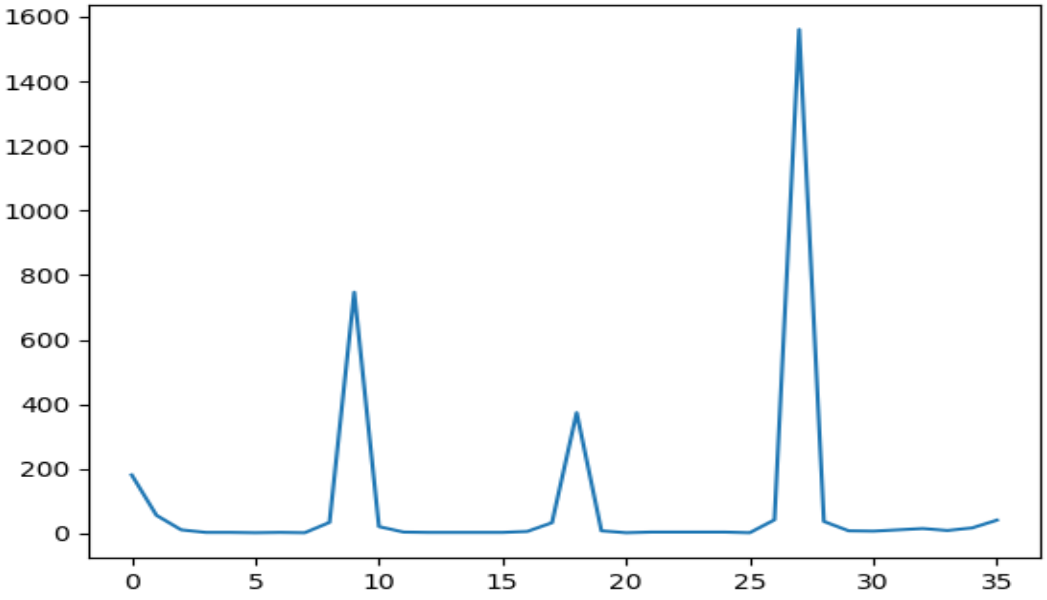


Angle4001.jpg

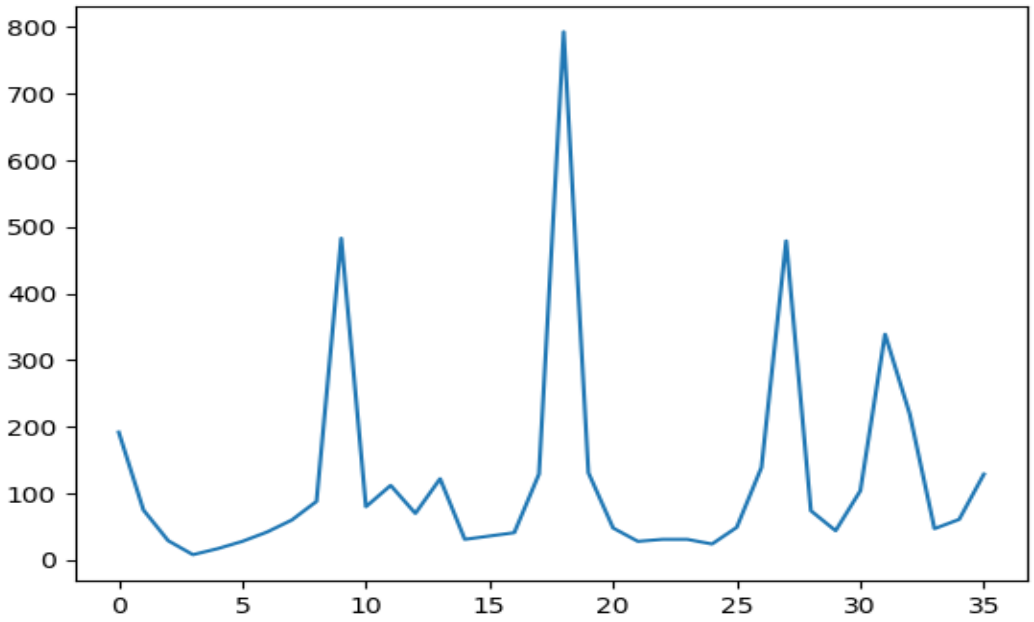


Angle4050.jpg

Later we use the canny edge detection image as a mask and plot histograms for the angles in the images. This masking helps to eliminate all the unwanted edges into consideration.



GrayHist4001.jpg



Gray4050.jpg

To compute the color level edge histograms, we first blur the original images with the same parameters given for the gray images.

We split the blur image into the RGB channels.

R4001.jpg

G4001.jpg

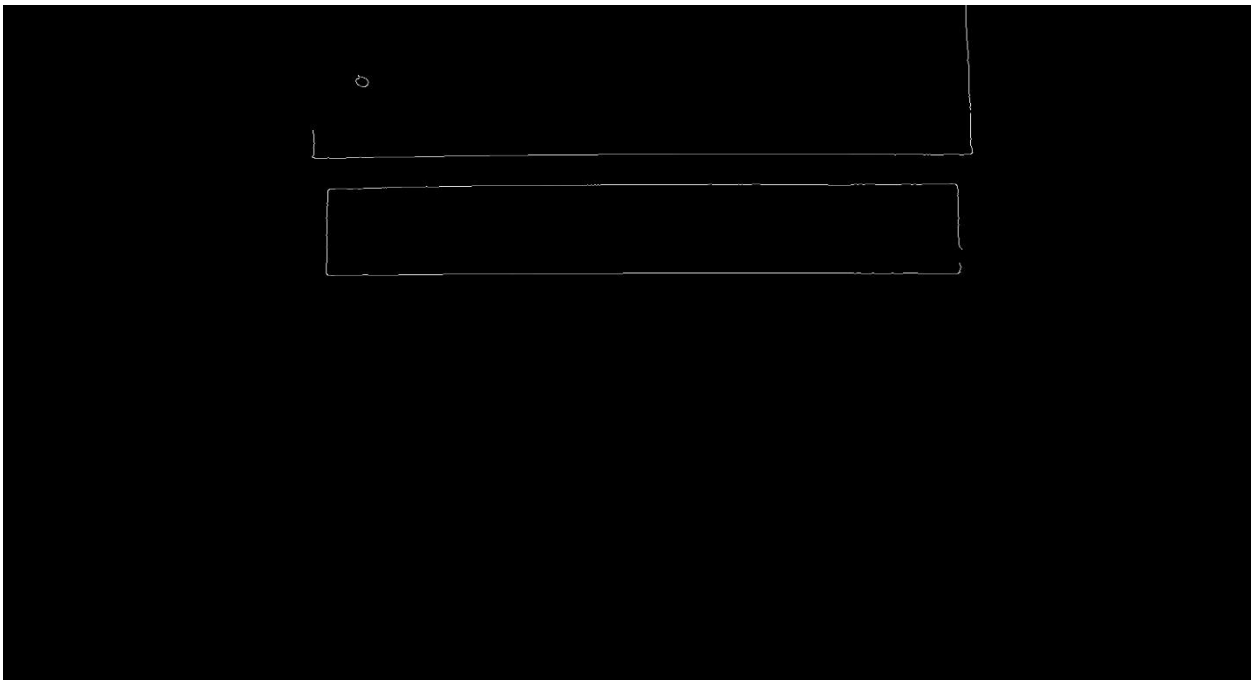
B4001.jpg

R4050.jpg

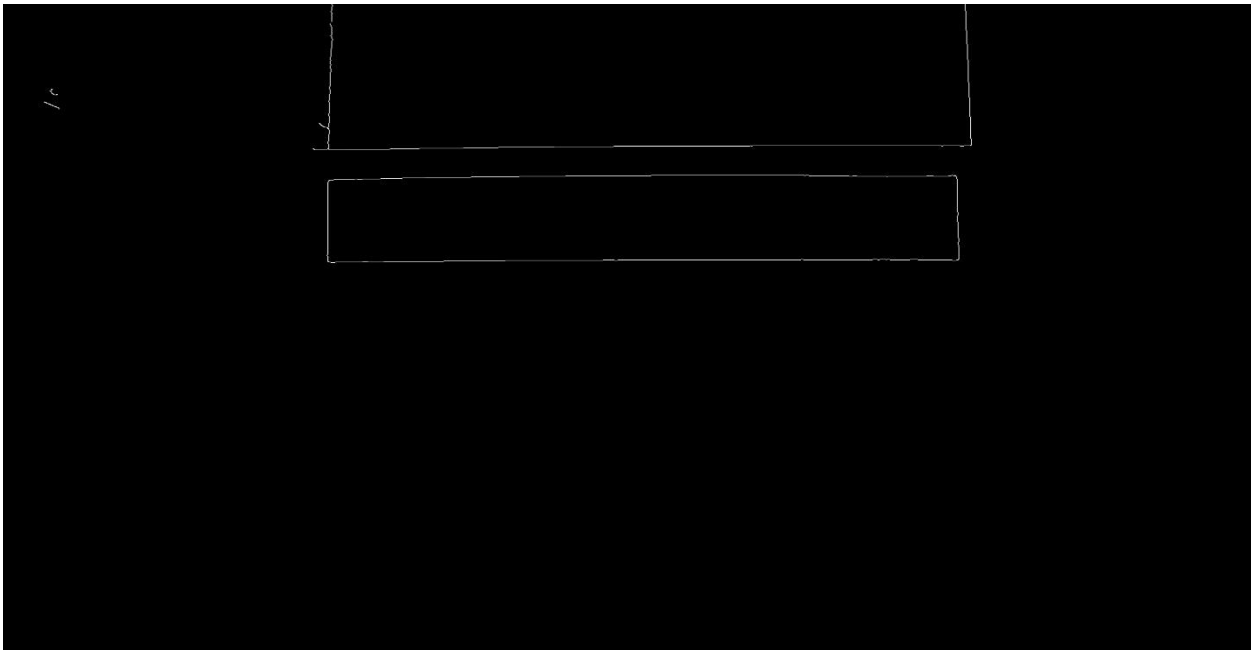
G4050.jpg

B4050.jpg

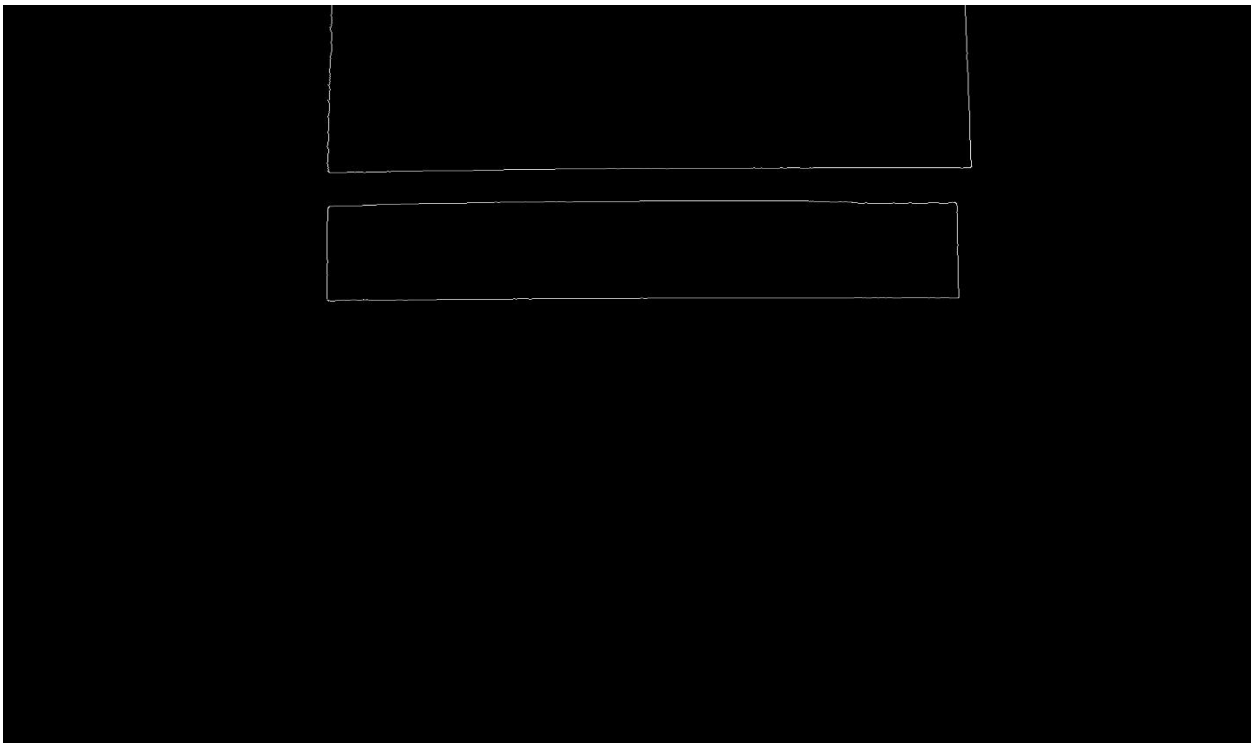
We find the canny edges for all the three channels by giving the lower threshold as 100 and higher threshold as 200.



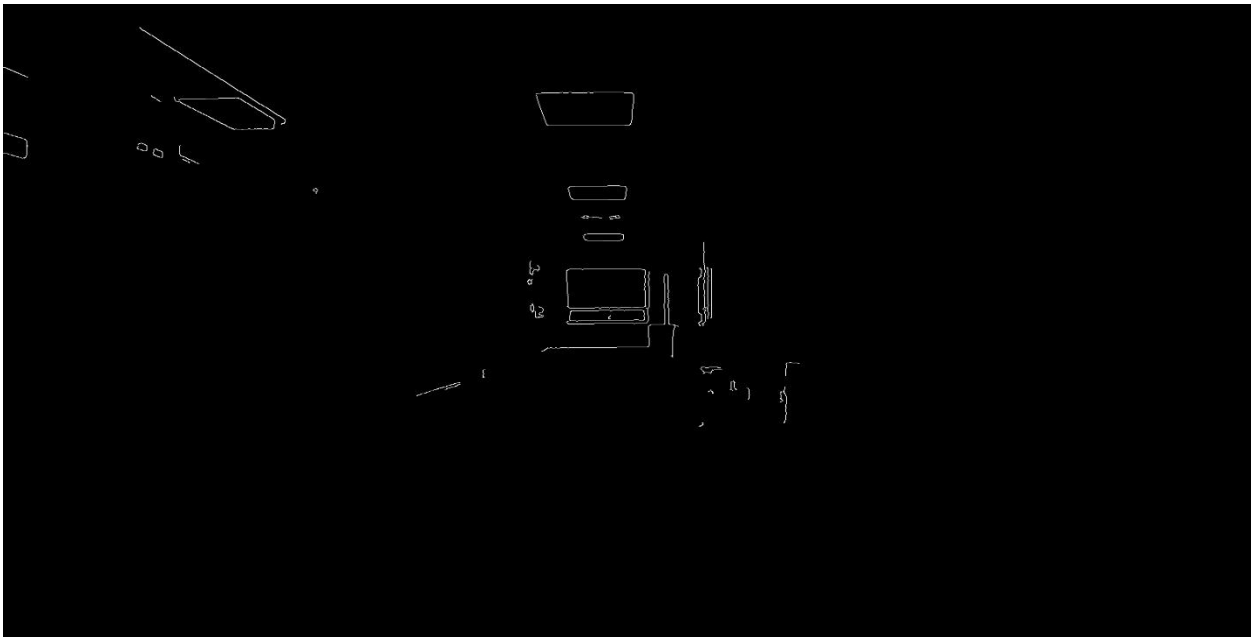
RCanny4001.jpg



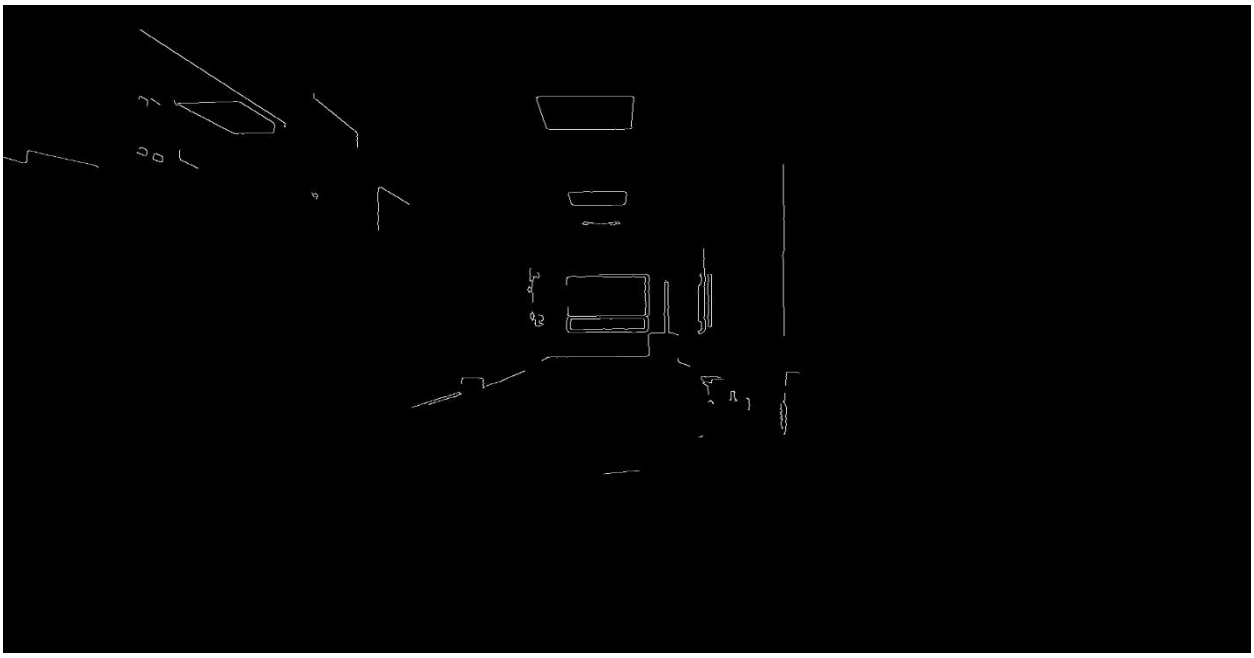
GCanny4001.jpg



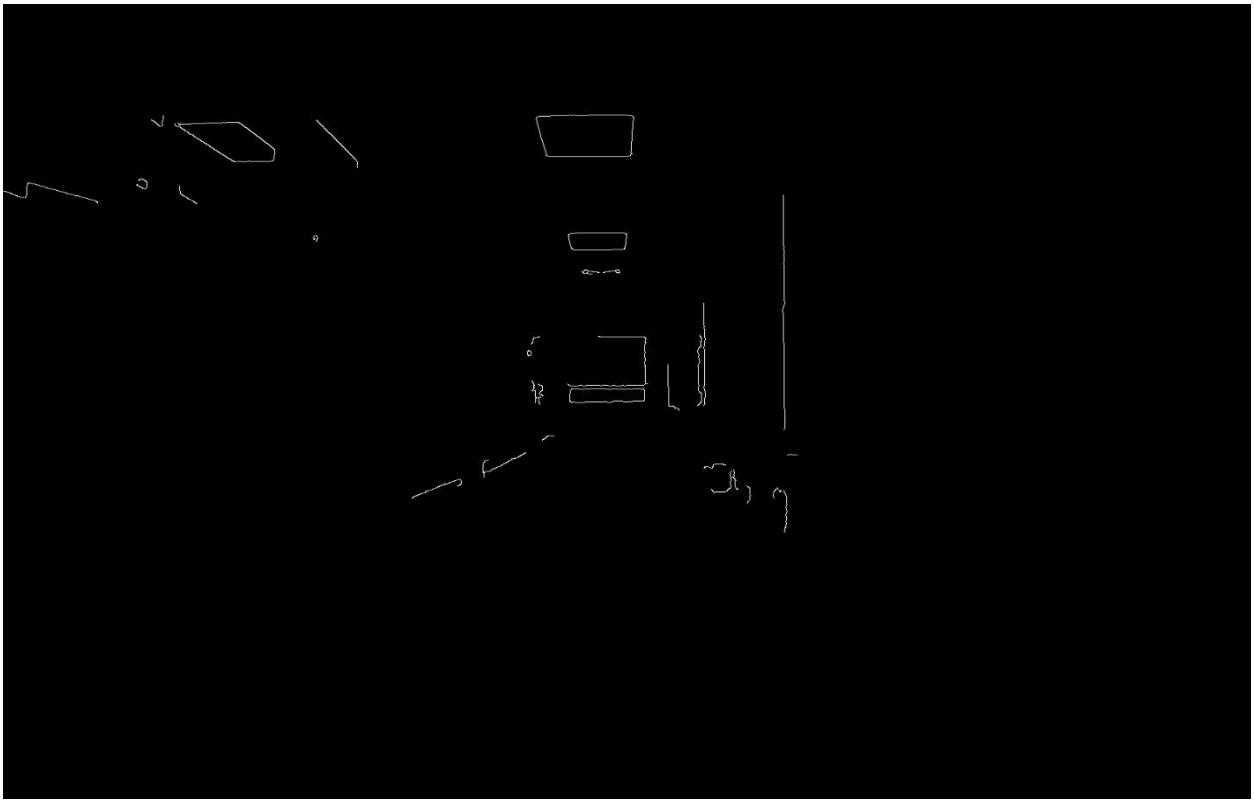
BCanny4001.jpg



RCanny4050.jpg

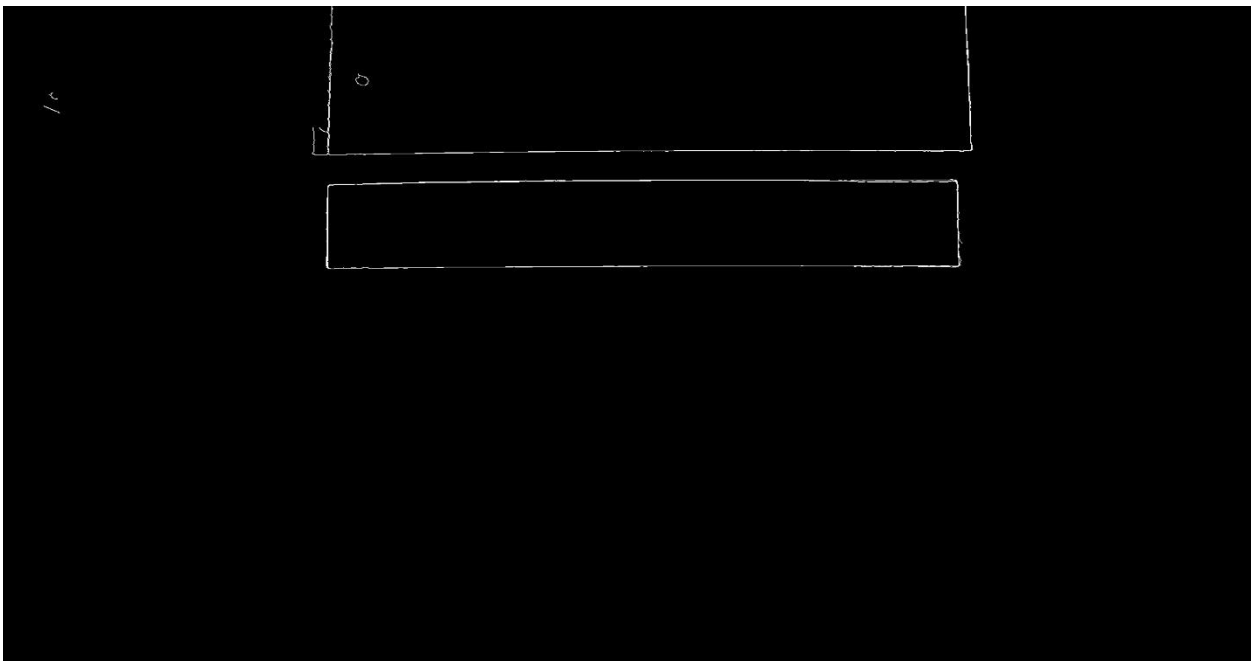


GCanny4050.jpg

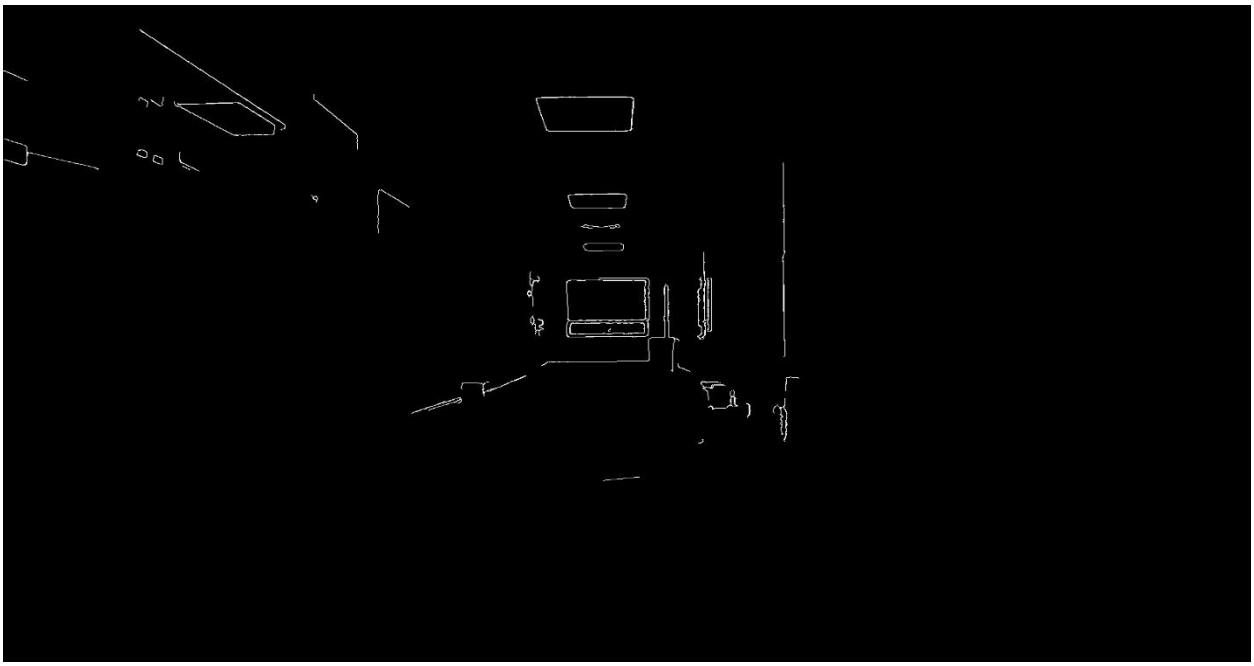


BCanny4050.jpg

Add all the RGB canny edges to get the final mask for building the histograms.

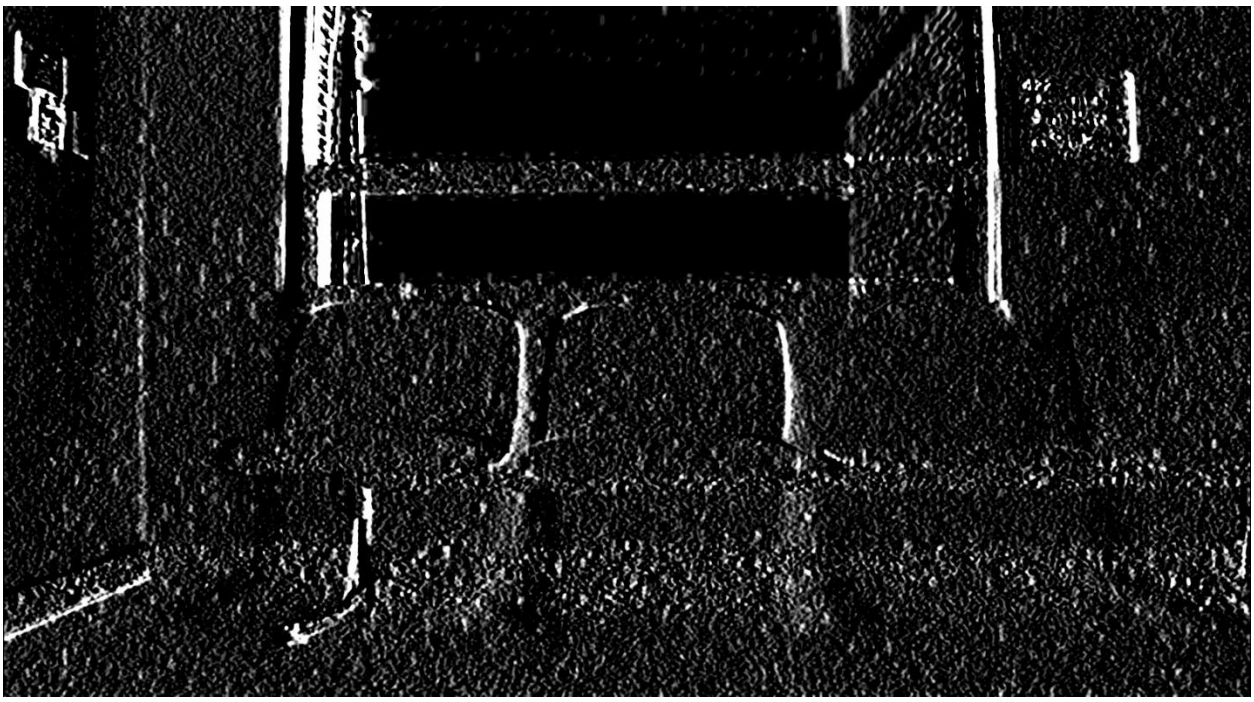


Canny4001.jpg

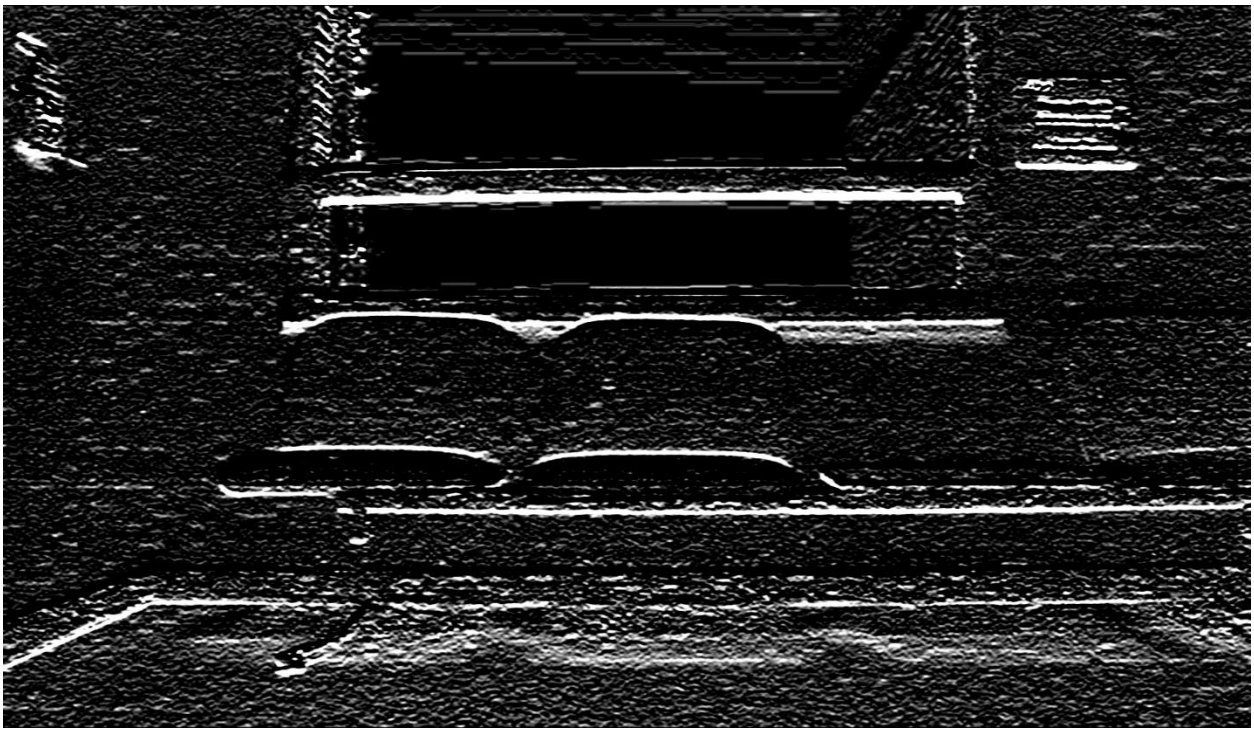


Canny4050.jpg

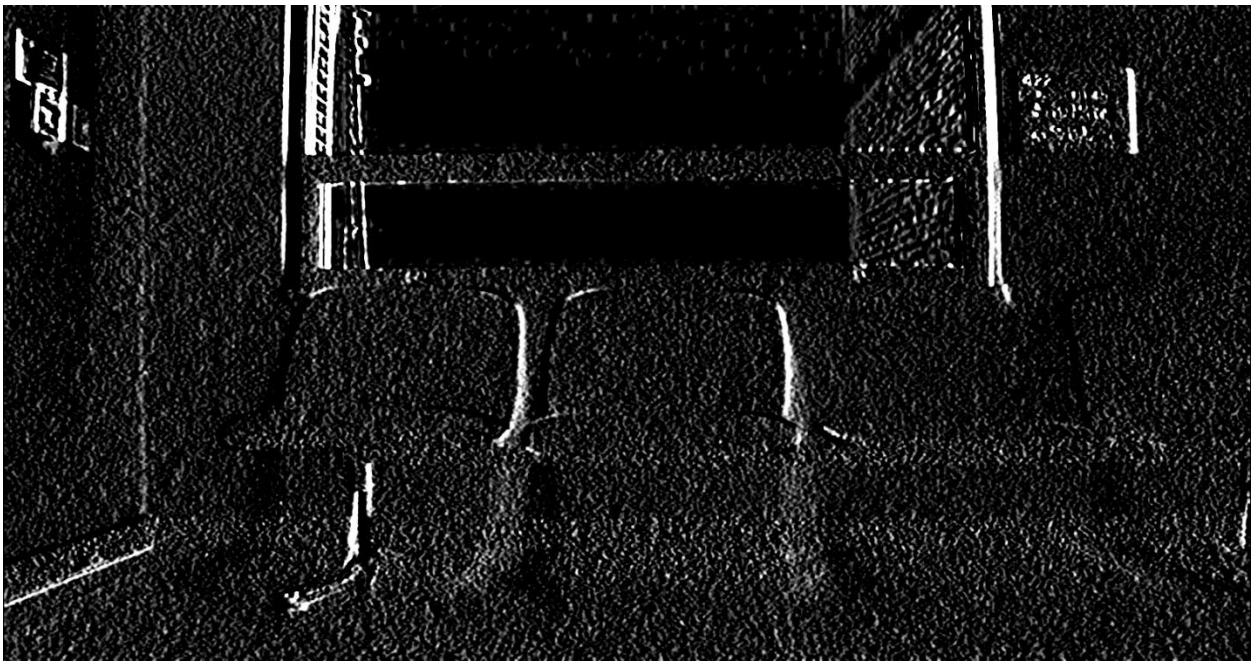
I then find the gradients for all the R G and B images.



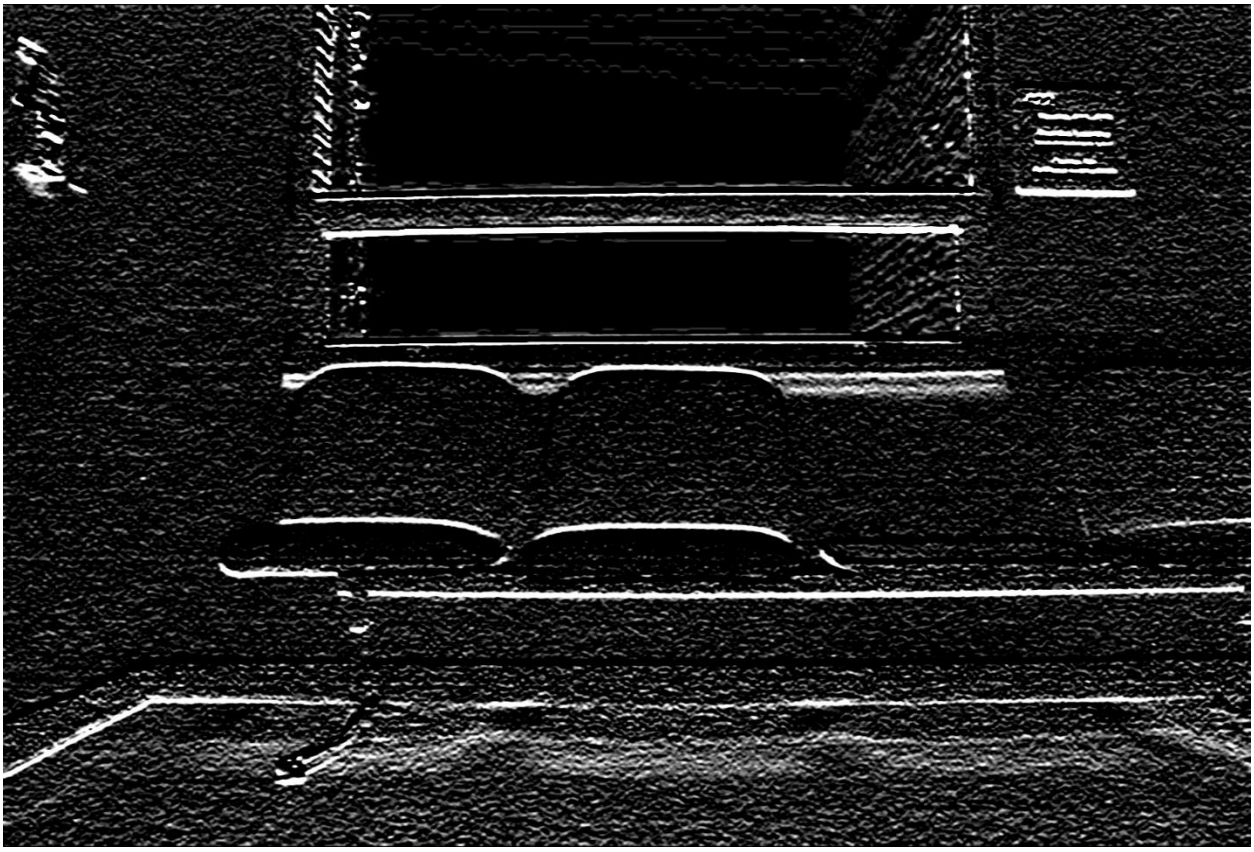
Rx4001.jpg



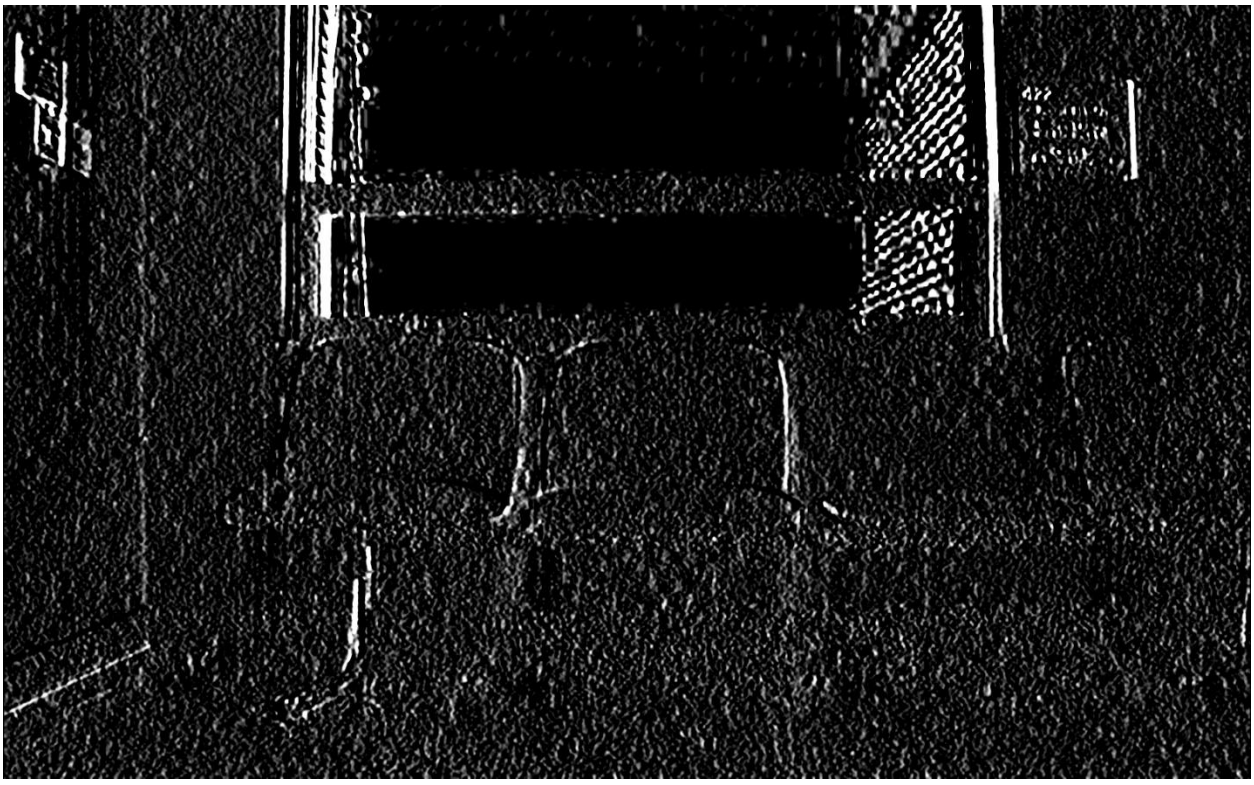
Ry4001.jpg



Gx4001.jpg



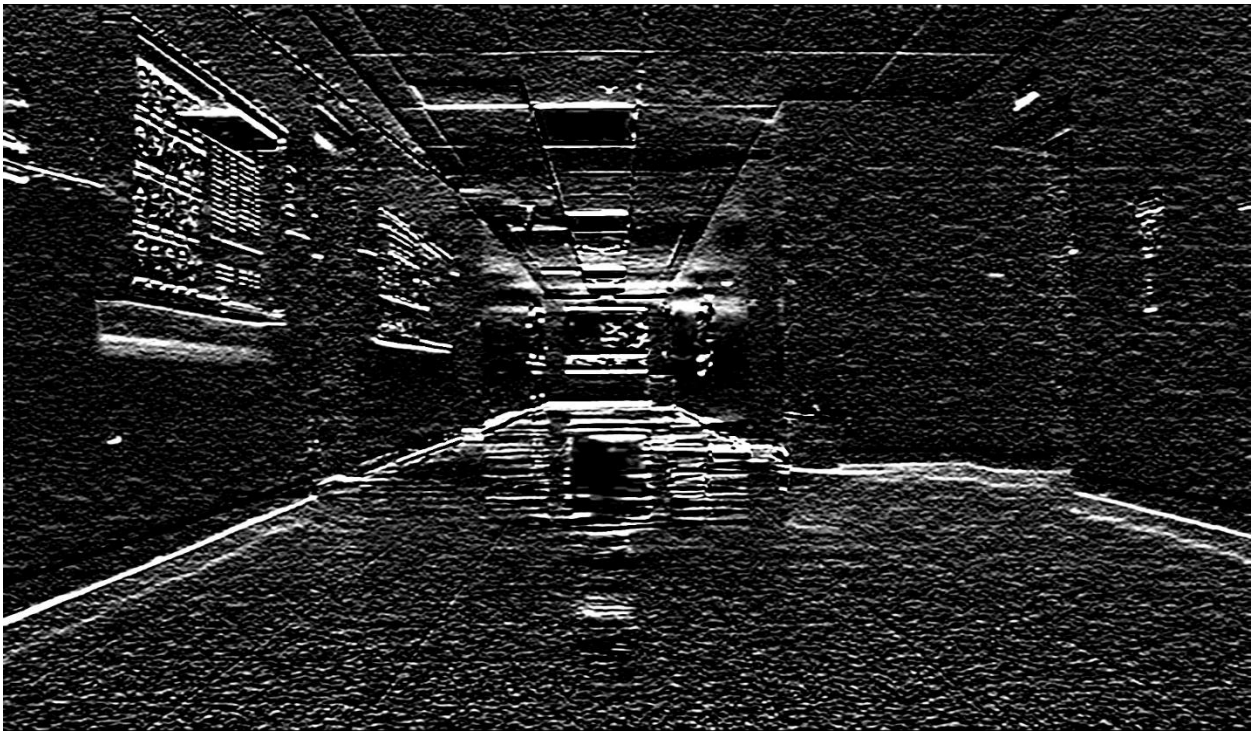
Gy4001.jpg



Bx4001.jpg

By4001.jpg

Rx4050.jpg

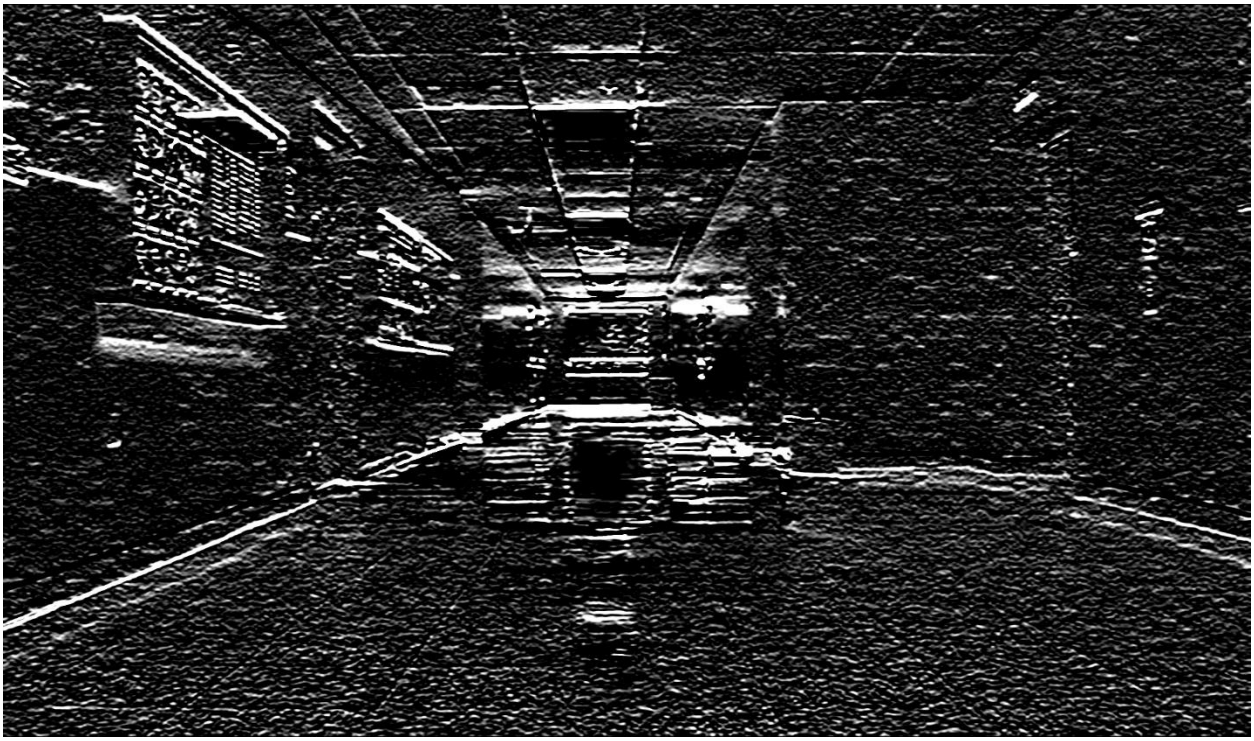


Ry4050.jpg

Gx4050.jpg

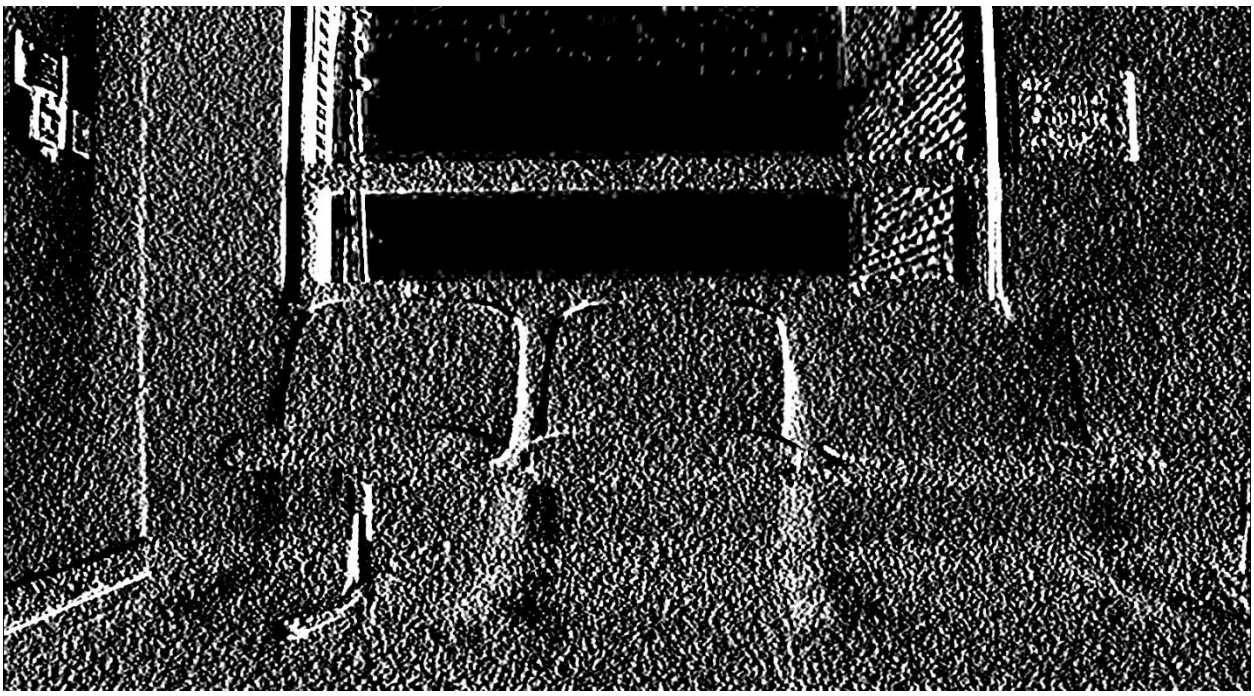
Gy4050.jpg

Bx4050.jpg



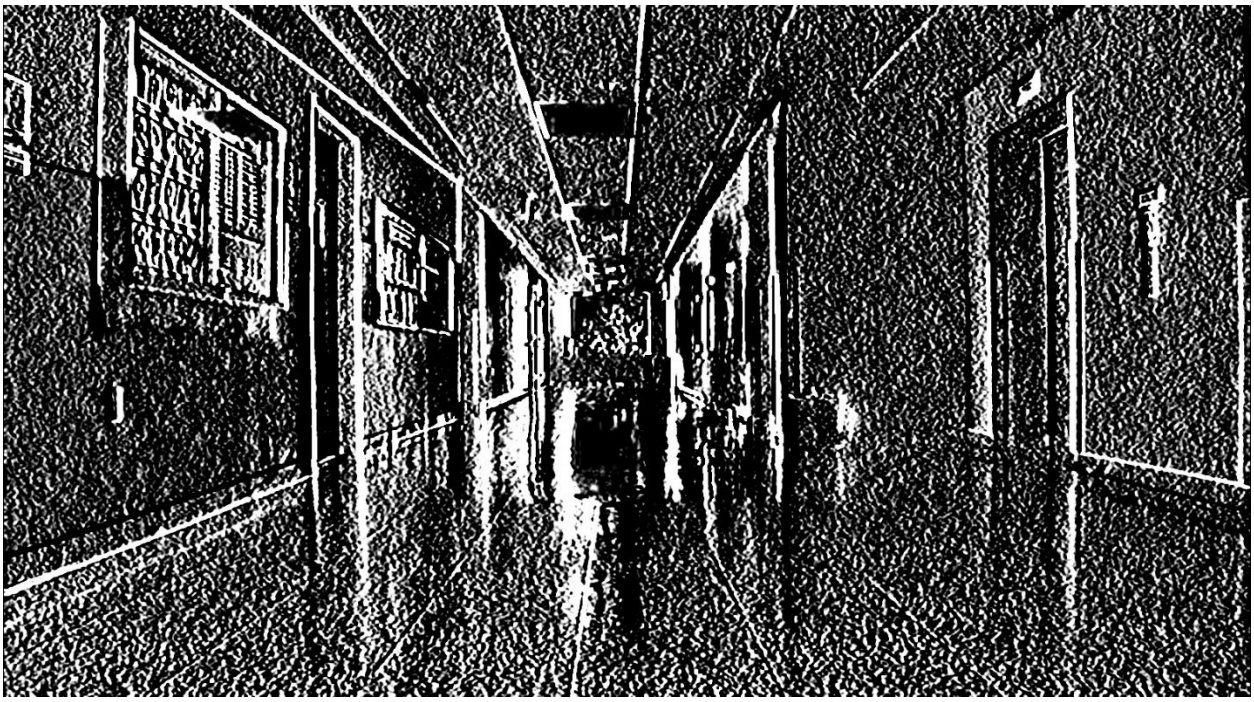
By4050.jpg

Finally create the U and V vectors, U = Rx+Gx+Bx and V = Ry+Gy+By

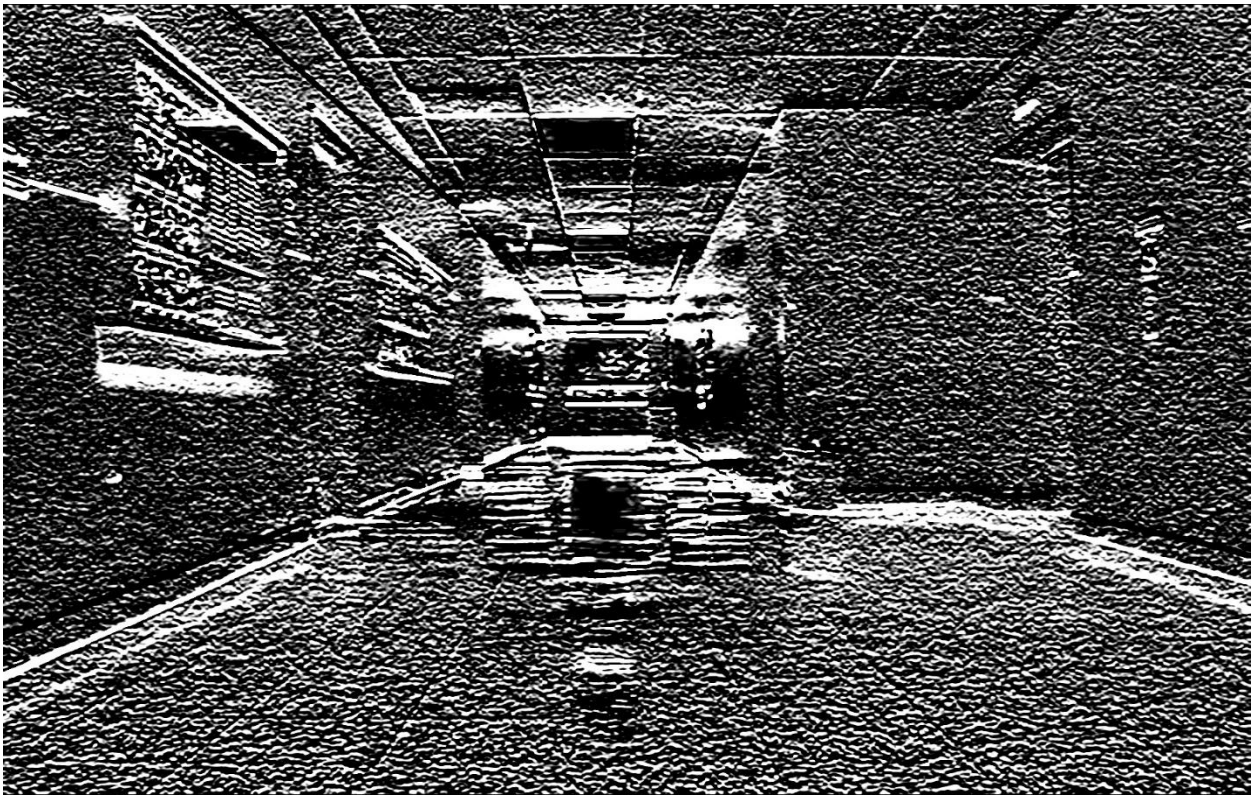


U4001.jpg

V4001.jpg

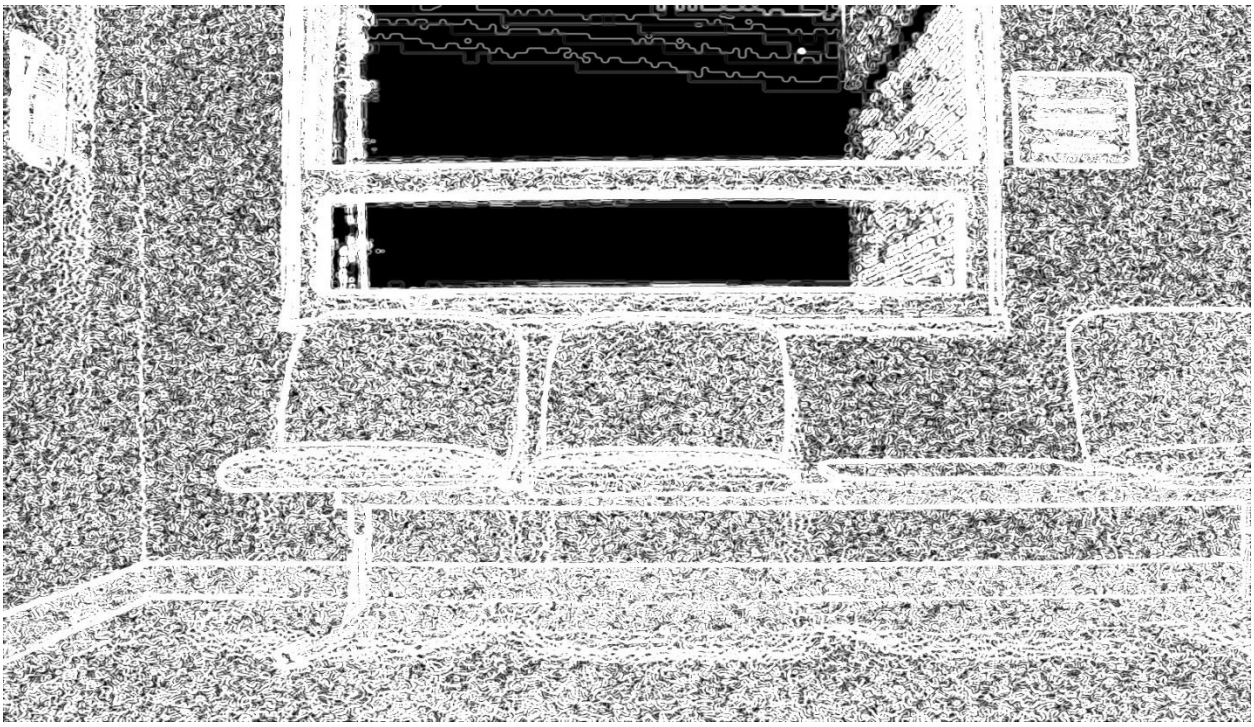


U4050.jpg

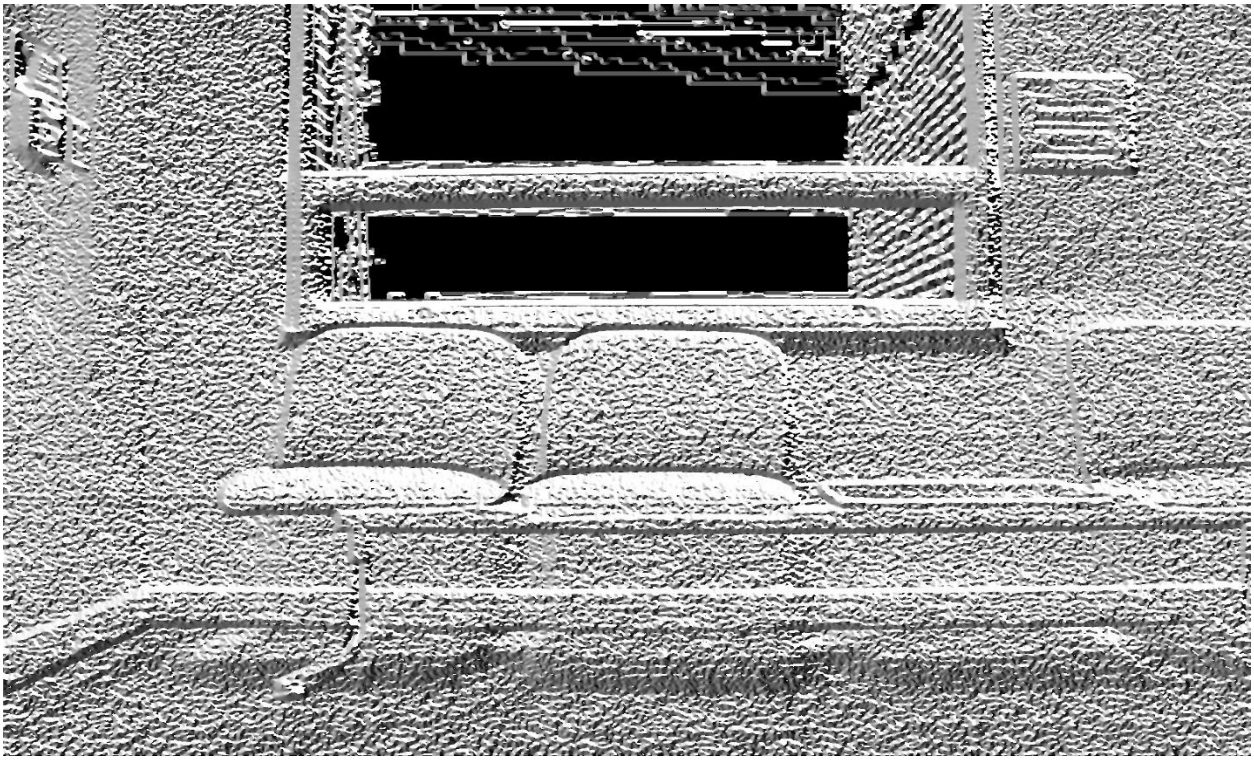


V4050.jpg

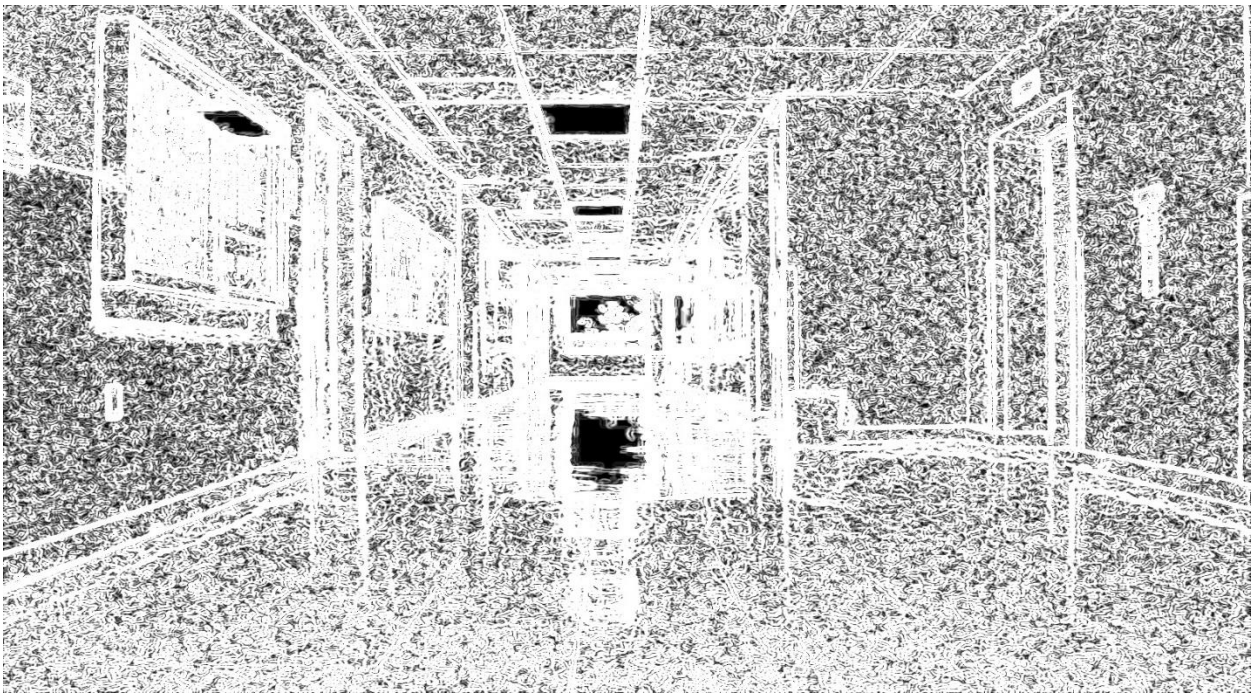
Using the cartToPolar built in function from OpenCV, we calculate the magnitude and phase of the U and V vectors.



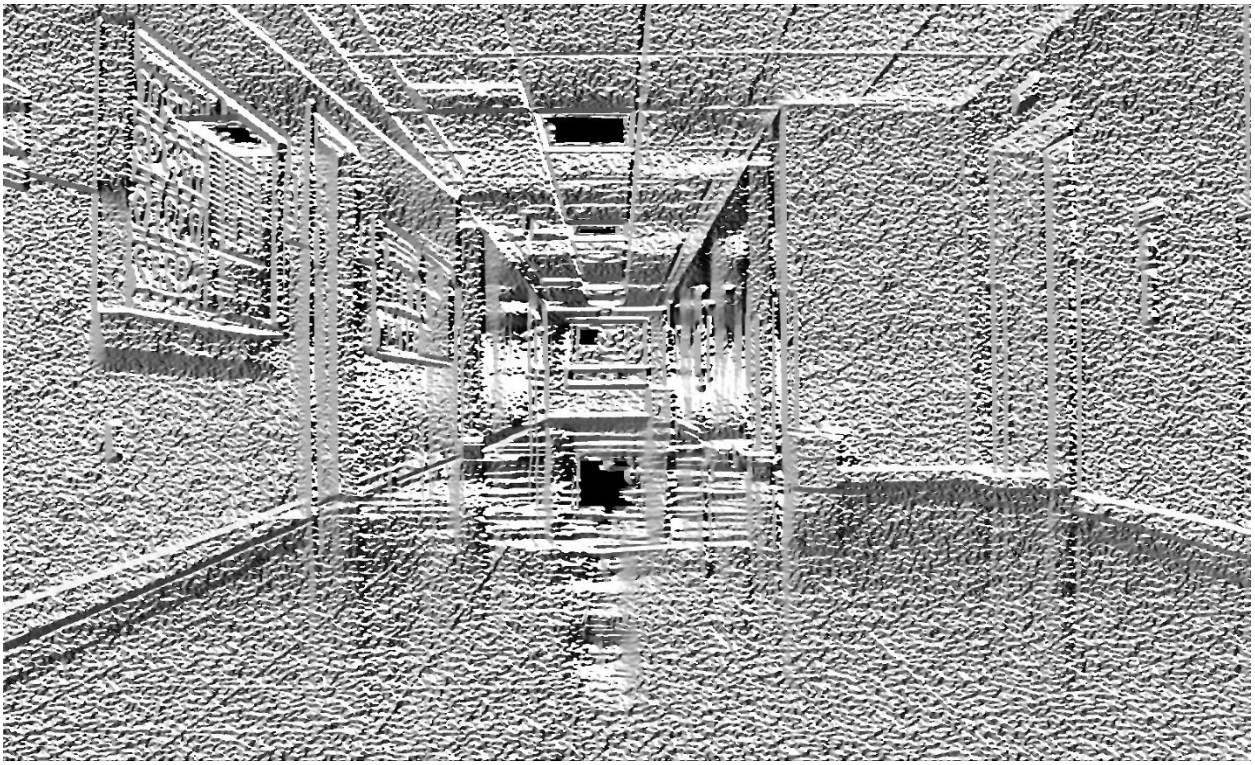
Magnitude4001.jpg



Phase4001.jpg



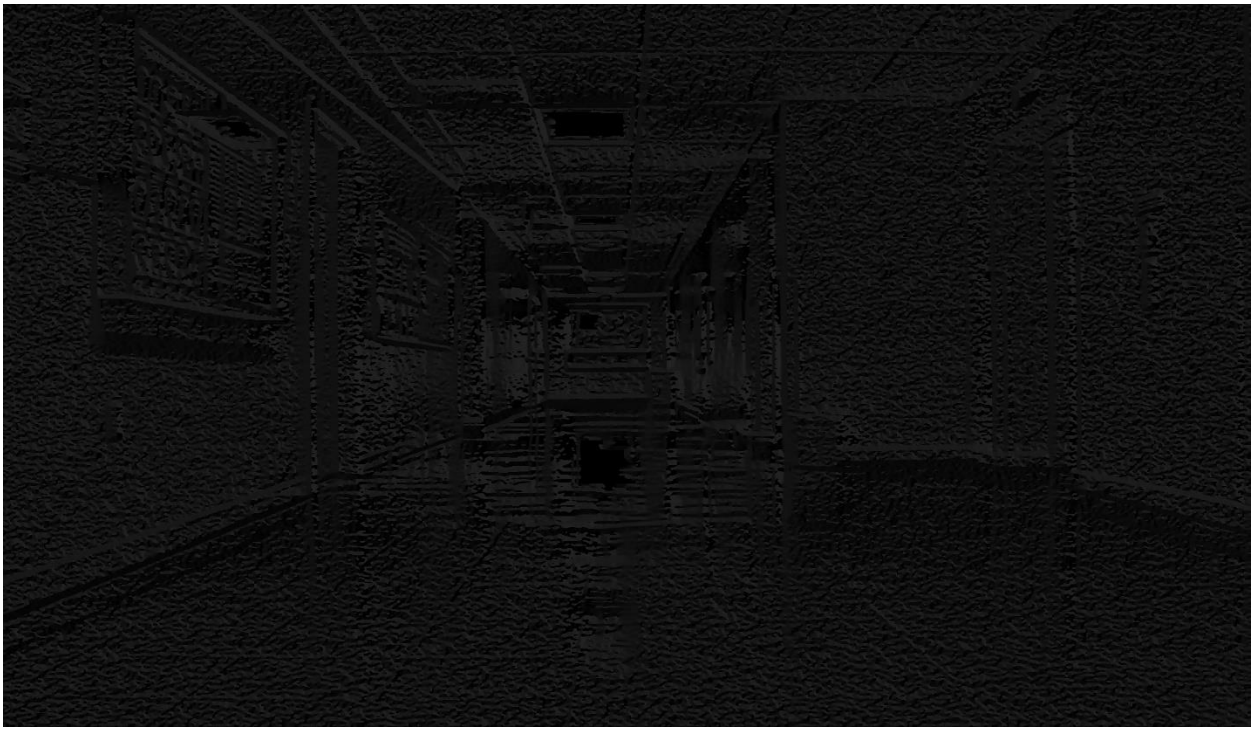
Magnitude4050.jpg



Phase4050.jpg

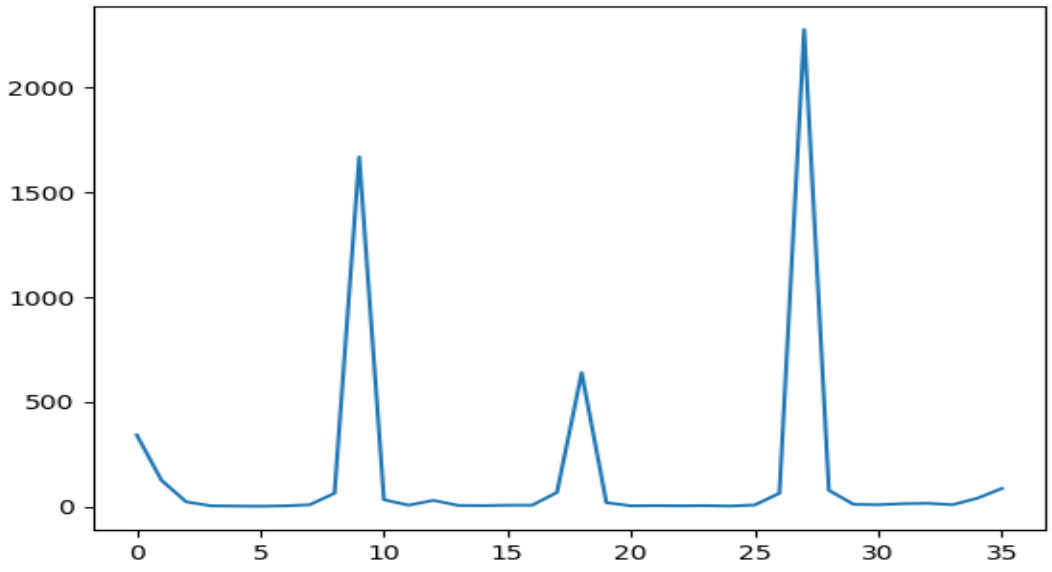
Also, we divide and round the phase to make it to fall in 36-bin range.

Angle4001.jpg

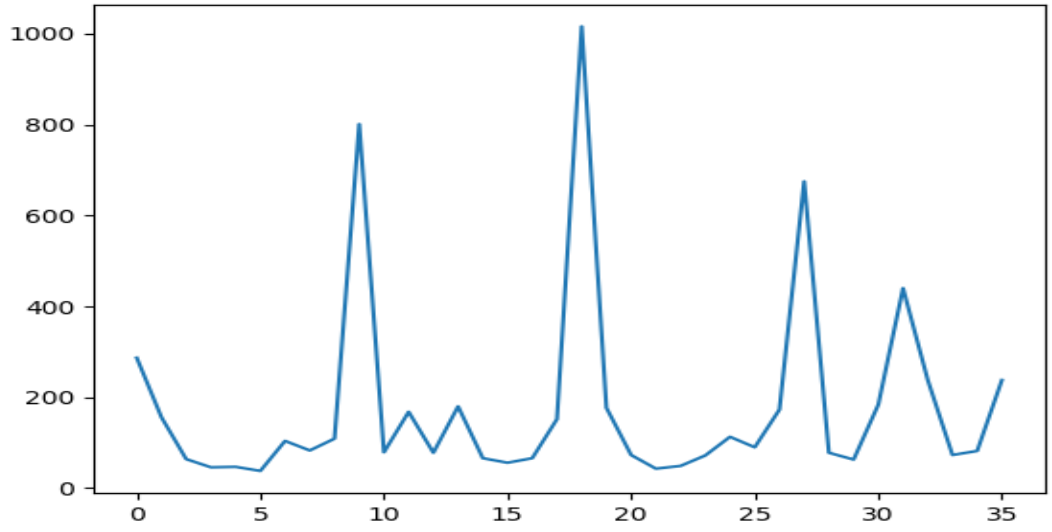


Angle4050.jpg

Later we use the canny edge detection image as a mask and plot histograms for the angles in the images. This masking helps to eliminate all the unwanted edges into consideration.



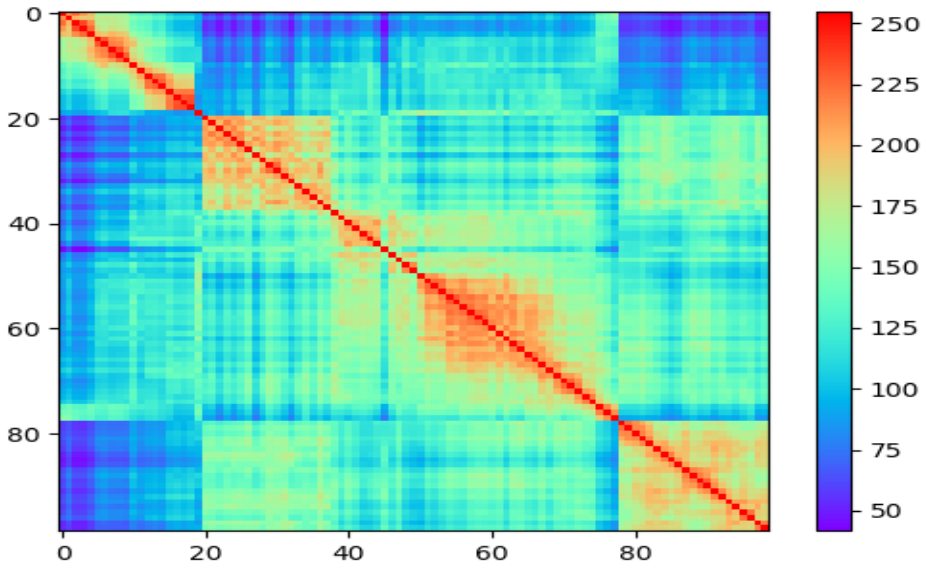
ColorHist4001.jpg



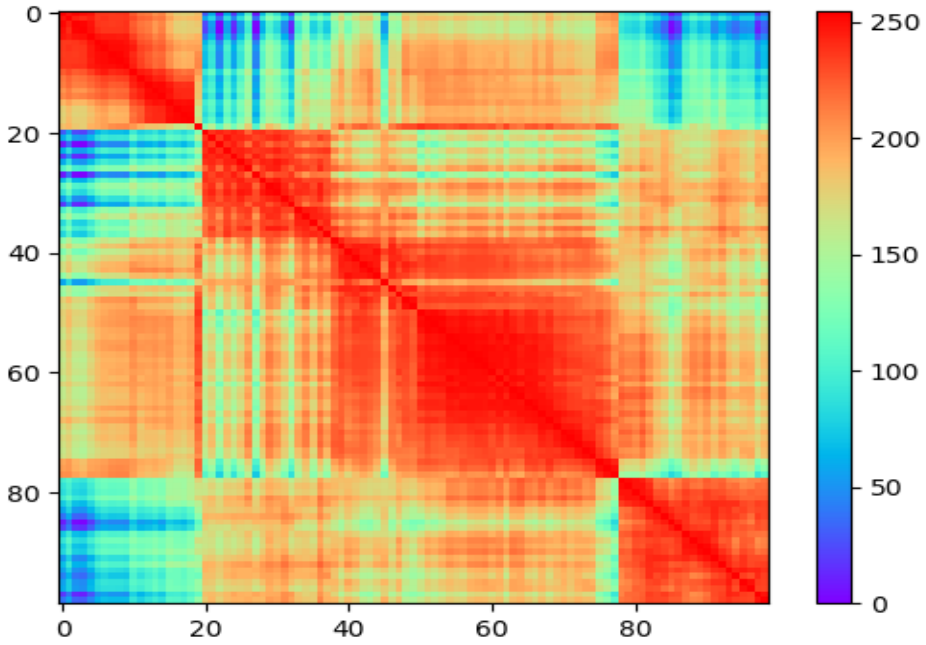
ColorHist4050.jpg

I have then used the histogram intersection and chi-square comparison to compare all the gray and color edge histograms. Unlike the previous homework I have normalized the Chi-Square function.

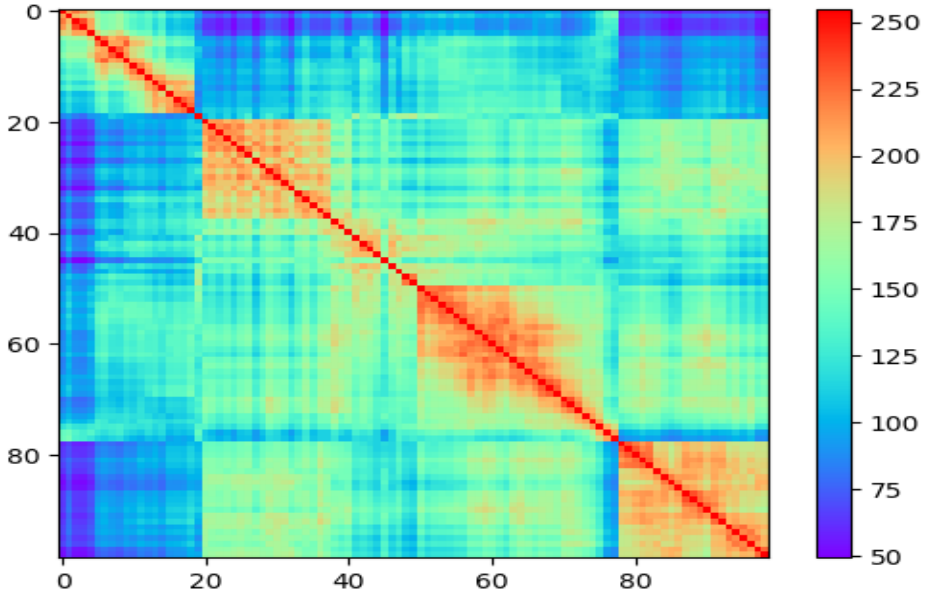
The results are given below and the histogram comparisons for the images look quite similar.



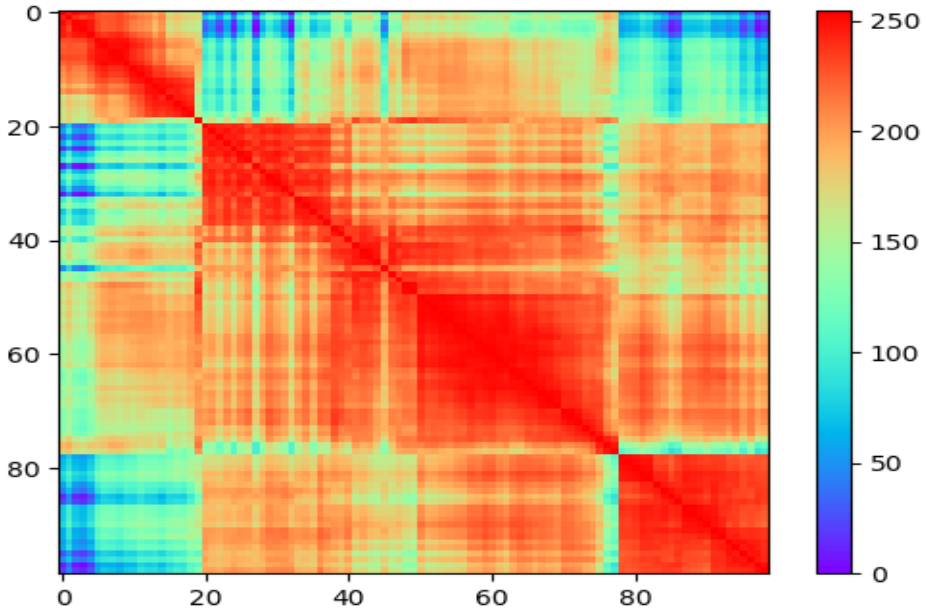
Gray Intersection



Gray Chi Square



Color Intersection



Color Chi Square