EE 569: Digital Image Processing

Homework #1

Problem 1: Basic Color Manipulation
This problem involves a series on simple manipulations on grayscale and color images to gain familiarity with image data access, processing and output.

a). Colour Space transformation

1. Colour-to-Grayscale conversion

Abstract
A colour space or colour model is an elaborate coordinate system or subspace which is a combination of primary colours written as tuples of 3 or 4 numbers which describe the range of possible colours in that model. A single dot represents a colour in the system.
The most commonly used colour model is the RGB system which consists of Red Blue and Green as the primary colours. This is an additive colour model as the colours Red, Blue and Green are added in various combinations to produce the broad range of colours used in the system. In an additive colour model, the colours are considered to be "pure" and all add up to produce white. In the RGB model each colour channel is represented by 8 bits, bringing the length of each pixel to 24 bits. Figure 1 is a simple representation of the additive property of the RGB model.

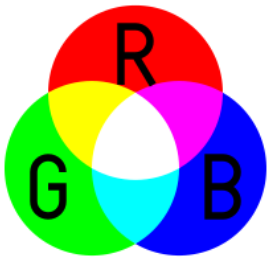


Figure 1 RGB Additive colour model

Grayscale is the model where each pixel is represented by 8 bits instead of the 24 bits as in the RGB model. While the RGB model can create 16,777,216 colour combinations for each pixel, the greyscale is basically 256 shades that lie between black and white which results in an 8-bit image.

Approach
C/C++ is the language of choice for most of this project. The given images for all of this project is in .raw format where the pixels are arranged in the recursive RGBRGBRGB… way. The method followed throughout this project is to first separate all the colours into three separate channels one each for R, G and B.
Greyscale can be achieved by many methods three of which have been used in this project:

- Average method: is the average of R,G and B values at every pixel.

$$Greyscale = \frac{R + G + B}{3}$$

- Lightness method: this averages the least and most prominent colours at each pixel.

$$Greyscale = \frac{\max(R, G, B) + \min(R, G, B)}{2}$$

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

- Luminosity method: while this is also an averaging method, it used weighted averaging to account for human perception. Humans have different sensitivity to the three primary colours and are more sensitive to Green and hence it is weighted heavier than the other colours. The formula for Luminosity method is:

$$Greyscale = 0.21\ R + 0.72\ G + 0.07\ B$$

Experimental Results

The images below are the results obtained for the three separate approaches.

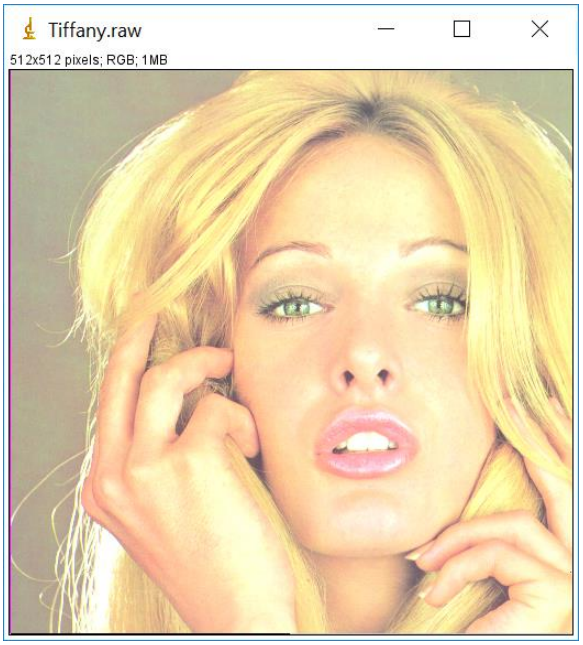


Figure 2 Tiffany.raw

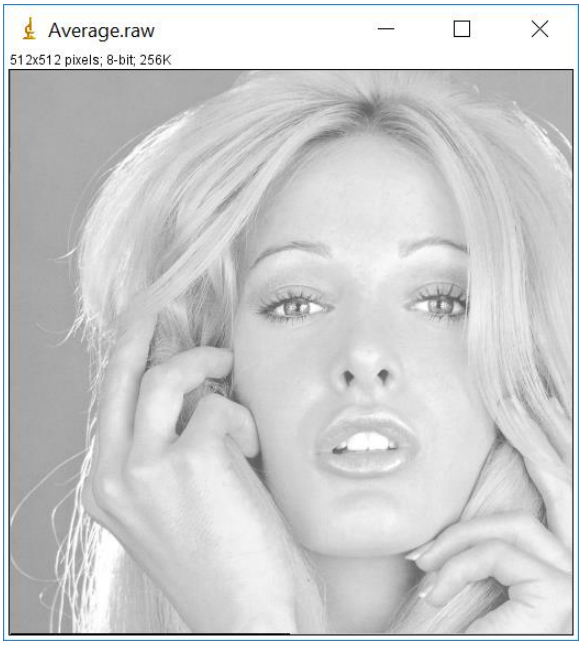


Figure 3 Greyscale from Averaging method

Figure 4 Tiffany Greyscale Lightness method

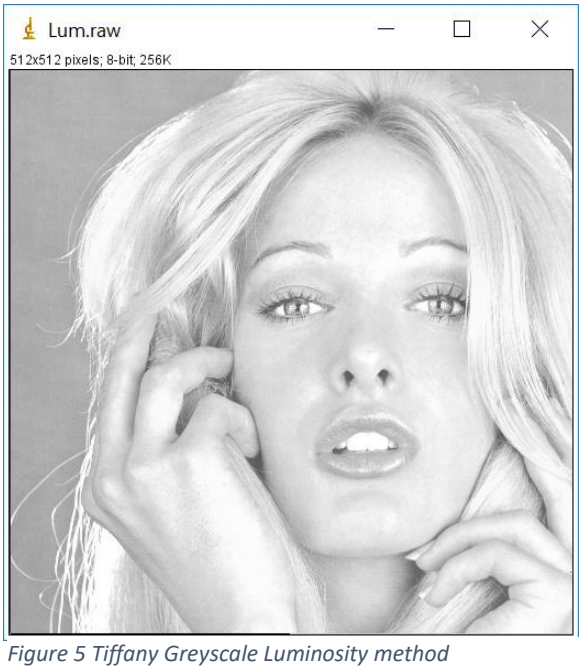


Figure 5 Tiffany Greyscale Luminosity method

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Discussion

Different method of converting colour to greyscale is used for different applications. The averaging method is a quick and simple method but here it produces a slightly washed out image compared to the other two. The grey shades produced are poor compared to the way humans perceive luminosity.

Looking at the images above we can see that the Lightness method has reduced contrast as compared to the other two but it has less loss during conversion than compared to the averaging method. The luminosity method works best when compared to the original image as it plays on the fact that the distribution of cones in the human eye means that they are more sensitive to green as compared to the other two primary colours. This is the default method used in most photo-editing software as it produces a more dynamic range of greys.

The weighting factors used vary according to the application of the resultant image. Over the years there have been many standards defined depending on the specifications of display devices popular at that time. [1] [2]

All three methods are used for different applications. But we can see that the luminosity method retains better contrast and is the most pleasing to the eye.

2. CMY(K) Colour Space

Abstract

Also considered in second part of this problem is the opposite of the RGB colour model- CMY(K) model. This is a subtractive colour model where Cyan, Magenta and Yellow are the perfect complements of Red, Blue and Green respectively. This colour model is commonly used in printing devices because of its subtractive nature; when white light is incident on it, it retains certain wavelengths of light and reflects the rest to give the effect of color. The colours that are reflected are the ones that are not absorbed i.e, the complementary ones to the ones absorbed. In the case of CMY, the reflected colours would be RGB.

The draw back to this system is that if a colour other than white light is incident on the surface of say the printed paper, the reflected colours no longer appear "true". In the CMY(K) system, K stands for black and is added to the system to preserve ink while printing as it would otherwise require a mixing of all the other colours and is wasteful.

Figure 6 CMY colour model

Approach

After separating the *.raw* file into 3 separate channels for R, G and B, the conversation formula shown below is used to calculate CMY from RGB. The R, G and B are normalized values because while the values of RGB lie between 0-255, CMY ranges from 0-1 and is calculated from;

$$\begin{cases} C = 1 - R \\ M = 1 - G \\ Y = 1 - B \end{cases} \quad \text{-------(1)}$$

Because we use normalised values, the formula is going to be;

$$C = \left(1 - \frac{R}{255}\right)$$

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

This means that the input and output arrays for the images would be of type *float.* This resulted in very grainy images as shown in the experimental results section. Therefore, to use type *unsigned char*;

$$C = \left(1 - \frac{R}{255}\right) * 255 = 255 - R$$

This was repeated for M and Y.
This entire process was repeated for both Dance.raw and Bear.raw and the results are shown below.

Experimental Results

Figure 7 RGB Dance.raw

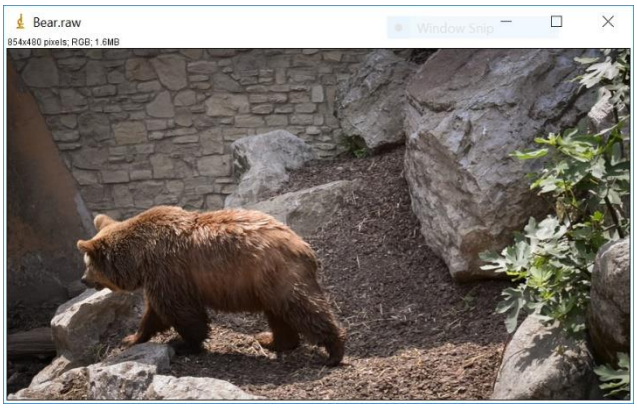


Figure 8 RGB Bear.raw

Figure 9 Cyan channel for Dance.raw

Figure 10 Cyan channel for Bear.raw

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Figure 11 Magenta channel for Dance.raw

Figure 12 Magenta channel for Bear.raw

Figure 11 Magenta channel for Dance.raw

Figure 12 Magenta channel for Bear.raw

Figure 13 Yellow channel for Dance.raw

Figure 14 Yellow channel for Bear.raw

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Figure 15 CMY representation for Dance.raw

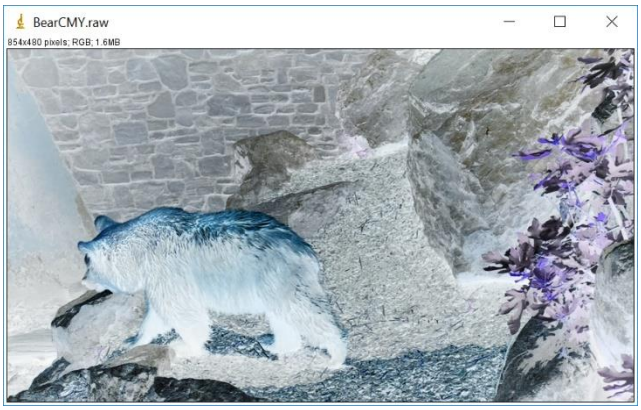


Figure 16 CMY representation for Bear.raw

Figure 17 Cyan when using type float for Dance.raw

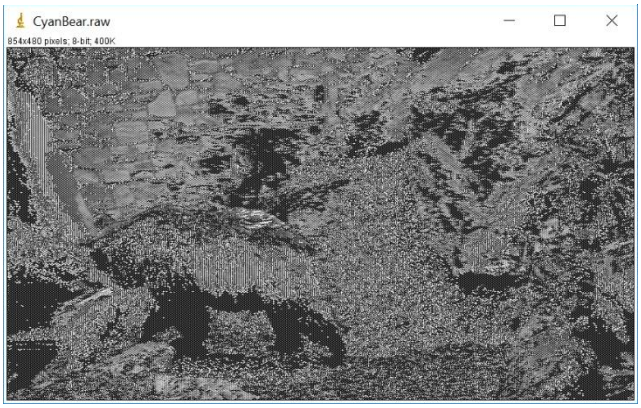


Figure 18 Cyan for bear when using type float for Bear.raw

Discussion

The images above were generated using the formula described in the approach. Figure 7 and 8 show the original input RGB 24-bit images for Dance and Bear respectively. Then the individual channels were generated and the negatives were taken to satisfy the relation shown in equation 1.

Figure 9 and 10 show the Cyan channel for the two images, 11 and 12 are Magenta channels, 13 and 14 are the Yellow Channels, all of which are 8-bit images. Figures 15 and 16 are the CMY colour model representation for Dance and Bear where all the three channels C,M and Y have been combined to produce a 24-bit image. These images were calculated by using datatype *unsigned char* to define the arrays containing input and output images.

Images 17 and 18 are the Cyan channels for the two images when using type *float* for the two images. We can see that the images are extremely granulated and it is difficult to infer the images they are trying to depict.

From all these images below, we can clearly see that CMY is the complement of RGB, as the definition implies. It makes sense that the CMY colour model has applications in printing and other display devices that are not self-Luminant and reflect the light that is incident on it. Due to its subtractive nature, the colours we see will be RGB.

Printers have an additional colour Black added to them to avoid otherwise having to use C, M and Y to achieve black. The result of such a high density of ink on paper will not dry quickly enough and cause *Bleeding.* The addition of black gives us the CMY(K) model.

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

b). Image Resizing via Bilinear Interpolation

Abstract

Bilinear interpolation is an extension of linear interpolation. It is basically linear interpolation applied in two directions.
In Linear interpolation we basically spread the pixels to meet the new dimension requirements and fill the space between them with a weighted sum of the neighbouring pixels.
For bilinear interpolation because we are increasing the dimensions in both height and width, we just repeat linear interpolation in both horizontal and vertical directions. This is depicted in Figure 19 and is explained below.
Bilinear interpolation is usually implemented after all other transformations like warping, rotating etc have already been applied. This is because these transformations move pixels tend to move around during transformations and will cause distortions and visually unappealing images if applied after bilinear interpolation.

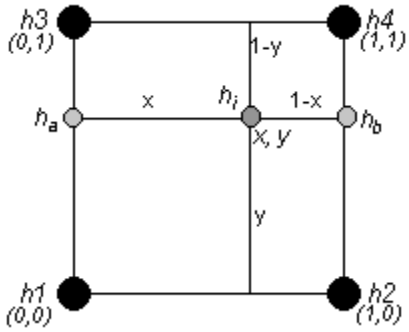


Figure 19 Bilinear interpolation

Approach

The approach here is explained with respect to the parameters used in this assignment. The aim is to resize Airplane.raw from 512 x 512 to 650 x 650. I first began by calculating the ratio of rescaling.

$$ratio = \frac{512}{650}$$

Looking at Figure 19, hi is the location of the pixel who's colour we are trying to estimate. h_1, h_2, h_3 and h_4 are the pixels of the smaller location who's colour is already known and will be used to calculate hi. We begin by using linear interpolation to calculate the colours of h_a and h_b. 1-y is the distance between ha and h3 and y is the distance between ha and h1. Y and 1-y is the weighting factors used to calculate ha. hb is also calculated in the same way. We then use h_a and h_b and through linear interpolation again, calculate the value of hi.
Because we are using a smaller image to populate a larger one, we use the ratio to locate the pixels in the smaller image. The weighting functions are calculated as;

$$x = (ratio * horizontal\ location\ of\ h3) - floor(ratio * horizontal\ location\ of\ h3)$$
$$y = (ratio * vertical\ location\ of\ h3) - floor(ratio * vertical\ location\ of\ h3)$$

Because we are using the 4 surrounding pixels to calculate the one between, we can combine the two linear interpolations into one generic bilinear interpolation formula given by;

$$\begin{aligned} OutputImage[i][j] = {} & inputImage[ratio * i][ration * j] * (1-x) * (1-y) + \\ & inputImage[ratio * i][ration * j + 1] * (1-x) * (y) + \\ & inputImage[ratio * i + 1][ration * j] * (x) * (1-y) + \\ & inputImage[ratio * i + 1][ration * j + 1] * (x) * (y) \end{aligned}$$

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Experimental Results

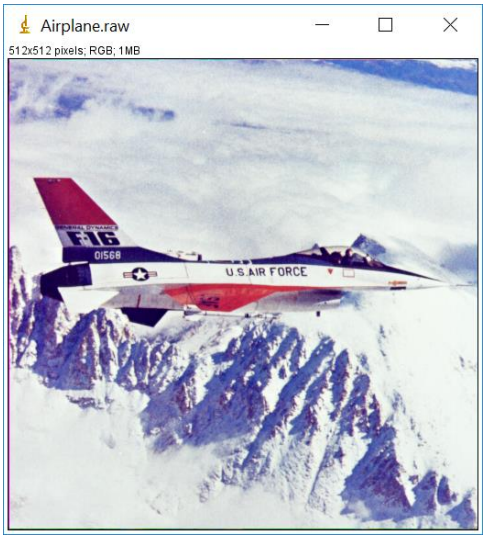


Figure 20 512 x 512 Airplane.raw

Figure 21 650 x 650 output of Bilinear Interpolation

Discussion

The output figure 20 and 21 show that the bilinear interpolation has been executed successfully and the new image looks pretty much looks the same as the input image except for some loss of clarity. This loss of clarity will be greater as the scaling factor increases leading to blurrier images. There are many image resizing algorithms. Bilinear is better than image resizing but Bicubic interpolation gives even better results. Where in Bilinear interpolation we use the 4 neighbouring pixels, in Bicubic interpolation for the unknown pixel x in amplified image, its influence sphere is extended to its 16 neighbouring pixels. The color value of x is calculated by these 16 pixels according to their distance to x. Regardless, Bilinear interpolation remains a very effective algorithm for image resizing.[3]

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Problem 2: Histogram Equalization

a). Histogram Equalization

Abstract

Histogram equalization is used to alter the contrast of an image by adjusting the intensity of the pixels in the image.
A simple representation is shown by Figure 22. The histogram on the left is that of the image which is equalized to look like the one on the right. Histogram equalization is applied to all the pixels of the image to better distribute the intensities. This basically spreads out the histogram so it occupies a greater more equalized range of values.

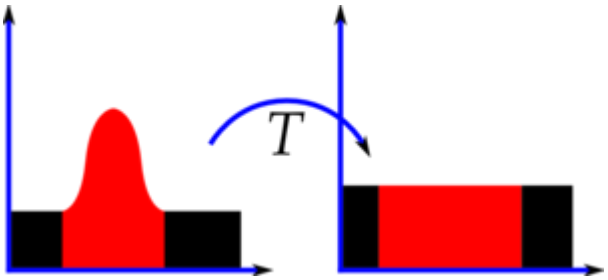


Figure 22 Histogram Equalization

There are various ways of achieving this, but for this assignment we consider two main types;

A. Transfer function based histogram equalization- In this method we use a look up table to increase the spread of the histogram.
B. Cumulative probability based histogram equalization method- in this method we use the bucket filling method to end up with each intensity having the same number of pixels.

B gives a more uniform looking distribution while A gives a just spreads out the existing histogram.

Approach

Method A

For histogram equalization using method A, a lookup table was created using the following steps:

- Calculate the histogram for the Red channel.
- Find the probability distribution function for each intensity.
- Calculate the cumulate probability distribution.
- Multiply by a scaling factor (256 for this assignment) so the histogram now covers this spread.
- Repeat for the Green and Blue channel.
- Write back into the output image to get the equalized histogram.

Method B

For histogram equalization using method B, I used the following steps:

- Calculate the total number of pixels per channel in the image- The *Desk.raw* is of the dimensions 300 x 400 pixels. This gives a total of 120000 pixels.
- Calculate the number of bins- here it will be the number of intensities i.e, 256.
- Calculate the number of pixels per bin. This comes to 120000/256=468.75
- Distribute the pixels so each pixel has the same number of pixels. I began by assigning 468 pixels to each intensity. This was done going through the image in order of increasing intensity. The first 468 0^{th} intensity pixels were retained as 0. The 468 were 1s and so on until all 256 bins had 468 pixels each. This left out 192 pixels. I went through the entire image again and assigned an extra pixel to the first 192 intensities. This resulted in a uniform distribution of 469/468 pixels per intensity. I used a flag to ensure that I wasn't changing the intensity of the same pixels repeatedly. Going through the pixels in increasing order of intensities ensures that there are no drastic changes in intensity of each pixel. This increases the contrast of the overall image while retaining the original appearance.

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Experimental Results

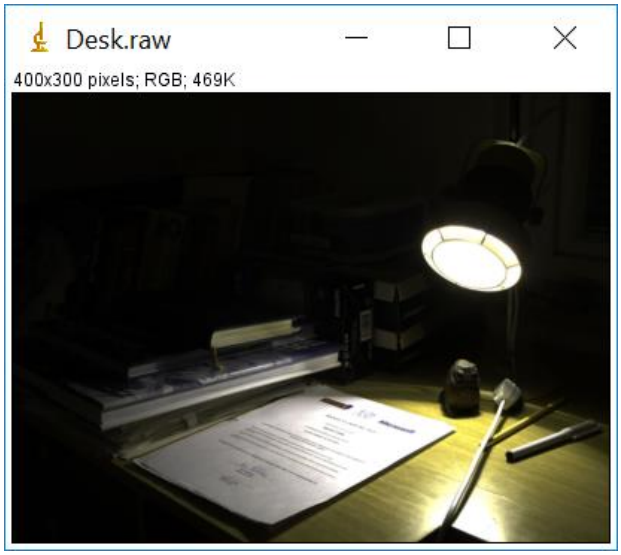


Figure 23 Original Desk.raw

Figure 24 Method A

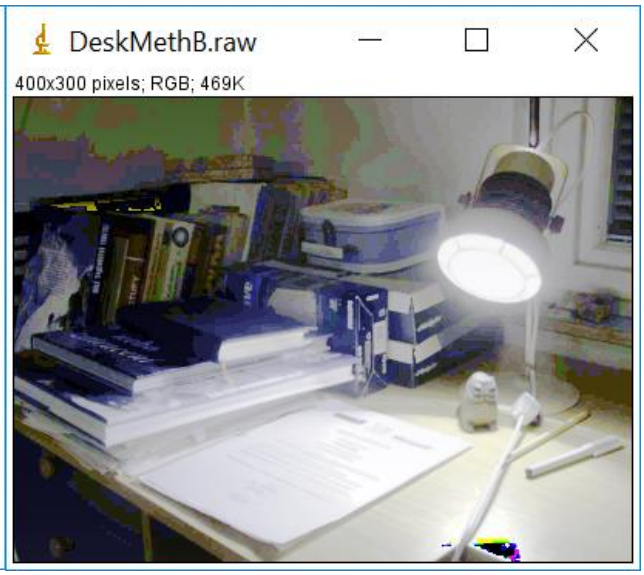
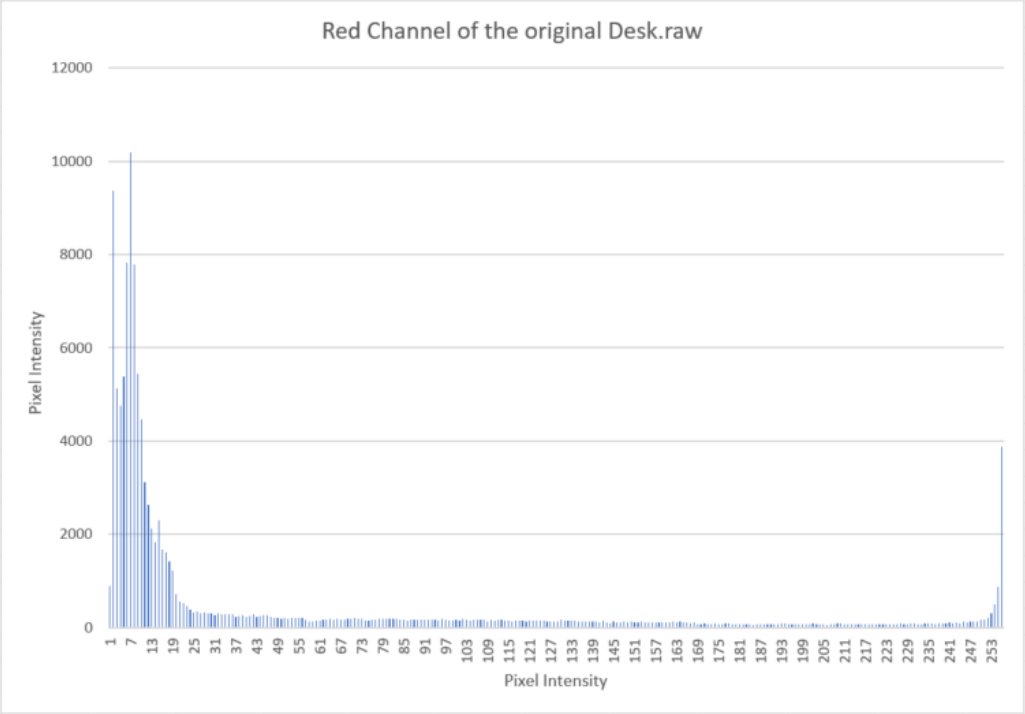


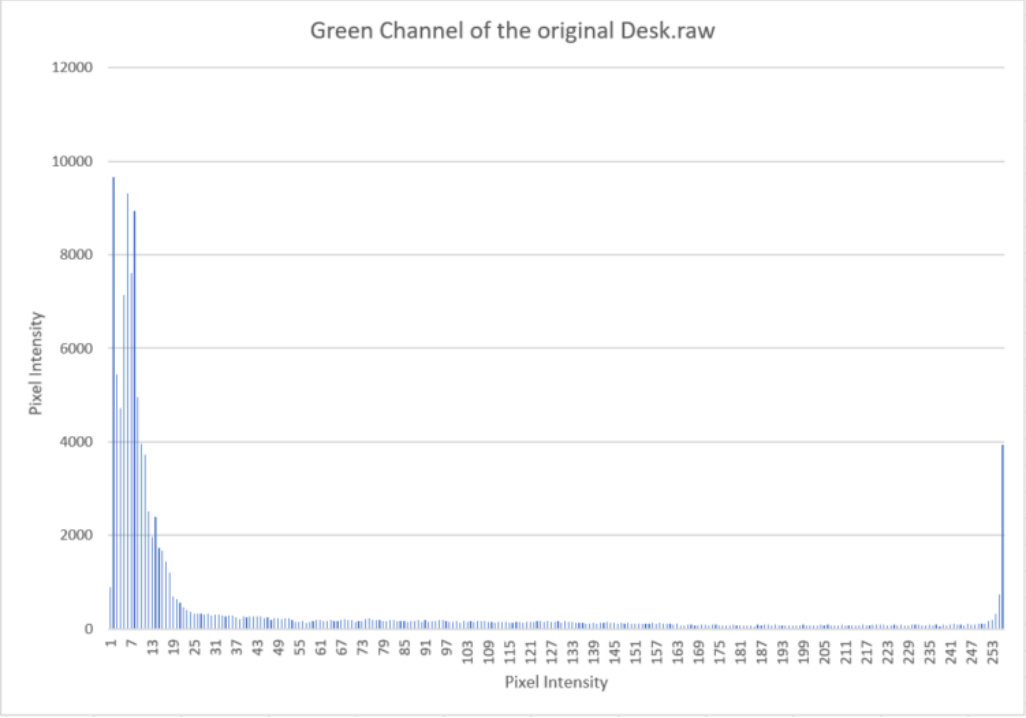
Figure 25 Method B

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Tabular Results
Method A

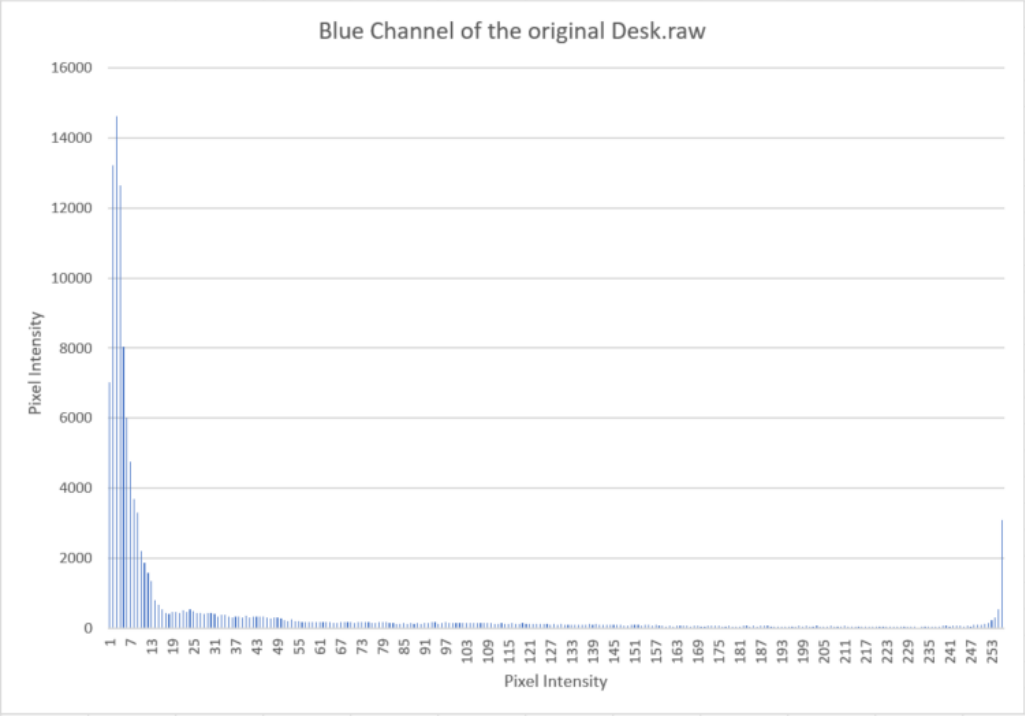


Graph 1

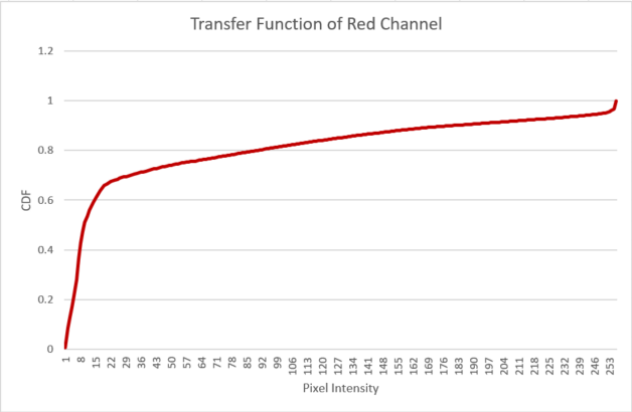


Graph 2

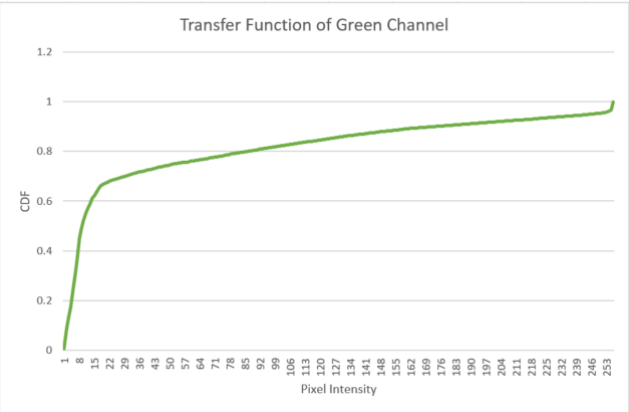
Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1



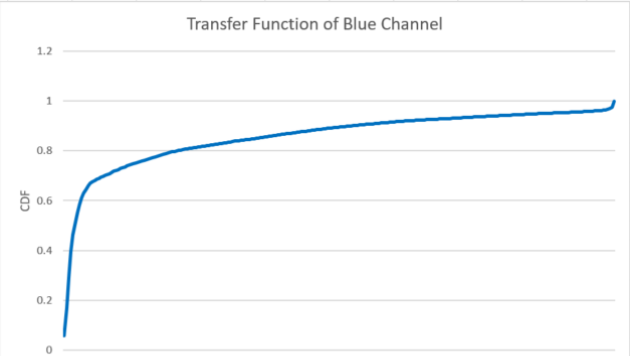
Graph 3



Graph 4

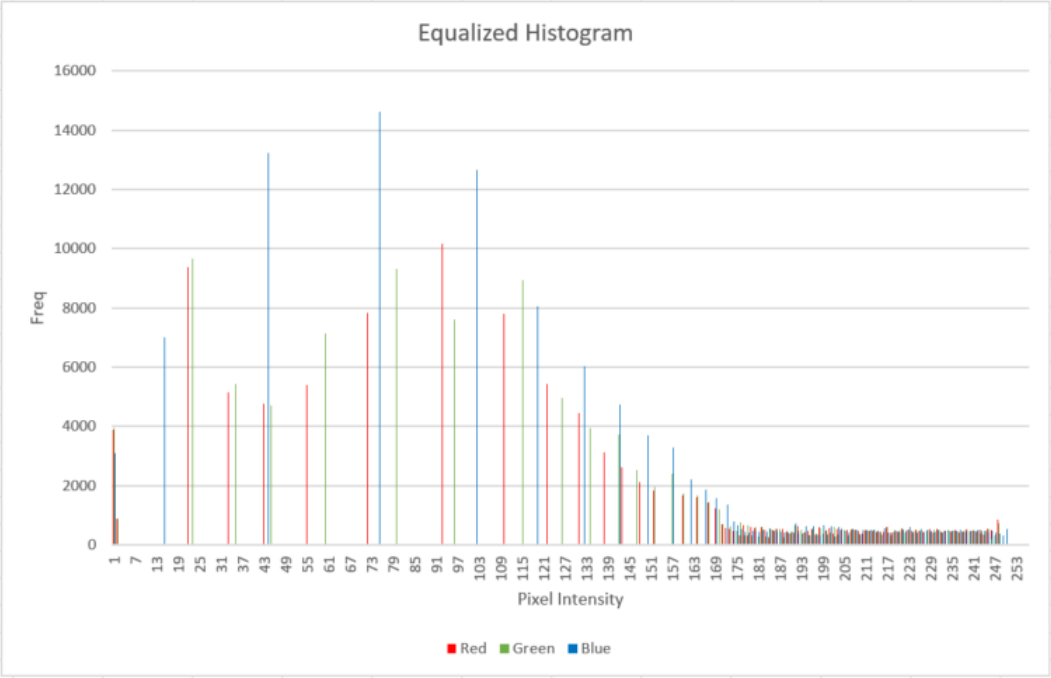


Graph 5



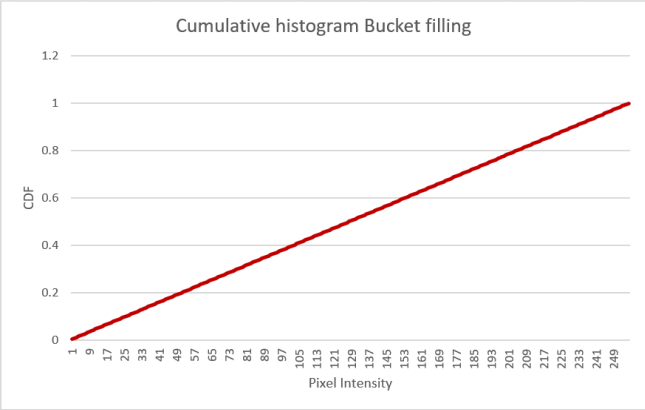
Graph 6

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

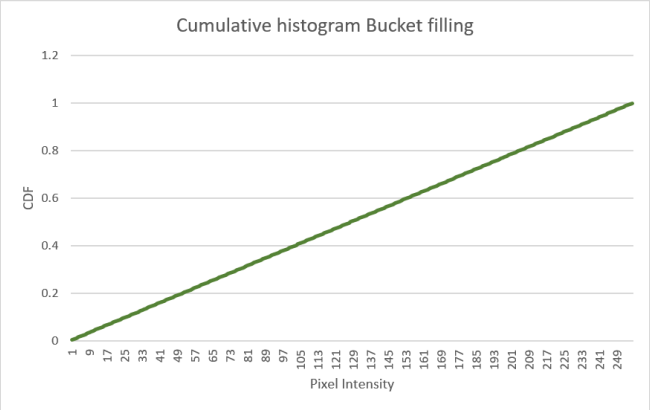


Graph 7

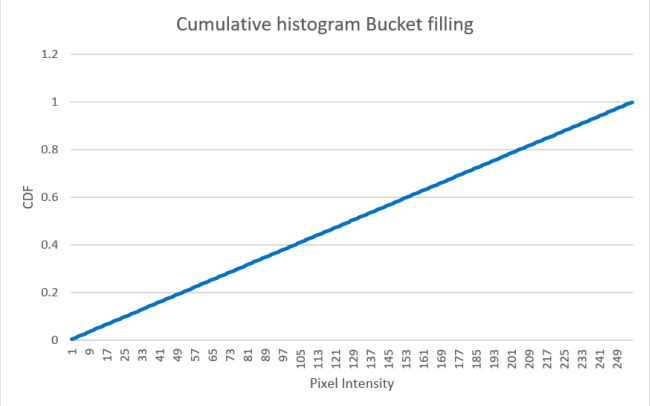
Method B



Graph 8



Graph 9



Graph 10

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Discussion

Graphs 1, 2 and 3 show the histogram for the original *Desk.raw* image. We can see that the graphs are clumped at either end. It has a large number of very dark and very light pixels and very low number of pixels for all the intensities. Hence the image is very dark and doesn't have much visible details.
Graph 4, 5 and 6 are the transfer functions calculated as part of the look up table. The CDF is non-linear and when multiplied by 256 gives more spread out histogram as seen in Graph 7.
The output image is shown in Figure 24. With the intensities more evenly spread out, more of the image is now visible. But the image is not very visually appealing. It while it makes backgrounds more visible, it also tends to amplify noise. This is because for histogram equalization, a lot of individual intensities are merged. This leads to loss of fine details in the images leading to loss of edges and artifacts are generated. This is most visible near the book shelf.
Method B gives us very linear cumulative histograms as compared to Method A (Graph 8, 9 and 10). It has similar issues as that of Method A. Because this method is more rigid and intense in terms of equalization, the artifacts in the background and under the lamp are more obvious. The resultant image as shown in Figure 25 looks more unappealing compared to method A.
To overcome these drawbacks, we can use the method called Histogram Equalization with Maximum Intensity Coverage (HEMIC). Intensities lost due to Histogram Equalization can be restored using a weighted combination of input and enhanced images. "The input colour image in Red–Green–Blue (RGB) is first stretched to span the permissible magnitude range, and converted to the Hue–Saturation–Intensity (HSI) space where the I-channel is further processed. Then it adopts a weighted sum strategy that combines a histogram equalized image with the original input image, with aims to filter out artefacts that are incorrectly introduced. The weighting parameter selection is made automated by using the golden section search algorithm. The enhanced intensity image is then reconverted to its original colour states for future processing." [4]

b). Image Filtering – Creating Oil painting effect
Quantization
Abstract
Quantization is the process of reducing the number of distinct colours used to display an image. An image captured by a camera has an infinitely continuous number of colours. We convert this into a digital image consisting of a limited number of colours while retaining all the details of the image. Normally 24-bits gives us enough colours so that the human eye cannot perceive the difference between the original and the digitized image. This is especially useful in printing and display devices as it reduces the complexity of the image while retaining clarity.
A normal 24-bit RGB image has 256 x 256 x 256 possible colour combinations. For this assignment we are reducing this to a total of 64 colours. This mean that each channel will have 4 colours.

Approach
The steps followed to obtain the quantized image of only 64 colours are;

- Calculate the total number of pixels in the image.
$$Total\ Pixels = Height * Width$$
- For a total of 64 colours, each channel needs to have only 4 colours. So, to find the number of pixels in the 4 bins each of which represents a colour
$$Pixels\ per\ bin = \frac{Total\ Pixels}{4}$$
- This gives the threshold value for each bin.

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

- I then created a histogram and calculated the cumulative sum to find the intensity values which define the edges of each bin.
- The average intensity per bin is found by multiplying each pixel intensity by the frequency at that intensity, all divided by the total number of pixels in that bin.
- Colour quantization is achieved by using the new look up table convert the 256 colours per channel into the 4 quantized colours.

For the Oil Painting effect I used the Quantized image- An area of NxN is analyzed and the pixel at the center is replaced by the most frequently occurring pixel intensity in the NxN neighbourhood. To repeat this for the outermost rows and columns, the image is padded by reflecting the required numbers of rows and columns to complete the NxN filter mask.

Experimental Results

Quantization

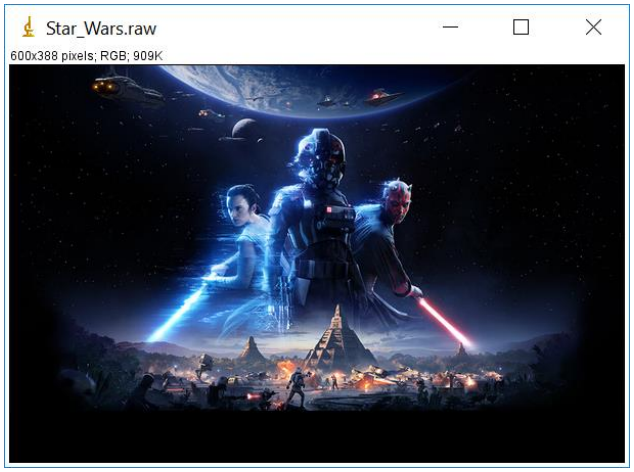


Figure 26 Original Star_wars.raw

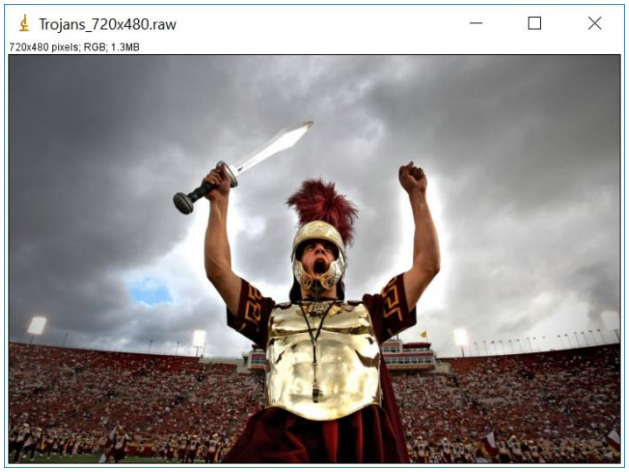


Figure 27 Original Trojans.raw

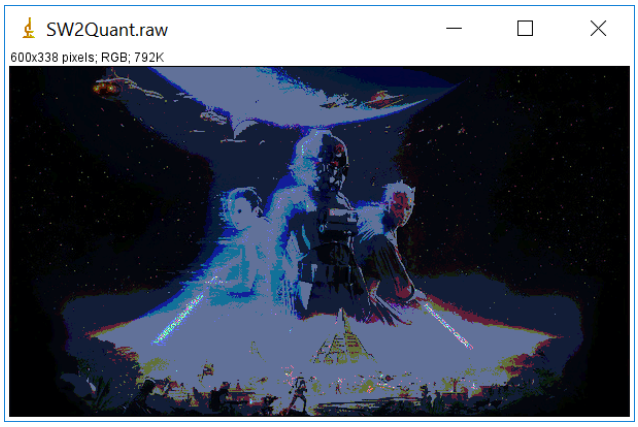


Figure 28 64 colour Star_wars.raw

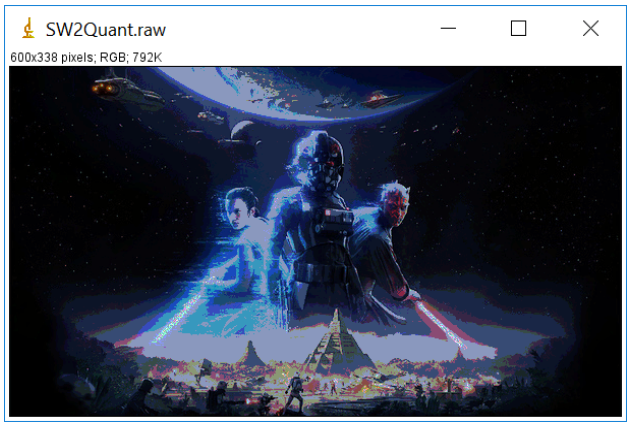


Figure 29 512 colour Star_wars.raw

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

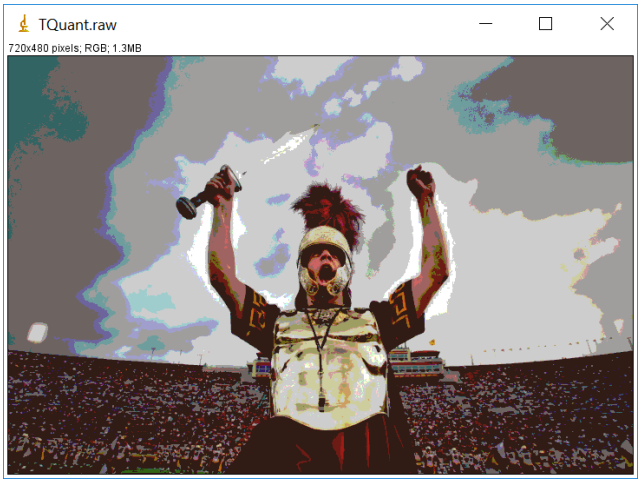


Figure 30 64 colour Trojans.raw

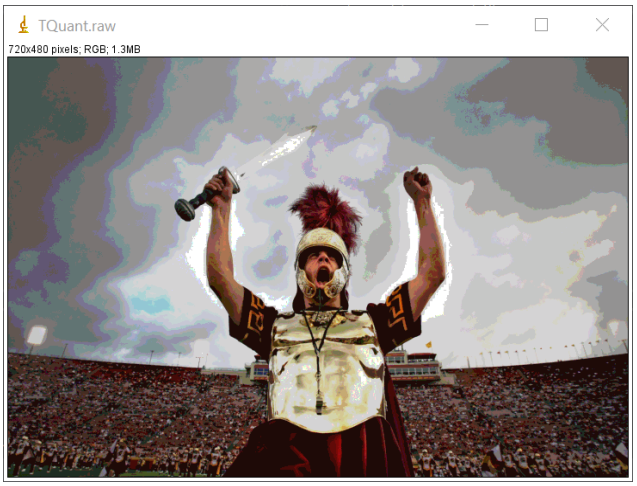


Figure 31 512 colour Trojans.raw

Oil Painting Effect
64 colours, 3x3 Filter

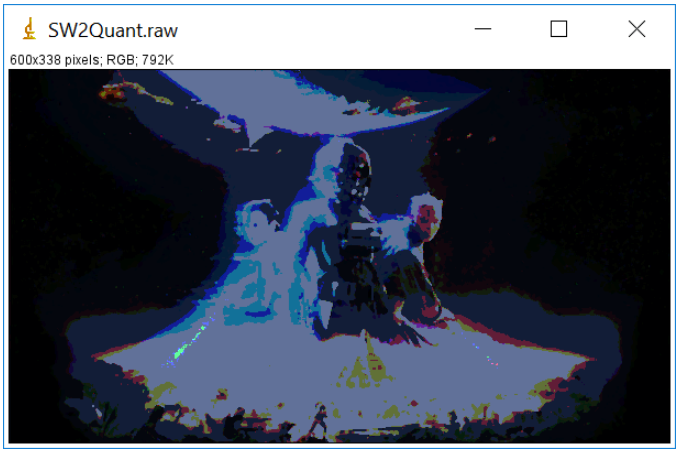


Figure 32

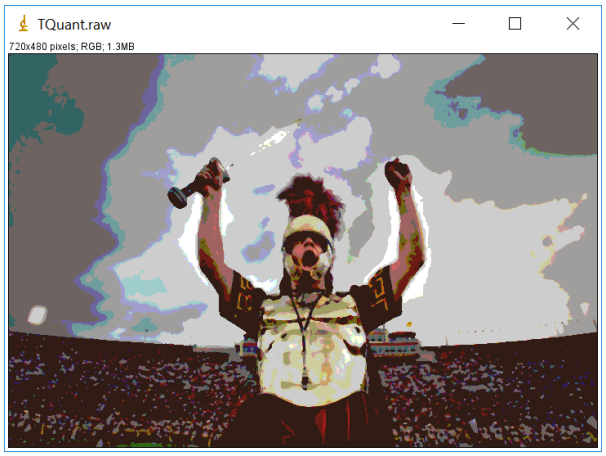


Figure 33

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

512 colours, 3x3 Filter

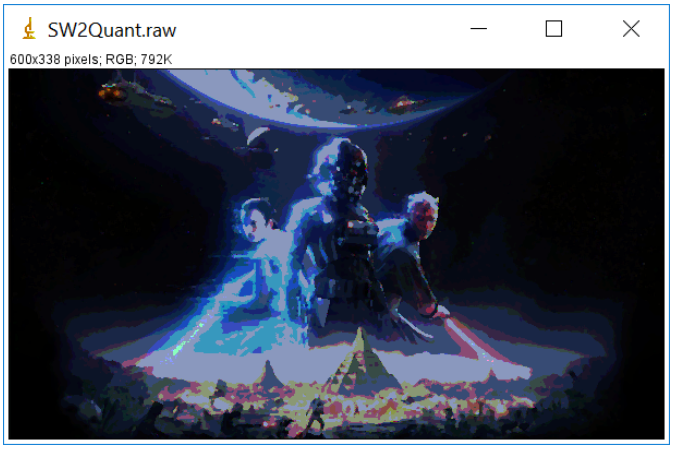


Figure 34

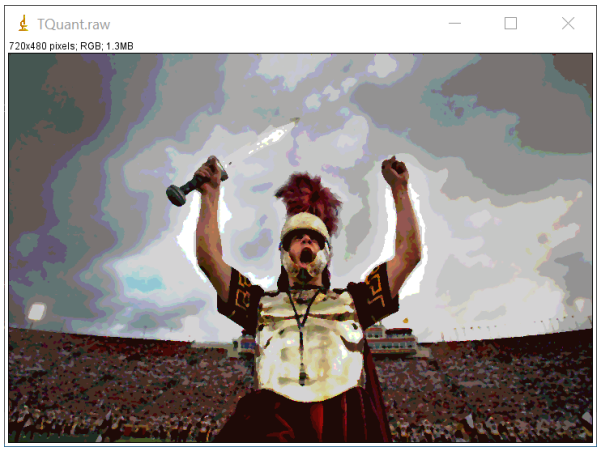


Figure 35

64 colours, 5x5

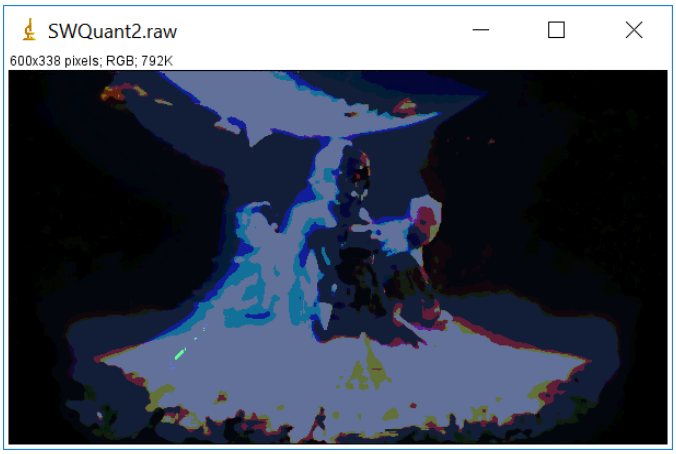


Figure 36

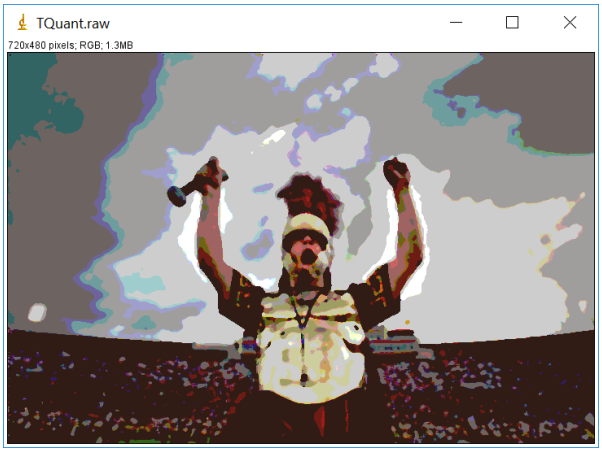


Figure 37

64 colours 7x7

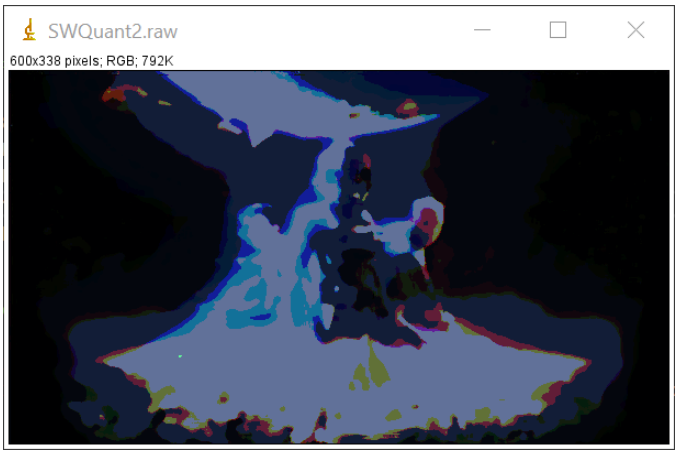


Figure 38

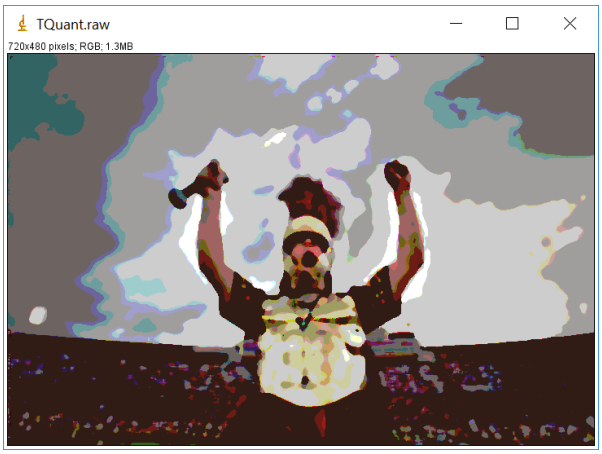


Figure 39

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

64 colours, 9x9

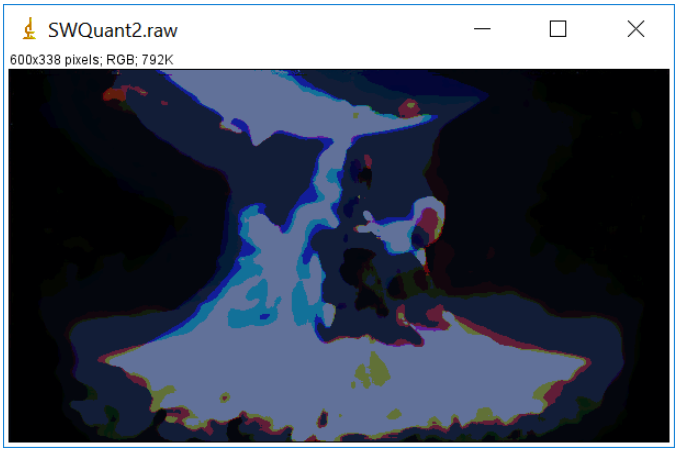


Figure 40

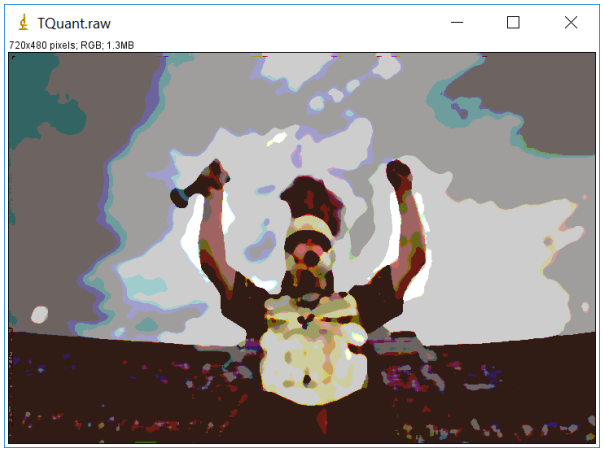


Figure 41

64 colours,11x11

Figure 42

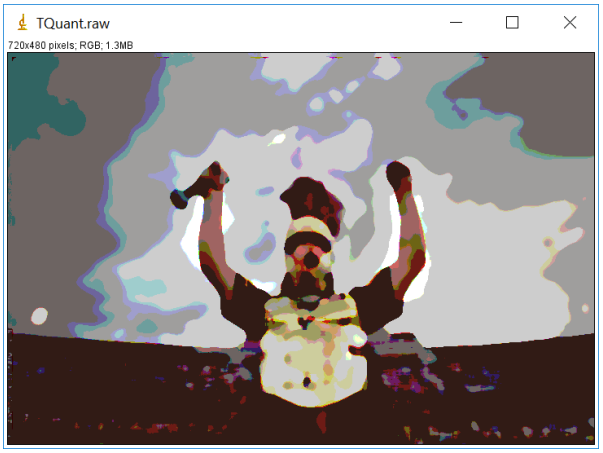


Figure 43

512 colours, 5x5

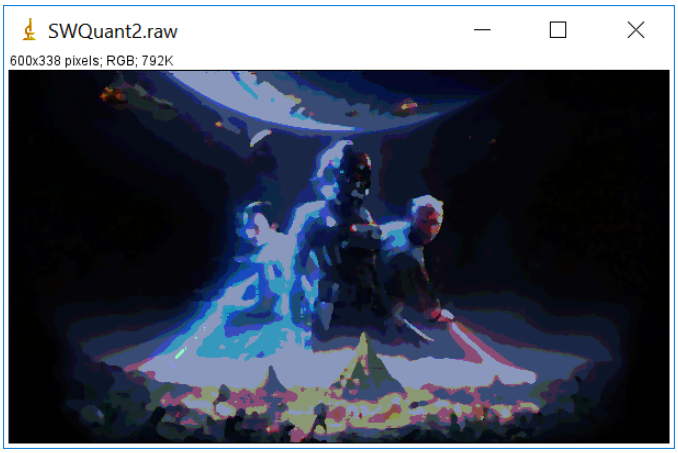


Figure 44

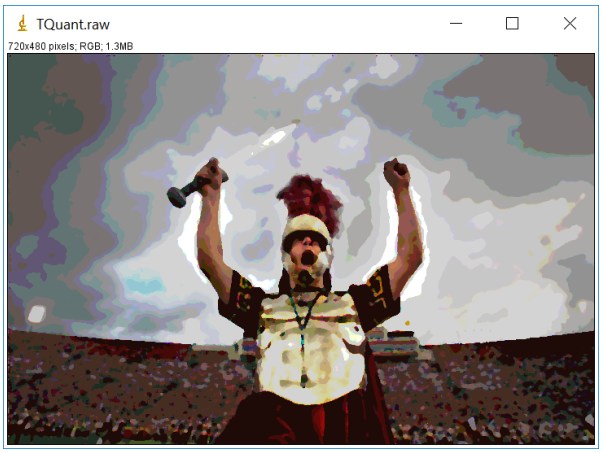


Figure 45

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

512 colours, 7x7

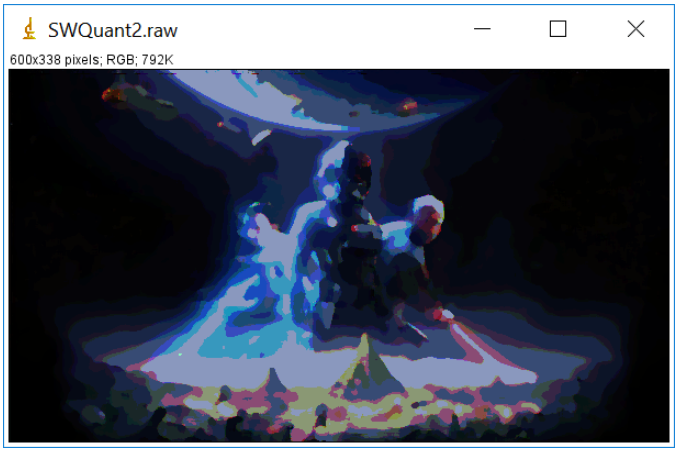


Figure 46

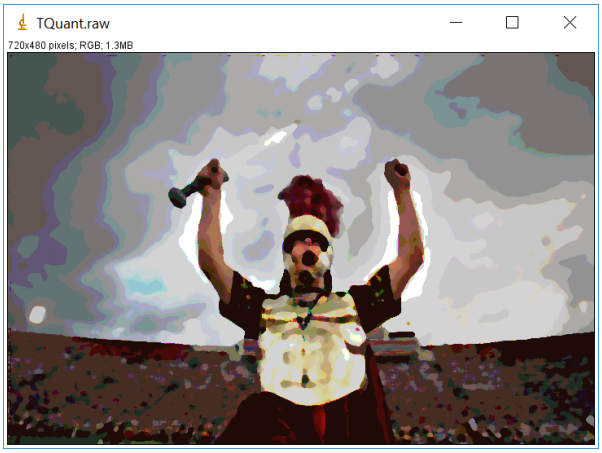


Figure 47

512 colours, 9x9

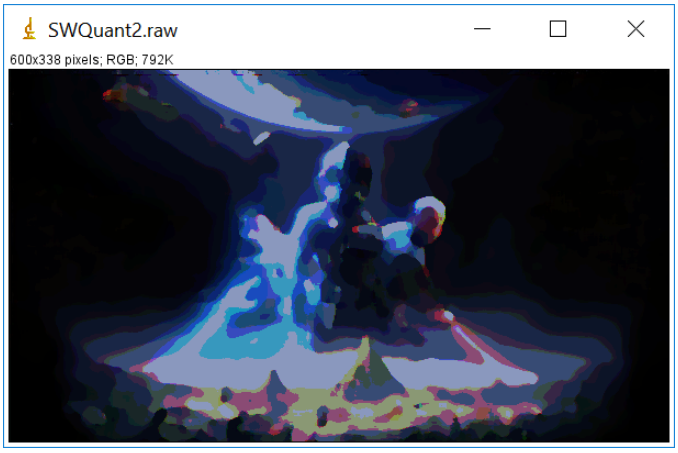


Figure 48

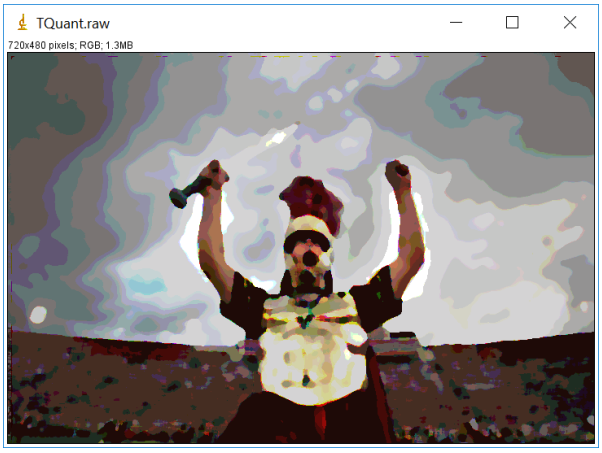


Figure 49

512 colours, 11x11

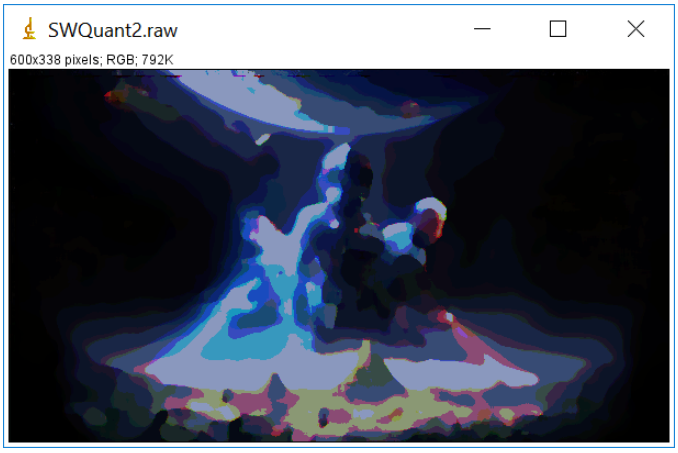


Figure 50

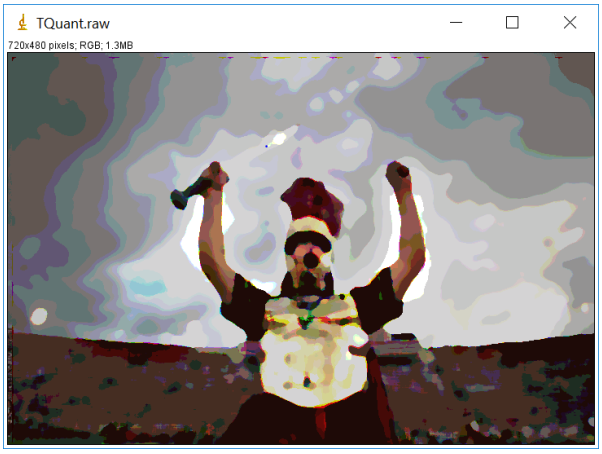


Figure 51

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Discussion

The choosing of threshold values has been discussed in the approach part. Quantizing for 64 colours, as seen in Figure 28 and 30 , shows that a lot of the finer details in the image are lost. Comparing this to 512 colours as seen in Figure 29 and 31 , shows that while there is still some obvious differences compared to the original images, it is closer to the original and the finer details (like the pyramid region in Star_Wars.raw and the sword in Trojans.raw) are clearer. These qualities are true for both images. In Star_Wars we can also observe a loss of edges and reduced contrast due to quantization. In Trojans.raw, continuous/smooth colour transitions like in the clouds are lost and the image no longer looks very appealing. As far as quantization is concerned, quantizing for a larger number of colours retains most of the original features of the image. More colours equals less differences perceived by the eye.

The oil painting filter is now applied for both quantization levels of both the images in varying dimensions of N. Beginning with Star_Wars, as the dimensions of N increases, the image gets increasingly blurry. For 11 x 11 the image is no longer discernable and is very hard to identify unless you are aware of the original. All the finer details of the image are lost.

This trend is also observed in Trojans.raw. But because the image has brighter and more defined edges and colours as compared to Star_Wars, it is still discernable at 11x11. For 64 colours, considering both Star_Wars and Trojans, 3x3 or a maximum of 5x5 Oil painting filter is better as it helps retain most of the details of the original image before getting too blurry.

The 512 colour versions fare slightly better than the 64 colours for both images. At 512 colours, even for larger filter dimensions, edges and finer details of the images are retained. In Star_Wars, the region around the pyramid is not completely whitewashed and can be identified at 7x7 filter dimensions also. Consequently, the same applies for Trojans where the colour transitions in the clouds are less drastic and the image looks identifiable with a filter size of 11x11 also, although it is significantly blurrier than the lower filter dimensions. Hence for 512 colours I think a filter size of max 5x5 produces the most visually appealing results with relatively smooth transitions of colour and preservation of edges.

C). Image Filtering–Creating Film Special Effect

Abstract

This problem follows the concept of Histogram matching where you have an image whose histogram is known and we use it to apply the same transformations to another image of choice. Histogram equalization as used in the previous methods are also a way of histogram matching where the specified histogram had uniform distribution.

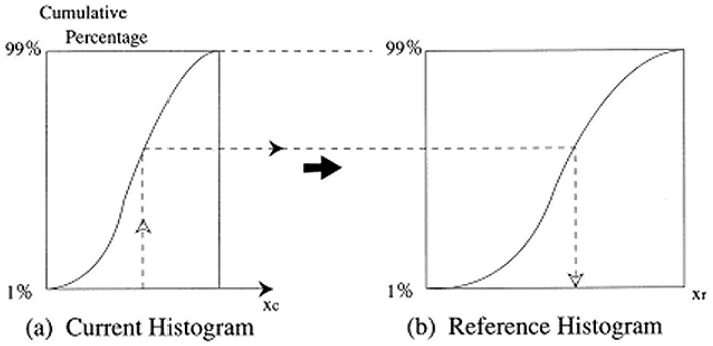


Figure 52 Histogram matching

Histogram matching can be either by generating a look up table and matching the CDF values of the source image to that of the destination image or by identifying the individual transformations and then comparing the Histograms of the original and output images and modifying them (histogram stretching or shrinking) to make them look identical.

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Approach

For this assignment, I began by trying to match the histograms using a look up table. This method got a little complex and I couldn't debug some of the errors and the image wasn't very similar to the desired output. I then used the second approach.

- I began by finding the negative of the image and then found the mirror reflection of it.
- I computed the histogram for each channel of this image.
- I also computed the histogram values of the desired output- Film.raw
- The histograms of Film.raw were more compressed compared to my image, hence Film.raw had lower contrast.
- To achieve this effect, I noted down the maximum and minimum values of each channel of the Film.raw image and applied Histogram Shrinking to my image by using the formula:

$$Shrink\big(I(x,y)\big) = \left[\frac{Shrinkmax - Shrinkmin}{I(x,y)max - I(x,y)min}\right][I(x,y) - I(x,y)min] + Shrinkmin \quad \text{--------(2)}$$

- After identifying the transformation, the same is applied to Girl.raw.

This formula is applied channel by channel where

Shrink(I(x,y)) is the resulting output image.
I(x,y) is the input image.
Shrinkmax is the max intensity of the Film.raw for that channel.
Shrinkmin is the min intensity of the Film.raw for that channel.
I(x,y)max is the max intensity of our input image for that channel.
I(x,y)min is the min intensity of our input image for that channel.

Experimental Results

Figure 53 Original Image

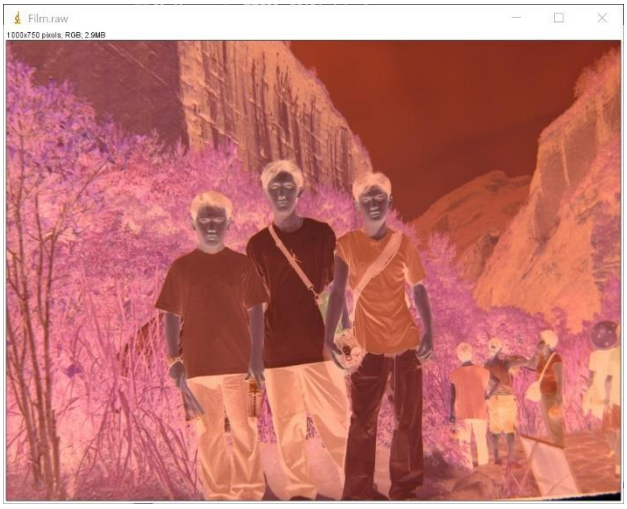


Figure 54 Desired output

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

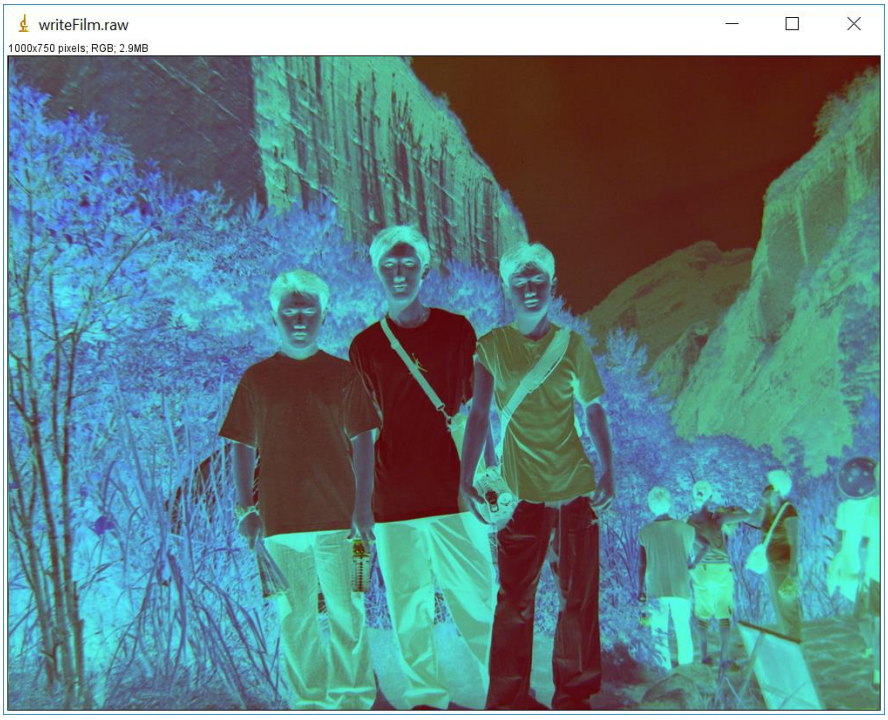
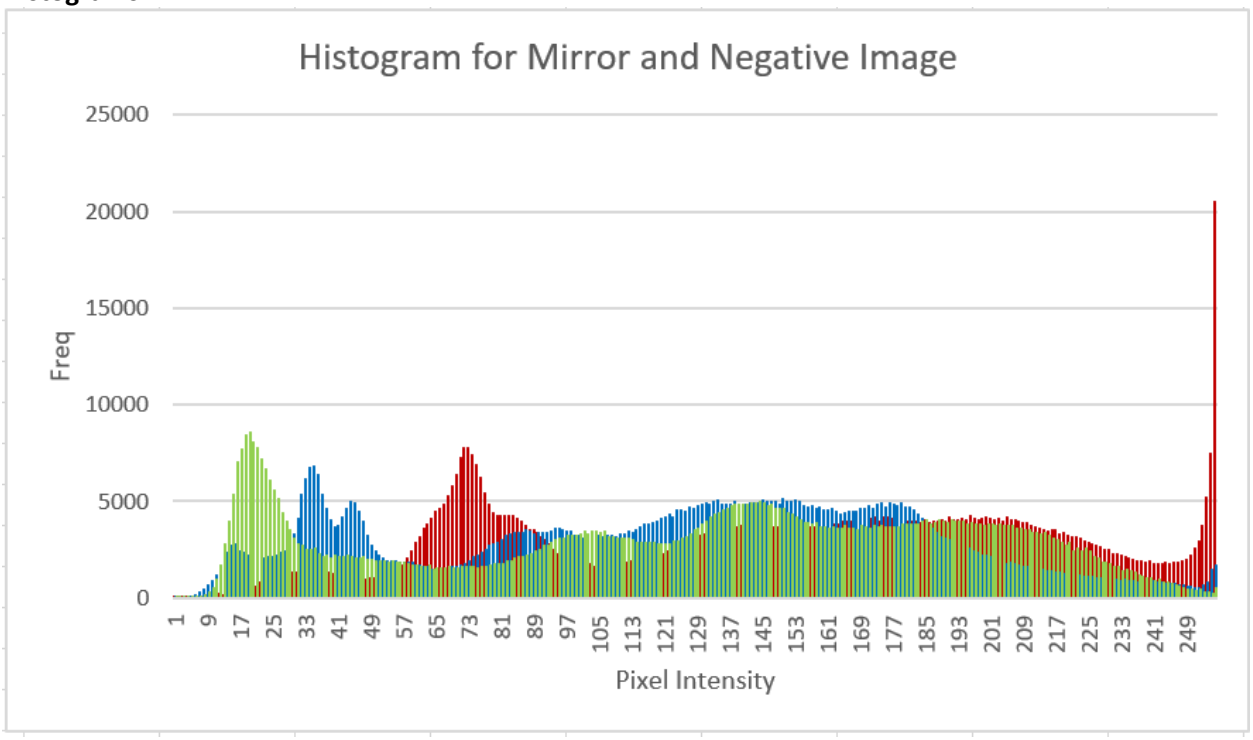


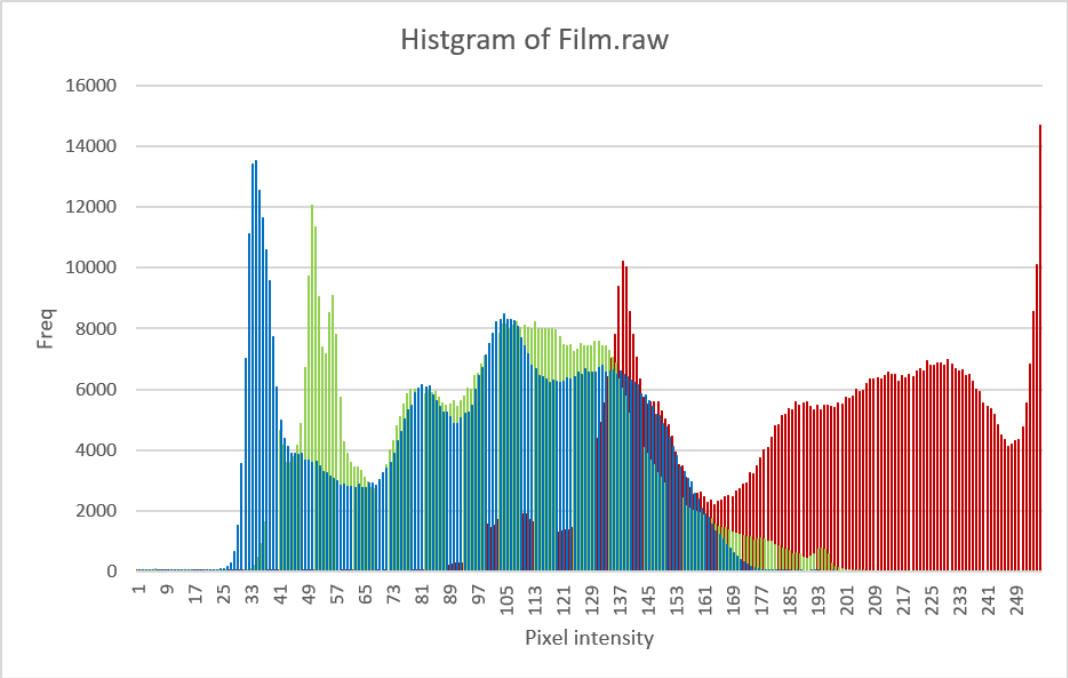
Figure 55 Output after taking negative and mirror image

Histograms

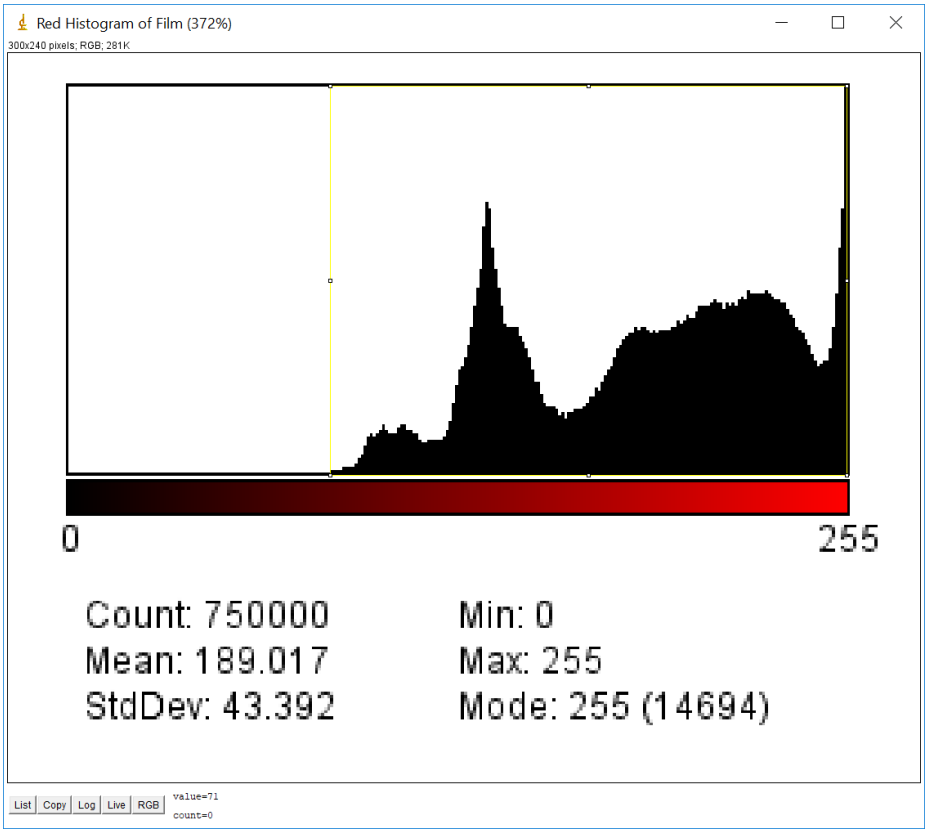


Graph 11

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1



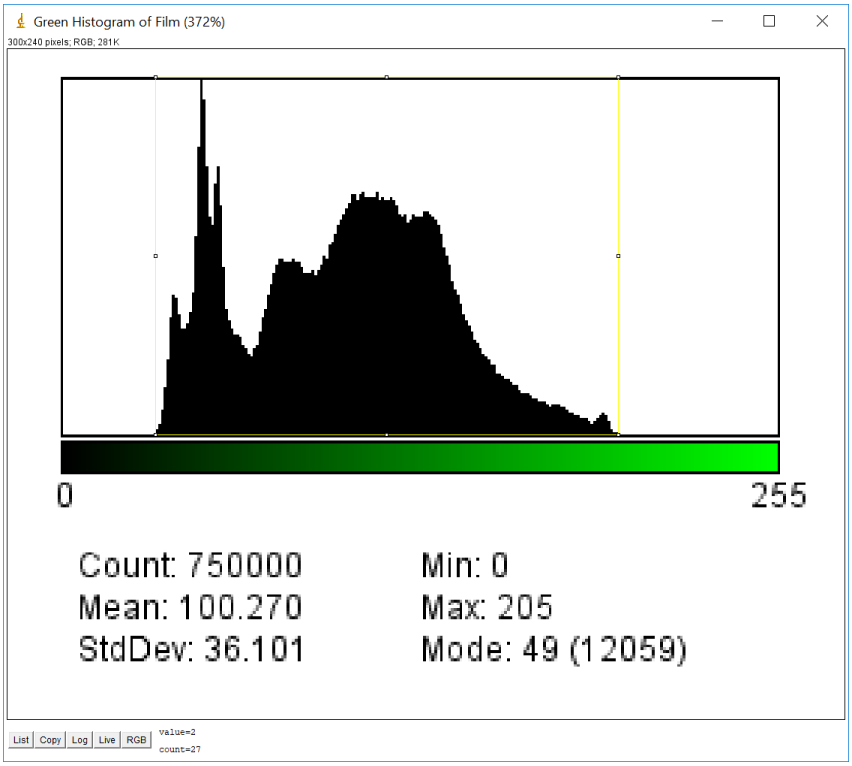
Graph 12



Min=86
Max=255

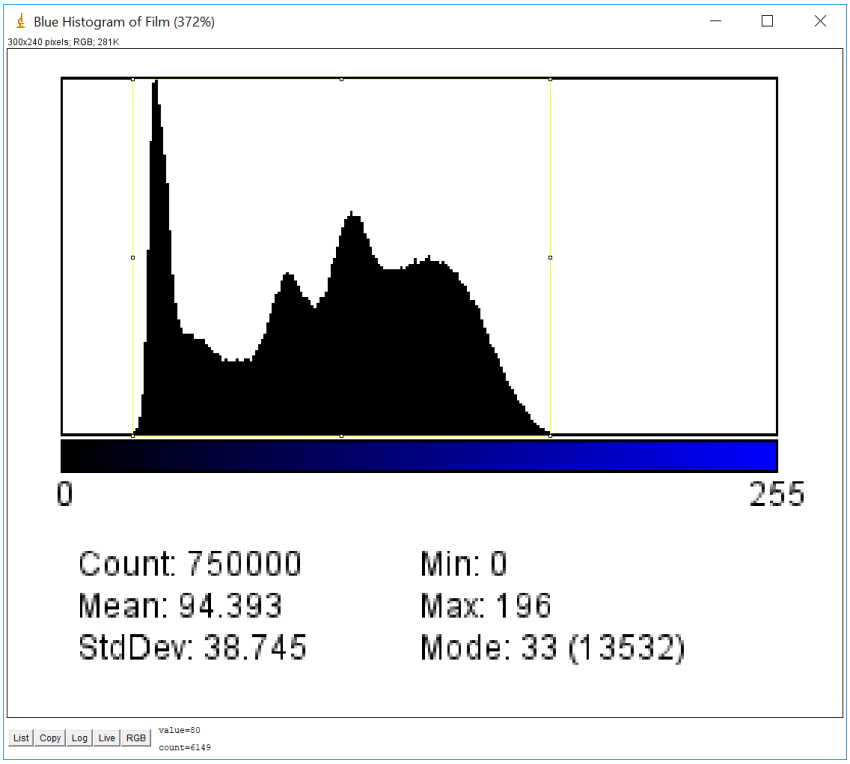
Graph 13 Calculating min and max of Red Channel

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1



Min= 33
Max=205

Graph 14 Calculating min and max of Green Channel



Min=24

Max=196

Graph 15 Calculating min and max of Blue Channel

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Output Image

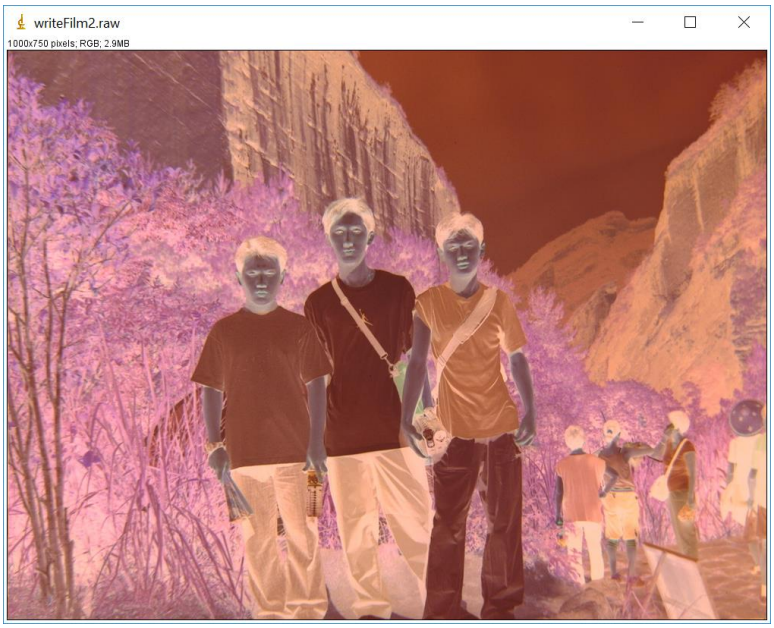
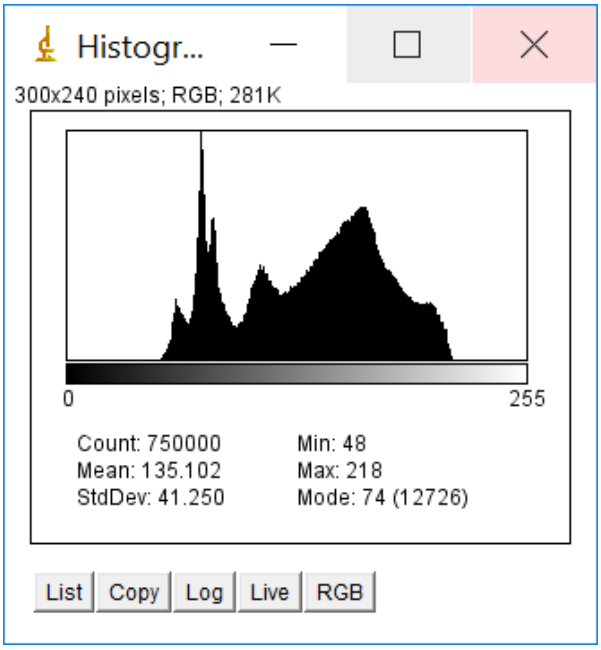
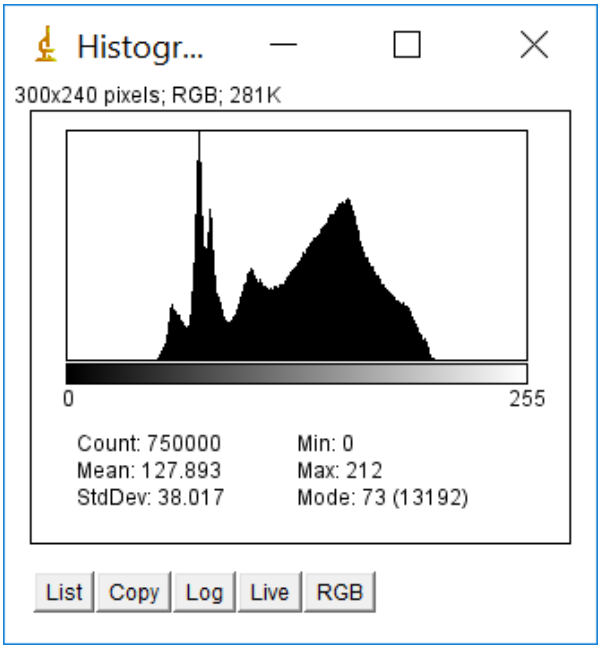


Figure 56 Final output Image



Graph 16 Histogram for final Output Image



Graph 17 Histogram for original film effect Image

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Figure 57 Input Girl.Raw image

Figure 58 Transformed Girl.raw image.

Discussion

Figure 53 and 54 show the original input image and the desired film effect to be applied respectively. Figure 55 shows the output image obtained when the original image has been taken the negative of and flipped. I then computed the histogram for both Figure 54 and 55. This is depicted by Graph 11 and 12. The aim here is to transform Graph 11 to make it look more like Graph 12.

To achieve this, I separated graph 12 into its 3 component channels. I then used that histogram to calculate the most visible minimum and maximum range. This is highlighted by the yellow box in Graphs 13-15. The minimum and maximum values calculated were then fed into equation 2 to gain the final output image as shown in Figure 56.

Comparing the histogram for Figure 56 and Figure 55 (as shown in Graphs 16 and 17) we can see that they are near identical. It is impossible to map histogram perfectly, but this is a very good result as the Figure 55 and 56 appear almost identical to the human eye. If required, this can be further improved by testing various minimum and maximum values.

I then used this transformation and applied it to the *Girl.raw* image (Figure 57) to achieve the required special film effect transformation as depicted in Figure 58.

Problem 3: Noise Removal

Abstract

Sudden and random variations in the colour and brightness of an image which effects the appearance of the image is called Image Noise. This can arise due to a variety of reasons: noise when capturing the image, camera limitations, noise in the channel while transmitted, noise at the receiver etc. There are different types of noise that can affect an image- Uniform Noise, Gaussian Noise, Impulse Noise etc. For this assignment there are two main types of Noise that we are aiming to denoise- Gaussian Noise and Impulse Noise.

Gaussian Noise is basically noise that has Gaussian Distribution. It has a mean centered at zero, so the noise is most likely to take 0 values. Gaussian Noise mainly arises due to sensor limitations of poor lighting, transmission or other electronic circuit noise. Removal of Gaussian noise can be done using an averaging

or mean filter. The main drawback of this method is that it leads to smoothening of the image and loss of information at the edges.
Impulse noise also results in Salt and Pepper noise. This is random flecks of white and black pixels in the image cause by sharp and sudden disturbances. This can be cured using a median or thresholding filter.

a). Mix noise in colour image

Abstract

In this part of the assignment we have an image- Lena.raw which is affected by a mixture of Gaussian and Impulse noise. The aim is to reduce this noise and achieve the best possible PSNR value.

Approach

A Median filter and Mean filter is implemented for varying filter dimensions. We begin by implementing the median filter first to get rid of the salt and pepper noise first as these are more sudden bursts of value, which if averaged will affect all the neighbouring values and will be hard to get rid of.
The filters are also cascaded in various orders. PSNR value is calculated for all these combinations using the formula:

$$\text{PSNR (dB)} = 10\log_{10}\left(\frac{\text{Max}^2}{\text{MSE}}\right)$$

$$\text{where MSE} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(Y(i,j)-X(i,j)\right)^2$$

X : Original Noise-free Image of size $N \times M$

Y : Filterd Image of size $N \times M$

Max: Maximum possible pixel intensity = 255

PSNR value is calculated for each channel and the combination of filter dimensions and filter cascading that gives us the best PSNR value is used for denoising Lena.raw.

Experimental Results

Table 1

Filter(dimension)	PSNR Red	PSNR Green	PSNR Blue	Average PSNR
No Filter	17.0868	16.5475	17.0142	16.88283
Median(3x3)	23.8946	26.6293	23.9378	24.82056667
Median(5x5)	22.0337	22.6791	22.0324	22.2484
Median(7x7)	19.8985	20.0563	20.4838	20.1462
Median + Mean (3x3)	24.4693	27.0414	25.3345	25.61506667
Median+ Mean x 2 (3x3)	24.867	26.6223	25.719	25.7361
Median + Mean x 3 (3x3)	24.1481	26.068	25.7125	25.30953333
Median x 2 + mean (3x3)	25.7358	27.1514	25.7358	26.20766667
Median x 3 +Mean (3x3)	24.9063	27.0299	25.867	25.9344

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Noisy

Original

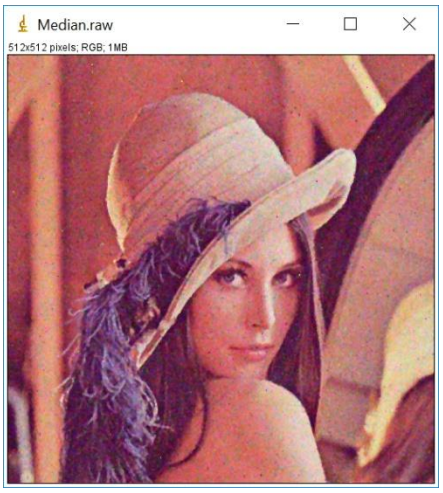


Figure 59 Median Filter (3x3)

Figure 60 Median filter (5x5)

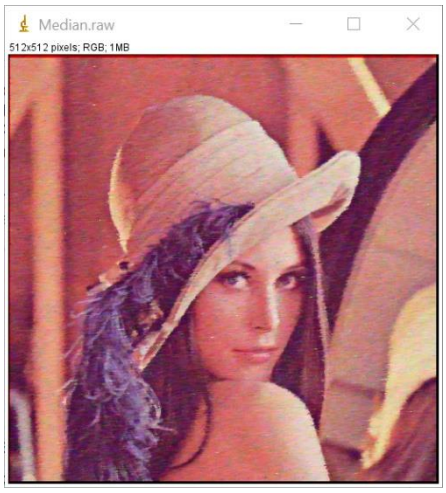


Figure 61 Median Filter (7x7)

Figure 62 Median + Mean (3x3)

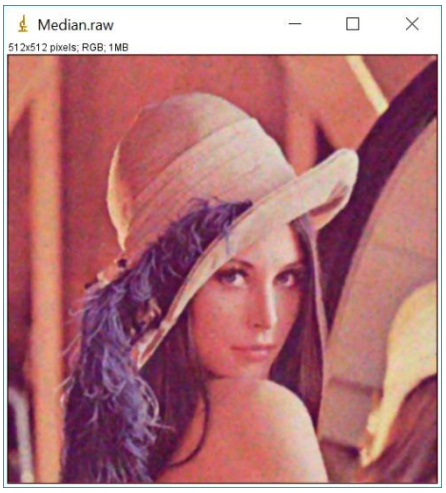


Figure 63 Median + Mean + Mean (3x3)

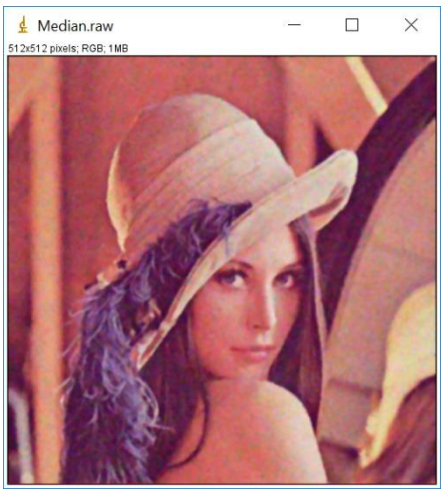


Figure 64 Median + Median + Mean (3x3)

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Discussion

- Looking at the PSNR values generated per channel in Table 1, we can see that they are all different and they are all affected differently by different filters. Thus, we can conclude that all channels have different noises of different intensities.
- Denoising is performed for individual channels by cascading one filter after another. We could perform different filtering for different channels by looking at how the PSNR value responds to different filters. But this can be laborious and can lead to uneven distortions.
- An ideal solution here would be to use median filter first to get rid of the impulse noise and then use a mean filter to get rid of the Gaussian noise.
- As discussed in the point above, Median Filter needs to be applied first. Impulse noise has the most extreme values and needs to be removed before we apply a mean filter otherwise it can cause spreading of the noise to neighbouring values.
- Looking at Figures 59-61, these are results of Median filter of increasing window sizes. We can see that as the window sizes increase, the image loses its edges and looks blurry and hazy at the edges. This is further supported by the PSNR values in the table above which decreased with increasing window size. Therefore, it is better to use smaller window sizes to retain accuracy.
- After trying various combinations of Median and Mean filters, looking at Table 1; I conclude that with these two filters, we can achieve the best PSNR values by first cascading two median filters then followed by a mean filter all of which have dimensions of 3x3. Adding another mean or median filter reduces the PSNR value and is not as effective.
- I have applied the same combination of filters in the same order to all three channels. This means that while applying a combination of filters to one channel, it adversely affects that of another. So this is one thing I would like to improve. This can be done by comparing the histogram of each channel to the original image to find out the type of noise so an appropriate filter can be applied accordingly. Another shortcoming and its improvement has been discussed in the point below.
- Another way to improve performance would be to use an adaptive median filter instead of a regular median filter. This is an improvement because in the current program a median filter is applied to the image regardless of checking if the pixel being corrected has impulse noise or not. In the Adaptive Median filter, median filter is applied based on if a certain condition is satisfied. This way only the impulse noise if targeted and the entire image will not look like it's been smoothened.[5]
- Another topic that we discussed in class that could be applied is the Structure Similarity which is a better at extracting structural information. Humans are only sensitive to certain types of distortion to the structure. SSIM is a method that helps us retain structural information in the image.

b) Principal Component Analysis (PCA) [6]

Abstract

This method utilizes Principal Component Analysis, which is a dimension reducing algorithm to apply denoising in an image. It was developed to overcome the time-consuming application of other filters where we apply operations pixel by pixel.

There are 3 main types of PCA:

- Patch based Global PCA (PGPCA)
- Patch based Hierarchical PCA (PHPCA)
- Patch based Local PCA (PLPCA)

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Approach

In PCA is a patch based method where we utilize patches extracted from the noisy image. The figure below is a pictographic representation of how patches and selected.

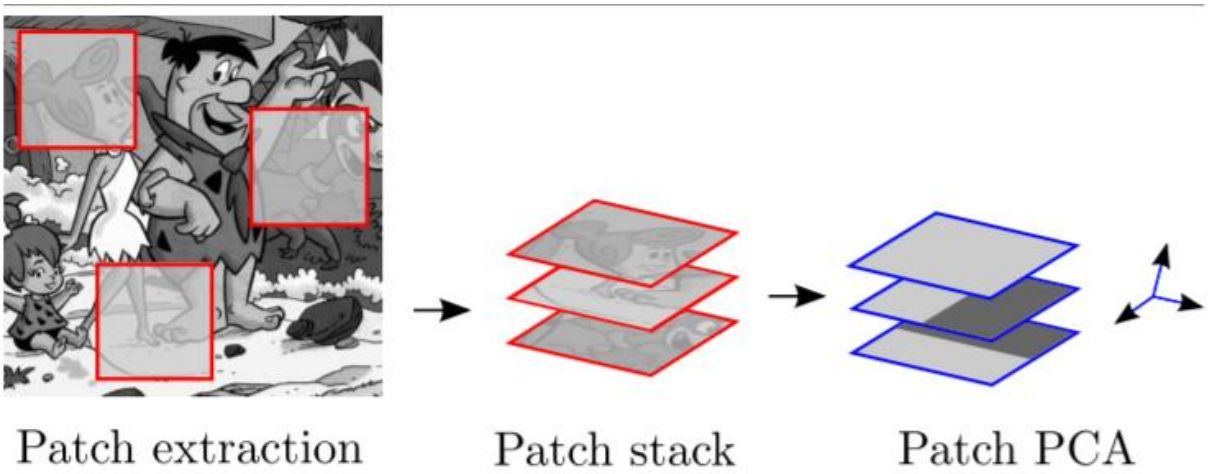


Image taken from http://imagine.enpc.fr/publications/papers/BMVC11.pdf

Patches are selected using a sliding window of NxN. Highly frequent patterns of the image and patch redundancy of natural images are used by PCA. In the global PCA rare patches which have limited contribution to the variance of the image are lost. In Local PCA we use subsets of patches, like inside small regions of the image or inside clusters. This way we can retain patches with less variance.

Experimental results

Table 2

WP	hW	factor_thr	PSNR
10	**10**	**2.75**	**31.04**
12	**12**	**2.75**	**31.08**
12	**12**	**4**	**30.66**
12	**12**	**3.5**	**31.13**
12	**12**	**3**	**31.32**
12	**12**	**2.5**	**31.21**
13	**13**	**3**	**31.31**

Where,
WP = Half size of patch
hW = half size of searching zone
factor_thr = factor in front of the variance term for hardthresholding the coefficients.

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

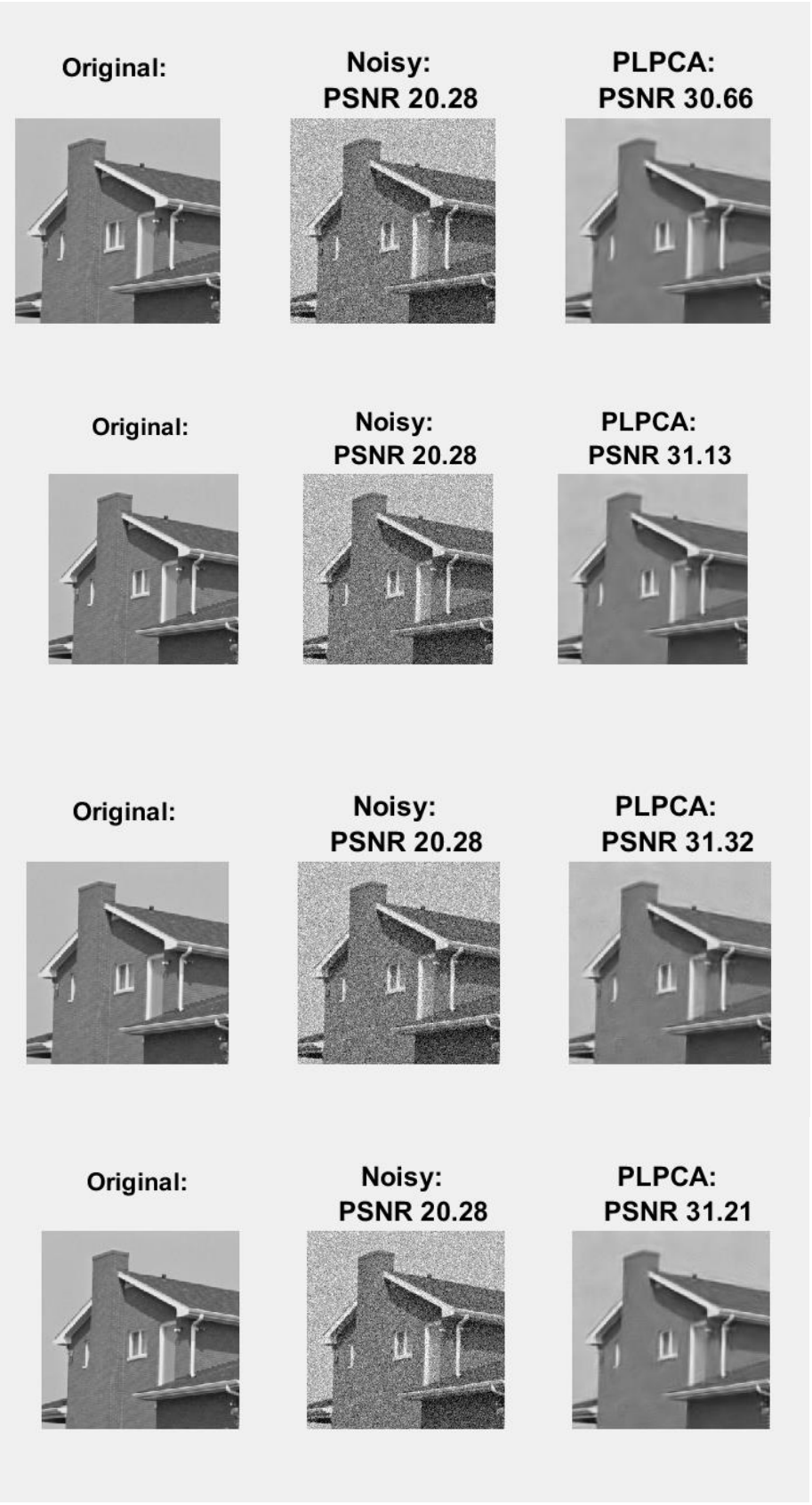


Figure 65 Denoising using PCA

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

Discussion

Components are chosen in decreasing order of variance. This way the most important, high frequency parts of the image are not lost either. Table 2 has been calculated for varying combinations of patch size, searching zone and thresholding factor. We can see that as the patch size and searching zone increases, to some extent the PSNR value also increases. But when thresholding factor increases PSNR decreases. The highest PSNR value was achieved at patch and searching zone half size of 12, thresholding factor of 3 giving a PSNR value of 31.32.
Repeating the method from Part A to House_noisy.raw, we get an output PSNR value of 27.9628 which significantly lesser than that of PCA. This is supported by the output image shown adjacent, generated using method A, which is substantially noisier than the PCA methods.
While we can reduce effort by using patching, computational complexity remains high as PCA needs to be applied repeatedly. Another drawback is that because we are using sliding windows, there is sometimes pixel overlapping which causes some pixels to be assigned multiple values.

c) Block matching and 3-D (BM3D) transform filter

Abstract

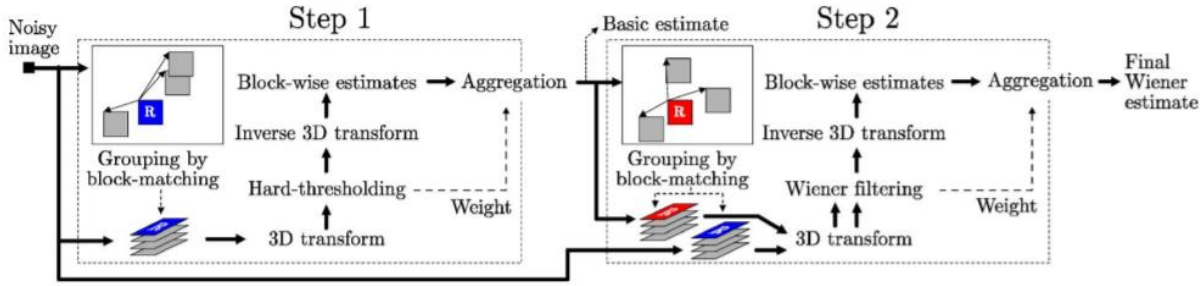


Figure 67 Process of BM3D [7]

This is the method where similar looking 2D image fragments into 3D arrays called “blocks”. This causes an improvement in sparsity and is derived from the concept of “enhanced sparse representation in transform-domain”. The collaborative filtering is achieved using the following 3 steps:

- 3D transformation of 2D group
- shrinkage of transform spectrum
- and inverse 3D transformation.

Using 3D blocks gives us the best 3D estimate of the joint 2D blocks. While this lets us identify the most common details shared amongst the images, it also lets us retain the finer details unique to each block. Due to the similarity in the groups, this transform function produced a sparse true signal in which the noise can be separated once shrinking is computed. Once the denoising is completed, each block is returned to its original location.

The grouping of the blocks can be done by one of various possible methods- K means clustering, fuzzy clustering, vector quantization etc. inverse of the distance is used to identify the similarity between blocks. Grouping is done by partitioning in such a way that each disjoint block only belongs to a single cluster. So we find the blocks that are similar to the reference one and stack them together to form the 3D block.

Approach

The steps followed to achieve BM3D are:

1. Group the blocks into a 3D array. Apply collaborative hard thresholding where a 3D transform is applied to the group and noise is attenuated by hard thresholding the coefficients. Inverse of the transform is used to produce an estimate for all the blocks which are returned to their original places. An estimate of the true image is obtained by using the weighted average of all the estimates of the blocks. This gives us a *basic estimate.*
2. Once the basic estimate is computed, we apply Wiener Filtering to find the *final estimate.* We again group similar looking blocks from the basic estimate. From the locations of these blocks two 3D arrays are formed- one for the basic estimate and one for the noisy estimate. After applying collaborative filtering to both the spectrums, we apply Wiener filtering to the noisy blocks using the basic estimate as the comparison for true energy spectrum. Estimates for all the blocks are again formed by performing inverse 3D transforms and the blocks are returned to their original locations. The true image estimate is again obtained from the weighted average of all the estimates.

Experimental Results

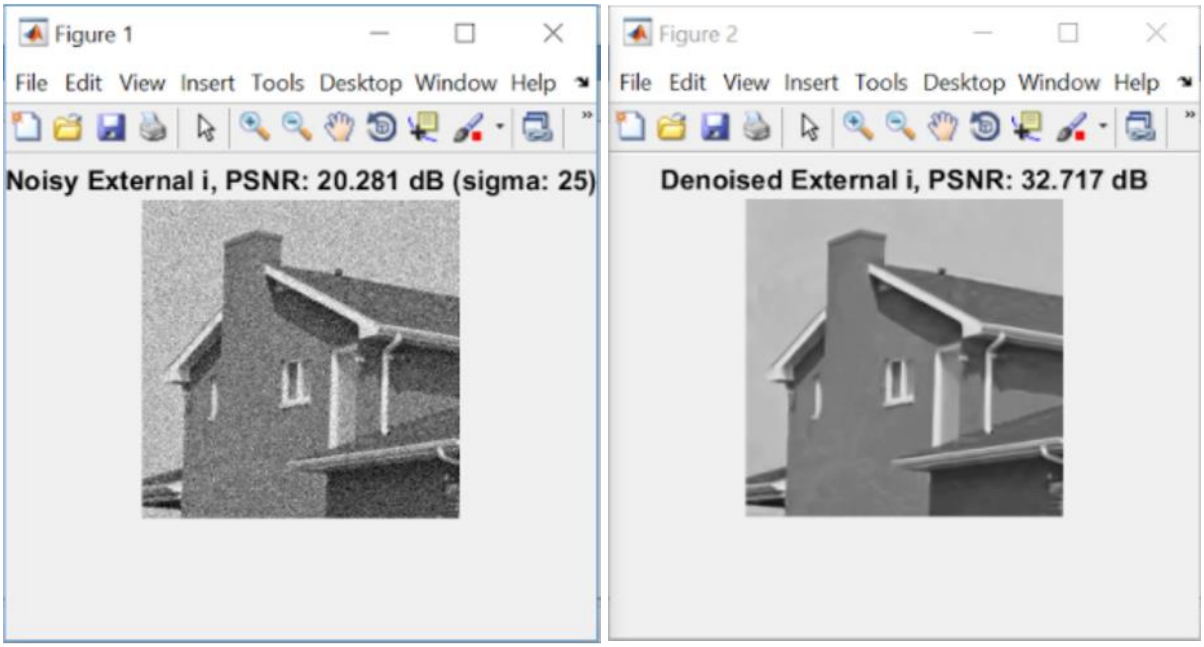


Figure 68 Standard output for BM3D

Ns=39
Tau_match=3000

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

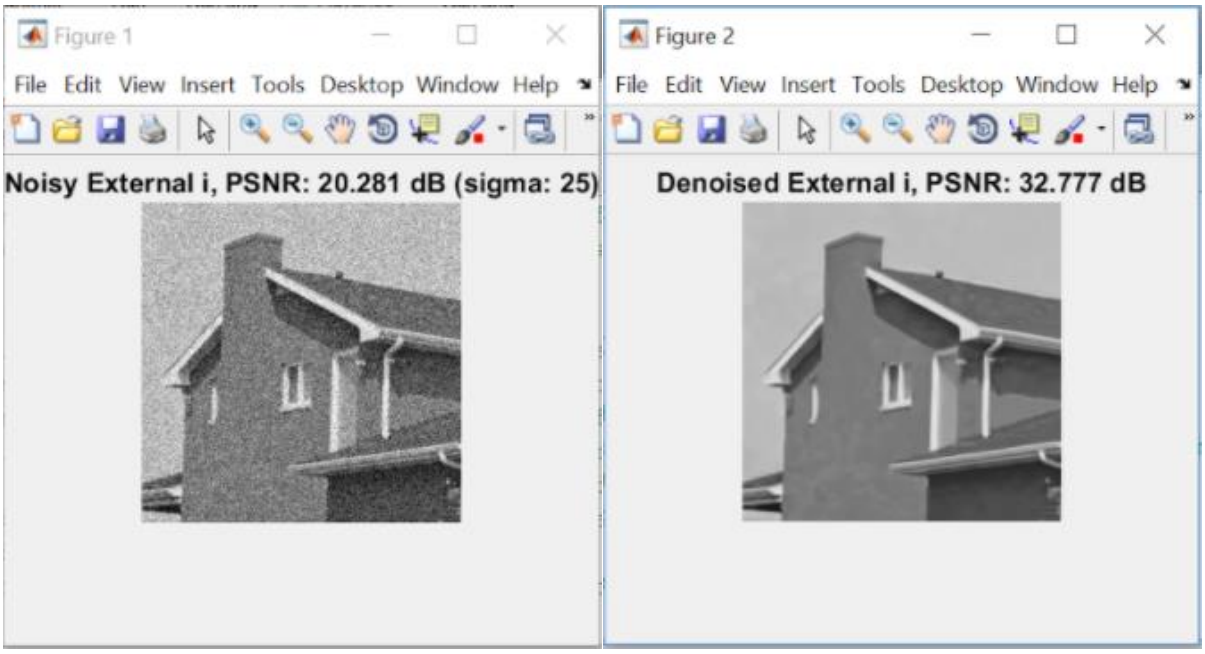


Figure 69 Improved output for BM3D

Ns=59
tau_match=2500

Discussion
Through trial and error, the two parameters whose variance gave the best results are Ns- length of the side of the search neighborhood for full-search block-matching (BM), tau_match – threshold for the block distance. Changing the other parameters did not produce any significant results. One observation was that an odd value for Ns produces better PSNR.

2) K means clustering involves partitioning the image into disjoint groups of high similarity. Partitioning causes unequal analysis of various components as the ones that are closest to the centroid give better responses than the ones further away. This is an occurrence even in equi-distributed fragments.

In block matching, we use a reference to compare it to each block and therefore this involves no partitioning. The groups are built on high similarity and each reference is the centroid for the whole group. This gives a fairer response than K means.

The Wiener method is used to find the final transform as using the basic estimates rather than the noisy ones produces much better results. Also using the basic estimate as the pilot signal for comparison is significantly more effective than hard thresholding the coefficients.

3)BM3D is a combination of the frequency domain and the spatial domain. The transform analysis and the use of the algorithm is performed in the spatial domain. But finding the similarity amongst the blocks and the Wiener filtering is performed in the frequency domain.

Apoorva Desai
5876264403
apoorvad@usc.edu
EE 569 HW1

4)As PCA uses a moving window, the overlapping values can produce redundancies. In BM3D each pixel or patch is part of a single cluster only. In PCA some of the low variance finer details of the image are last when performing dimensionality reduction. In BM3D all the final unique details of each block are still retained. PCA can require larger windows and is computationally heavier than BM3D.

References and citations

[1]https://web.archive.org/web/20160303201512/http://www.cis.rit.edu/mcsl/research/broadbent/CIE1931_RGB.pdf
[2] http://cadik.posvete.cz/color_to_gray_evaluation/
[3] https://www.atlantis-press.com/php/download_paper.php?id=4822
[4] http://www.tandfonline.com/doi/full/10.1080/09500340.2016.1163428?src=recsys&
[5] https://arxiv.org/ftp/arxiv/papers/1410/1410.2175.pdf
[6] http://imagine.enpc.fr/publications/papers/BMVC11.pdf
[7] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4271520
https://www.codementor.io/tips/7814203938/resize-an-image-with-bilinear-interpolation-without-imresize
http://josephsalmon.eu/code/index_codes.php?page=GPPCA
http://praveentutor.com/gray-scale-conversion-weighted-average-methods/
Wikipedia.org