ANKARA UNIVERSITY

Computer Engineering

COM 466/4504

Digital Image Processing

Practical Coursework 2

Zehra

In this work we are going to use the golbasiCampus-Degraded.jpg image that was obtained with reducing contrast and addition of noise. We will apply 4 tasks and try to improve the image in different aspects.

Helper Functions

Helper function means a function that performs part of the computation of another function. We wrote various functions to read and write images and express the results more clearly.

read: It reads an image from its path.

read_convertBGRtoRGB: It reads an image from its path, then converts it to RGB image.

read_convertgray: It reads an image from its path, then converts it to grayscale.

write: It takes a RGB image then converts it to a BGR image. As a final step, it saves the result in the new_path location.

show_img_cv: You can show the image result thanks to opencv.

show_img: You can show the image result thanks to matplotlib. It shows the result in size of 10×20.

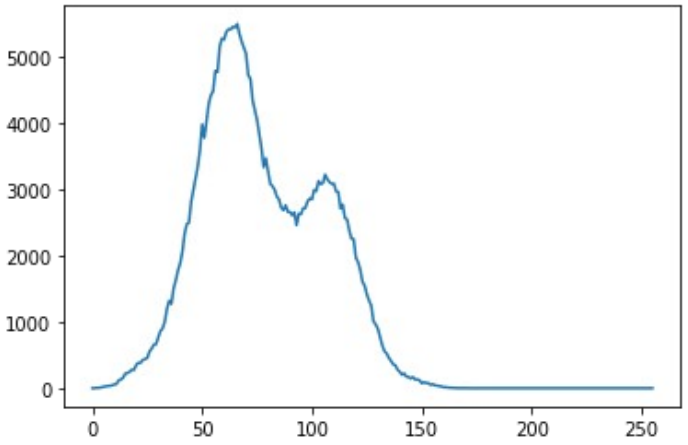
visu: It takes the old image and the new image that has been processed. It shows these two images side by side.

grayscale_histogram: It takes the path of the image then, converts it to grayscale and calculates the histogram. Finally, it plots the image histogram.

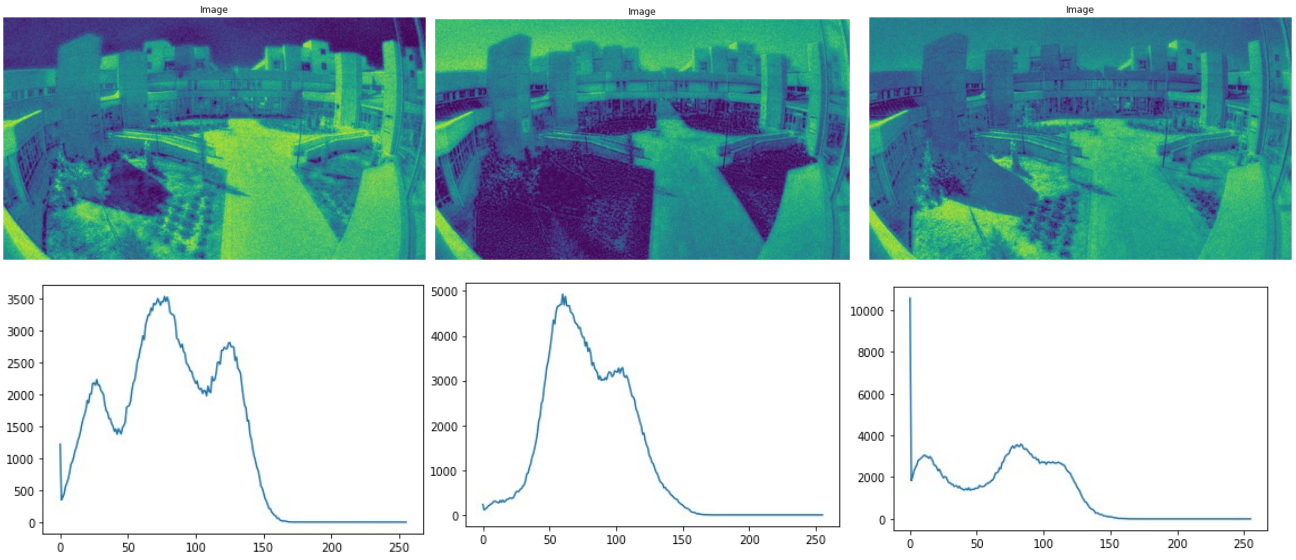
color_histogram: It takes the path of the image then calculates three separate histograms one each for the R, G and B channels. Finally, it plots histogram results.

Analysis

When we are looking at our image, we can say the image has salt and pepper noise. Model of noise detection will be our next step. We should look at the histogram of the image. Finally, we realized that our image has a Gaussian noise.



In some cases only one channel of image has the noise. So to understand this, We split the image R,G,B channel then,look at the results and their histograms. All 3 channels have salt and pepper noise.



Task 1

Task description: Work on the degraded image and apply image restoration (noise removal) techniques to remove noise.

Method: We applied two different methods.The first method is the non-local means algorithm. The second method is the combination of the contra-harmonic mean with the median filter.

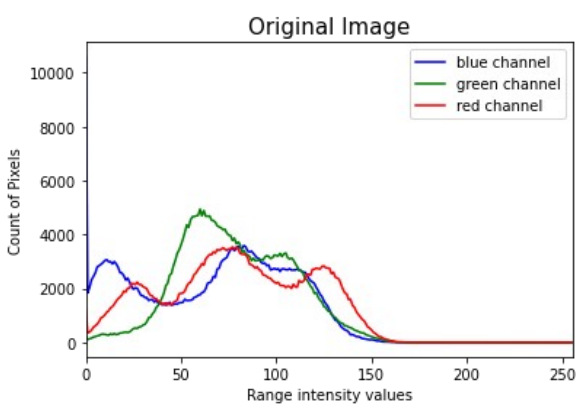
1. Method: Non-local means algorithm

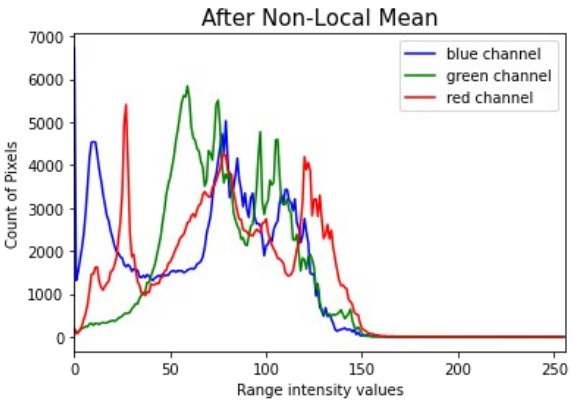
The Non-Local Means algorithm is used for refinement of image and preserving detail of image while removing the noise.

It restores the original image much better than standard median-based filter and it is more effective at reducing the salt and pepper noise.

When setting h and hColor, we tried 3. The noise in the photo was not removed to the desired extent. Also, when we set it to 10, we got a very blurry image. As a result, we decided to set the h and hColor values to 5.

When deciding the value for templateWindowSize and searchWindowSize, we used the recommended values(7 pixels and 21 pixels).





Result of First Method: Salt and pepper noise has been removed and the details of the image were preserved.

2. Method: The combination of the Contra-harmonic Mean with the Gaussian Filter

Contra-Harmonic Mean filter is used to remove Gaussian type noise and preserve edge features. Based on this expression we write CHM_operator() function.

$$\hat{f}(x,y)=\frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

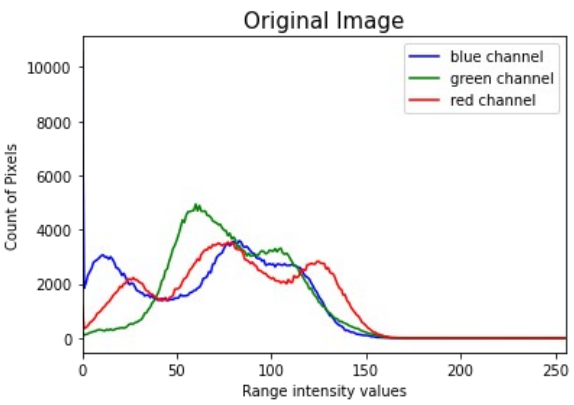
We apply Contra harmonic mean algorithm to all 3 channels then, we combined them.

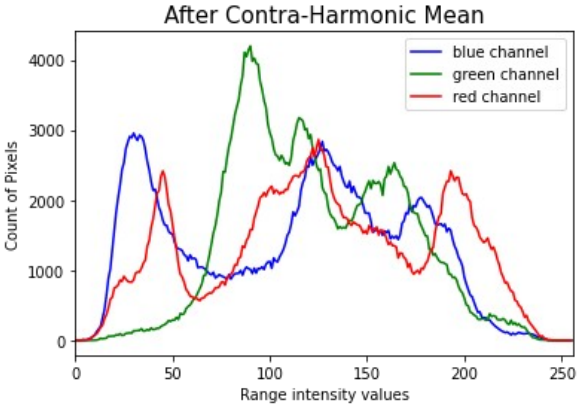
We used Contraharmonic Mean for pepper and salt noise. Positive values of Q eliminate pepper noise(Q taken as a 3).

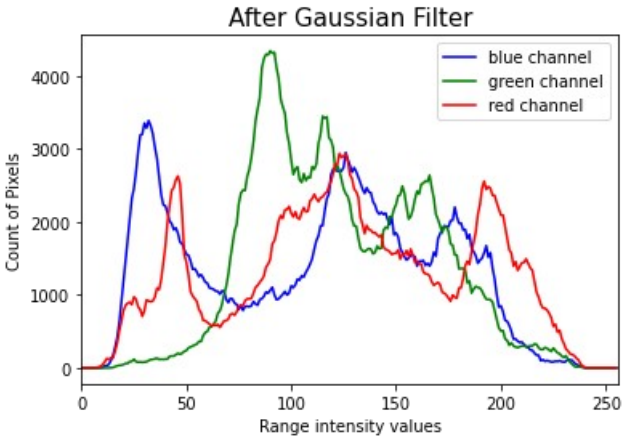
Then we aimed to remove the salt noise by taking the value of q negative(Q taken as a -1.5).

Then we apply the gaussian filter(kernel size taken as a 5)

Let's look at the results of the operations







Result of Second Method: Salt and pepper noise and gaussian noise has been removed but image details have been lost.

When we compare the two methods, we can say that the first method is better. Since we are only allowed to use the methods discussed in the course, we will continue with the result of the second method.

Task 2

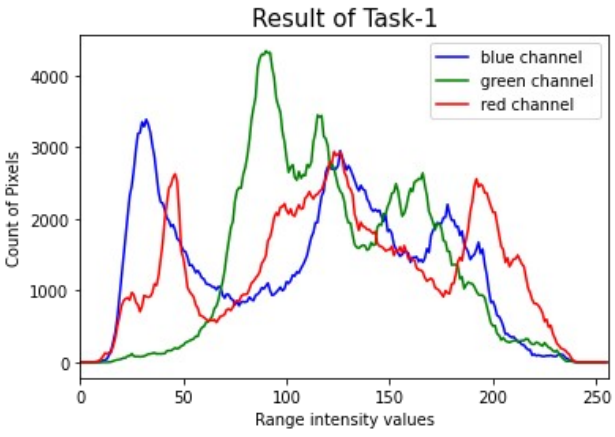
Task description: Enhance the output from Task 1 in terms of contrast.

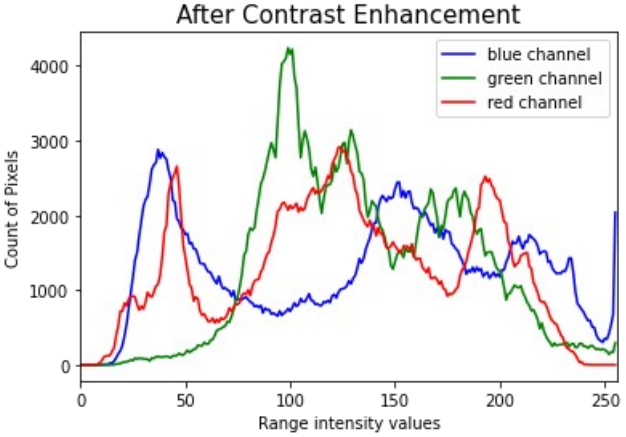
With applying this formula $g(i,j)=\alpha \cdot f(i,j)+\beta$ to each pixel ,we can change the contrast and brightness. We can control contrast with alpha value. In the OpenCV module, there is no particular function to adjust image contrast but we can use the addWeighted() method.

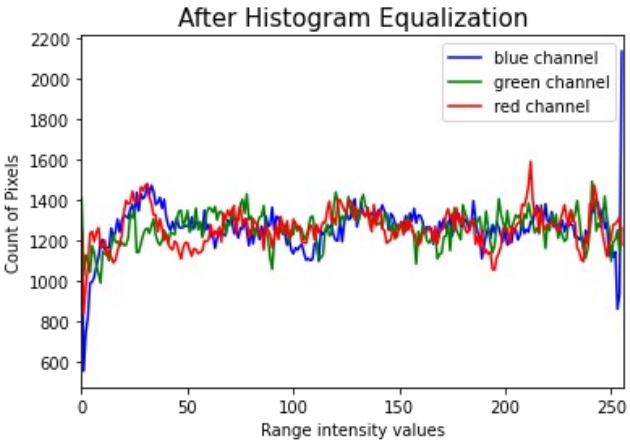
After dividing our image into R,G,B channels with the split function, we made contrast enhancements to all of them separately. As a result of our experiments, we decided not to change the contrast of the red channel too much. If we do this, the picture becomes dominant in red.

We adjusted our contrast values as R = 1.01, G = 1.1, B = 1.2. We don't want to change the brightness value so we set the beta value to zero.

Then we did histogram equalization for 3 channels of the image separately and then combined them. As a result, the colors of the image look very good.





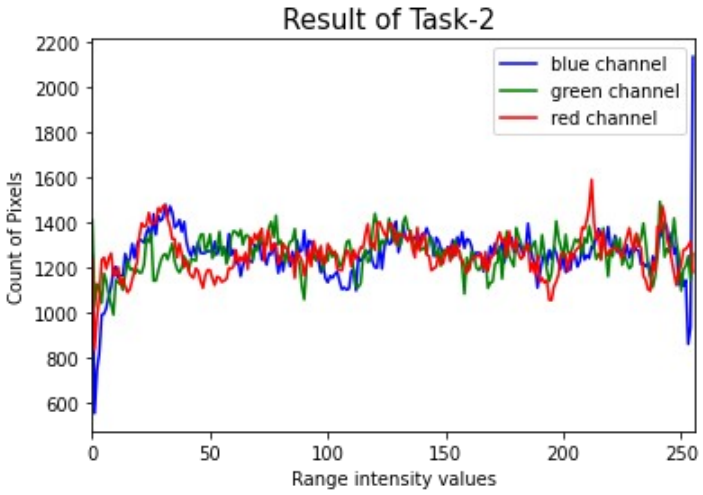


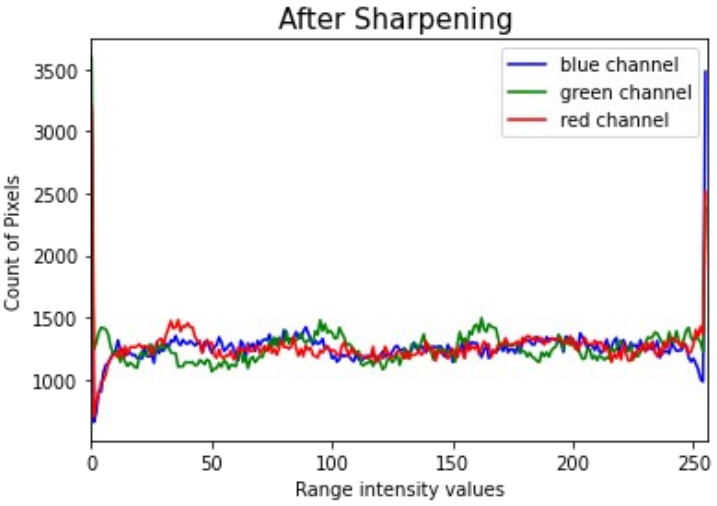
We spreaded out the most frequent pixel intensity values . By accomplishing this, histogram equalization allows us the image’s areas with lower contrast to gain a higher contrast.

Task 3

Task description: Analyze the image and find out if you can enhance this image for visual appeal.

The color balance of our image looks good. But we have a blurry image. We will sharpen the image to solve this problem. We used [[0, -1, 0], [-1, 5, -1], [0, -1, 0]] filter to do this operation.



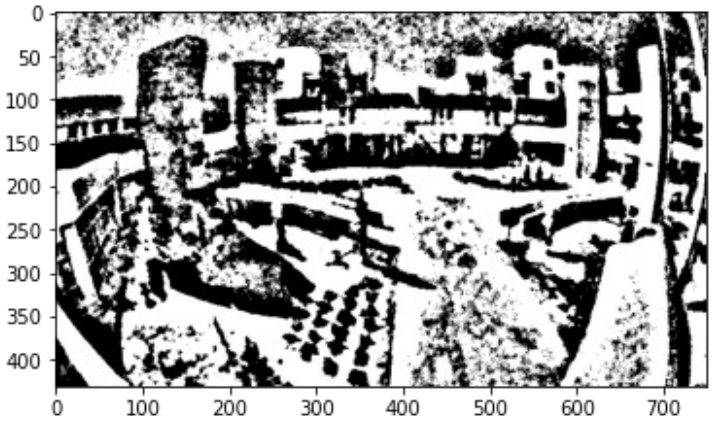


Task 4

Task description: Convert the image to grayscale. Then, apply thresholding to produce a binary image.Then, apply dilation, erosion, closing, and opening morphological operations on the binary image.

In simple thresholding, the threshold value is global and it is same for all pixels in the image. When we applied the simple threshold with average intensity as the threshold value, the image became very fuzzy. So we decided to apply an adaptive threshold. In adaptive thresholding, the threshold value is calculated for smaller regions so there will be different threshold values for different regions. We used adaptiveThreshold() method to apply it on our image.

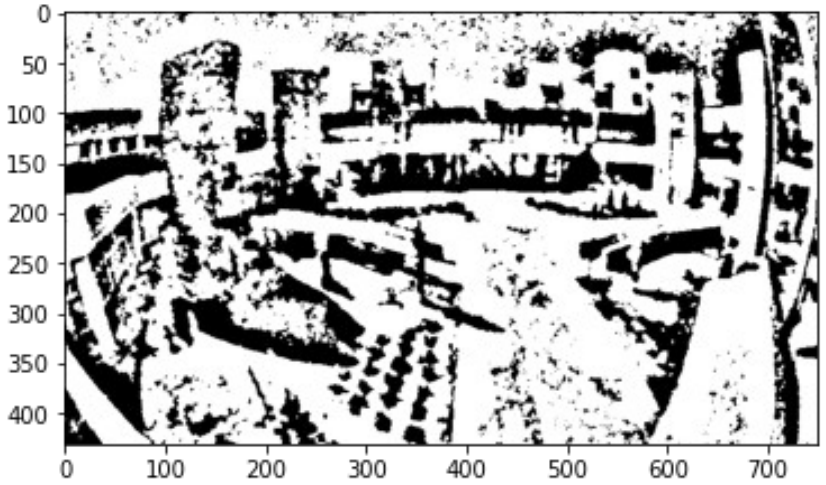
If the pixel value is more than the threshold value, let's change it as a 255. And our adaptive method is ADAPTIVE_THRESH_MEAN_C. Threshold value will be the mean of neighborhood area. The blocksize is 51. It is the size of the pixel neighborhood used to calculate the threshold value. The reason we made this choice is to see the differences between morphological processes more clearly.



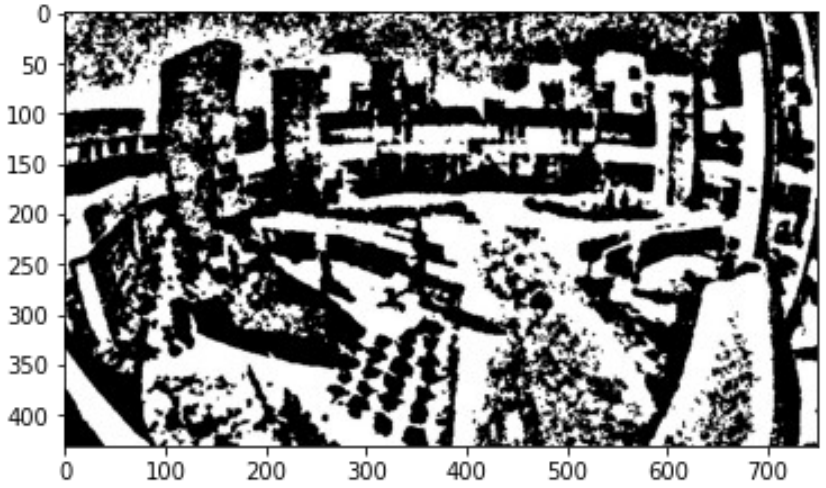
Result of the adaptive threshold

We chose a 2 × 2 kernel size for the morphological operations. We will use the same size for all.

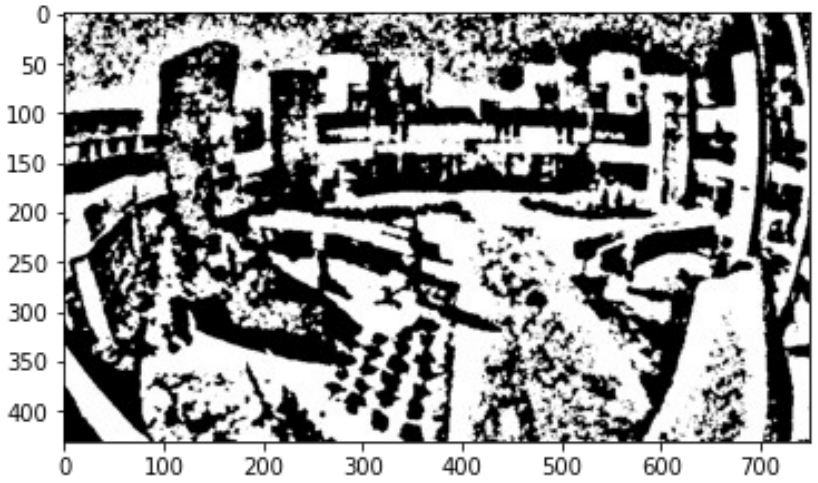
Dilation



Erosion



Opening



Closing