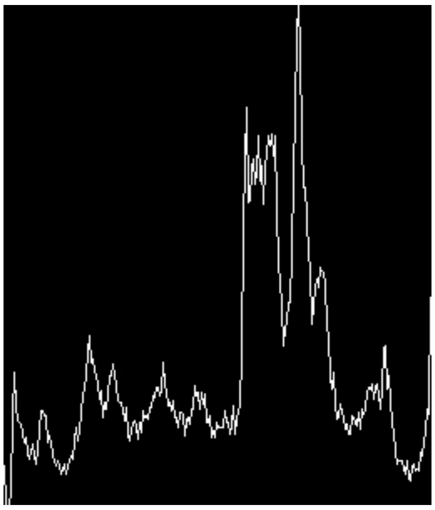
DIP_HW2

R05922130 王瀚磊

The environment of development is OSX with c++ and opencv.

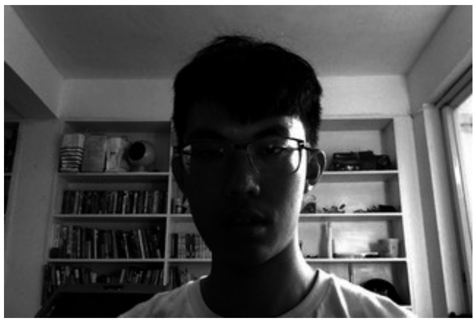
1-1 Draw Histogram

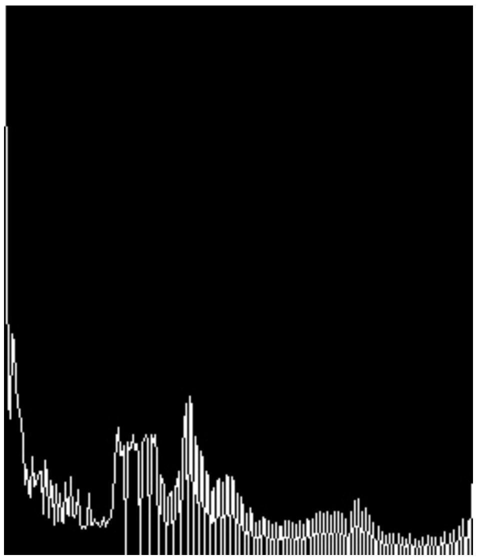
To implement the function which can draw a histogram of image, I use the function in opencv named cv::calcHIst(), which can output the histogram matrix. Then I use a for loop to draw the output of the cv::calcHist() function with another function in opencv named line().



2-1 Gamma correction with (c=1, r=2.5)

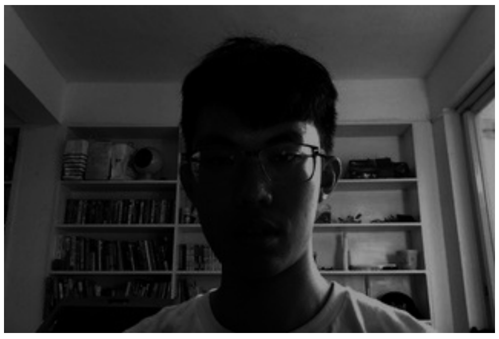
I use the two for loops to read the source image which is gray-scale. First, normalize all pixels' value between 0 and 1, which is called intensity. Then I calculate it in the power of 2.5, and un-normalize the pixel value that multiple 255.

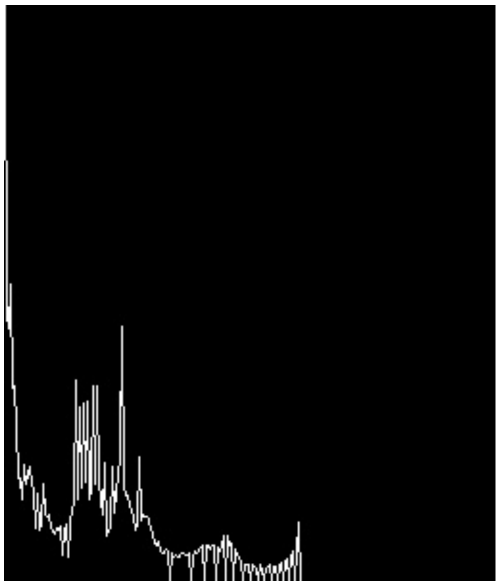




2-2 Reduce dynamic range of 2-1

Same as 2-1, but in the last step(un-normalize) I just multiple 60% of 255. Then I can get the dynamic range which is reduced to 60%.

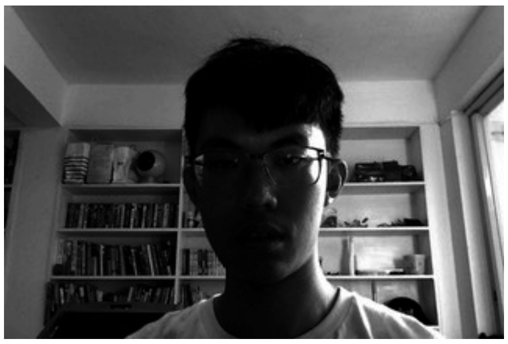


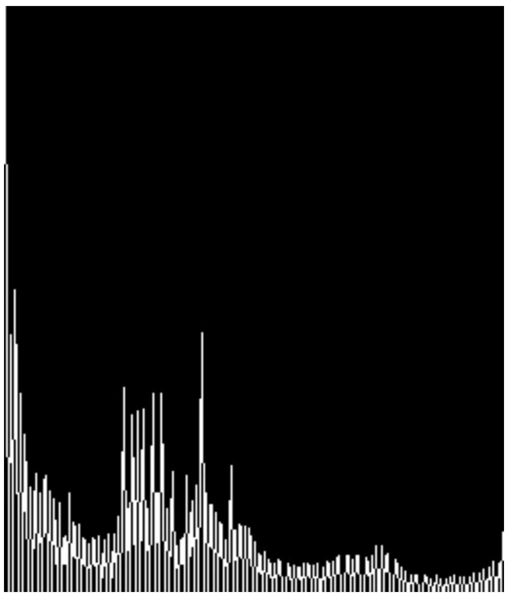


3-1 Full-scale image

Generally we have to find to points (r1,s1) and (r2,s2) which can help use find the curve of the pixel value, then multiple the pixel value with the scope of the slope.

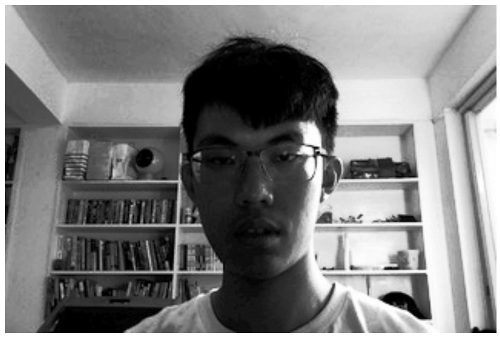
But in this case, the two points of the 2-2 image output are (0,0) and (r2,255). So I just need to find out the value of r2 and multiple all pixels with 255/r2, which means to scale the largest value to 255.

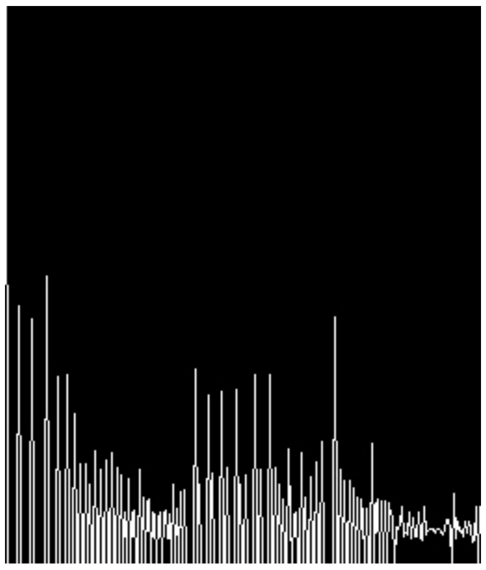




4-1 Histogram equalization

This is the easiest requirement. I just use the function in opencv named cv::equalizeHist.





5-1 Question I

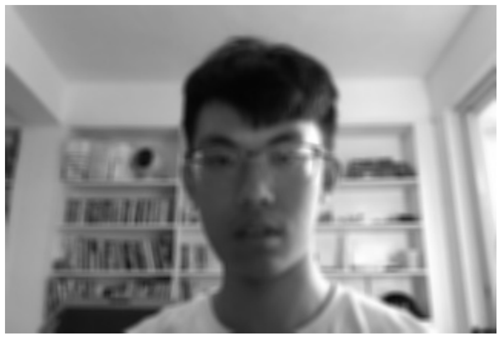
Because during histogram equalization, in effect, we use a discrete sum to approximate a continuous integral of the cumulative PDF of the pixels. In other words, pixel values are discreet integers and the number of pixels is also an integer, which affects the final outcome.

5-2 Question II

We can find out that it looks brighter in the result of 4-1. Because Histogram Equalization can make distribution more uniform than full-scale stretch though it's not absolute uniform. In full-scale stretch, it can just processing each same pixel value , it cannot separate each same pixel value to its' neighborhood.

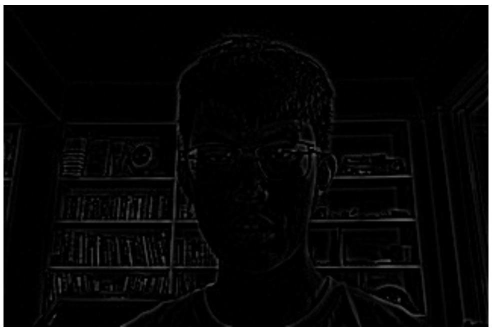
6-1 Smooth selfie

This is also easy. I just use the function in opencv named cv::blur().



6-2 Unsharp mask

Just subtract the source image and the output of cv::blur() which mentioned above.



6-3 Add the mask to the original image

Just add the output of 6-2 image in k-scale to the source image. In the image below, I set the k-value as 3 because that I think its’ contour information is enough. If I set k-value larger than 3, its’ contour information will make the image seems like noise.