Name: Deepika Kanade ***Student Id: 5369288614***

EE 569: Homework #1

Table of Contents:

Problem 1- Basic Image Manipulation……..……..2
a. Color Space Transformation……..2
i) Color to gray scale conversion…….2
ii) CMY(K) Color Space……..5
b. Image Resizing via Bilinear Interpolation……..……12

Problem 2- Histogram Equalization……..15
a. Histogram Equalization……..……15
b. Image filtering- Creating oil painting effect……..……..24
c. Image filtering- Creating film special effect……..….35

Problem 3- Noise Removal……..……45
a. Mix noise in color image…….....…45
b. Principal Component Analysis (PCA)……56
c. Block Matching and 3-D (BM3D) Transform filter…….....60

References……63

Problem 1: Basic Image Manipulation

a) Color Space Transformation

I. Color to Gray Scale conversion

1. Abstract and Motivation

Image processing is a part of our daily lives as we tend to process the image which our eye captures and extraction of important parts of the image is done by our brain. The main goal of this exercise is to familiarize us with the reading, writing and processing of the image.
The data read from any colored image consists of three components namely, R, G and B which contain 8 byte i.e. 256 bits of information each. The R, G, B channel are hence regarded to have gray scale value and the combination of these R, G, B channels results into a colored image of 24 bytes.
When the data is read from the raw image, it is in the form of R,G,B,R,G,B,R,G,B…. Hence, while reading the image, the data has to be separated distinctly into 3 components namely R,G,B.
Now coming back to the color to gray scale conversion, there are various methods employed to convert a colored 24-bit image into an 8-bit gray scale image consisting of 0 to 255 values. The black color is considered to have a value of 0 while the white color has a value 255. Every value in this range of 0 to 255 is considered to be gray.
The average method averages the values of R,G,B whereas the lightness method chooses the most and the least prominent colors. The luminosity method ,however, takes advantage of the way humans perceive images, as green is the most prominently recognizable color for a human eye. Hence, the green channel is given more weight compared to the R and B channels in the luminosity method.

2. Approaches and Procedure

Task: Convert the colored Tiffany image into a gray scale image by using average, luminosity and lightness method.

Theoretical Approach: The formulae for the three methods of conversion of colored image to gray scale image are given as follows:

a) Average method: (R+G+B)/3
b) Lightness method: (max(R, G, B)+min(R, G, B))/2
c) Luminosity method: 0.21R+0.72G+0.07B

(Source: EE569_2018Spring_hw1_v3.pdf)

Algorithm for color to gray scale Conversion:

Step 1: Read the input raw image ‘’Tiffany.raw’’ from a file into a 1D array whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array into separate 1D arrays for R, G, B channels respectively.
Step 3: Run a for loop across the image height*image width of the image

Step 3a: Apply the formulae for the average, luminosity and lightness method by finding the maximum and minimum values and the weights.

Step 4: Write the contents of the average, luminosity and lightness 1D arrays into a raw file for displaying.

3. Experimental Results

The three images obtained after carrying out the average, luminosity and lightness methods have been included in this section.

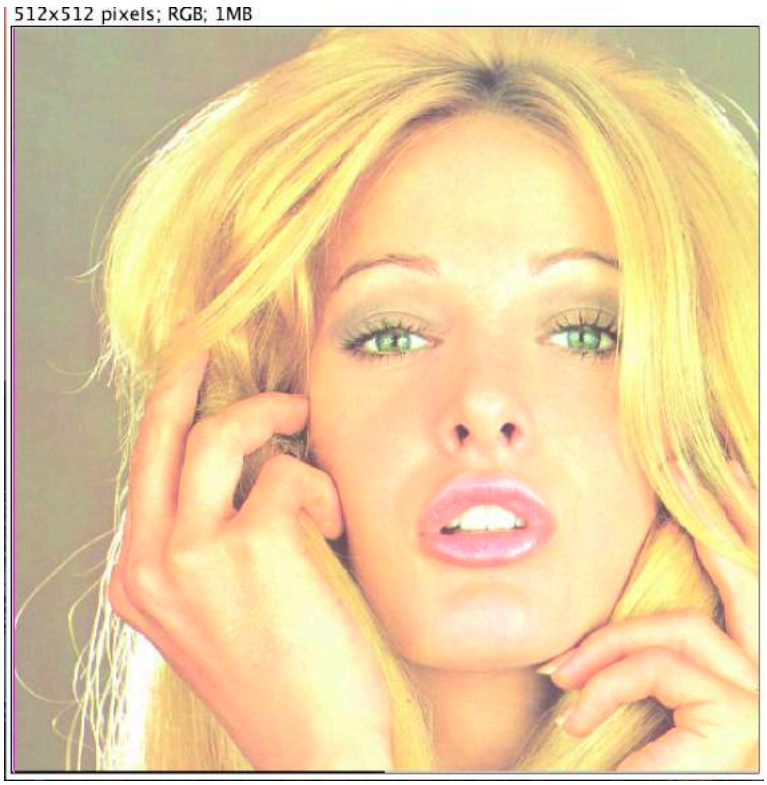


Figure 1: Original Image

Figure 2: Lightness method

Figure 3: Average Method

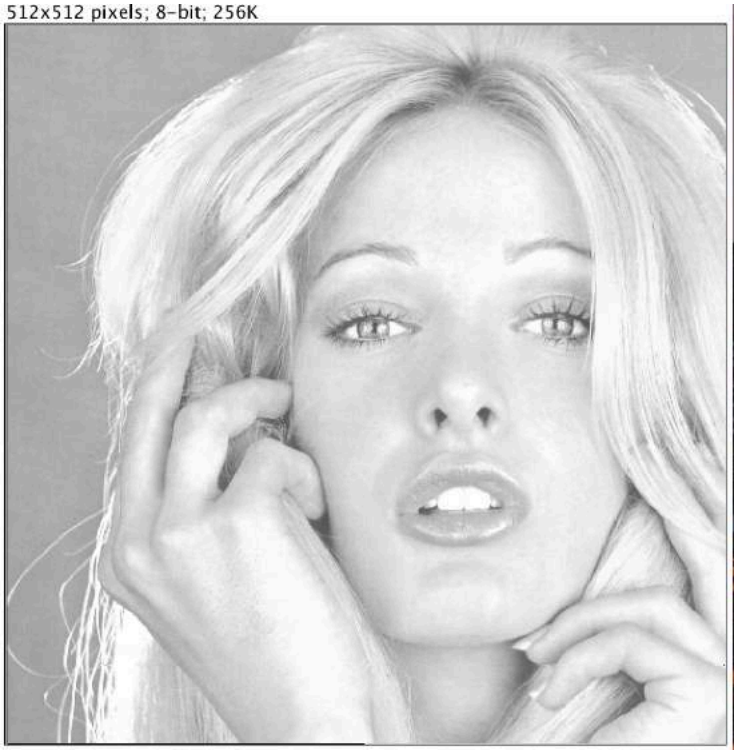


Figure 4: Luminosity Method

4. Discussion

The gray scale images after conversion using the three methods were studied and it can be deduced that the luminosity method works the best as the contrast in the gray scale image is maintained. The lightness and the average method tend to reduce the contrast, but the image looks smoother. However, the luminosity method is the best according to me as it assigns weights to each channel and the resultant image is soothing to the eye.

II. CMY(K) Color Space

1. Abstract and Motivation

There are various color spaces that are used in image processing namely CMY(K), RGB, HSL, YUV and so on. The CMY uses the concept of subtractive mixing of colors and is mainly used in printing. Generally, the paper used for printing is white which is why the ink used for printing should absorb the color rather than reflecting it. Hence, black ink is used mainly for printing as it is a good absorbent of light.
The CMY model can be obtained from the RGB model as follows:

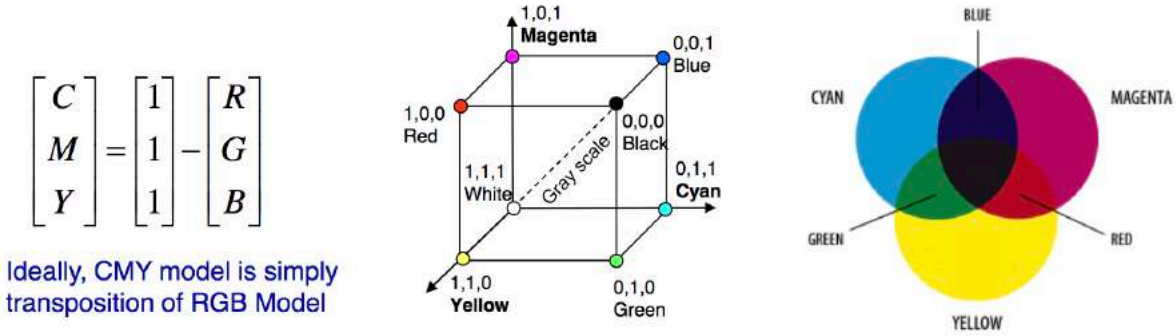


Figure 5: CMY Color Representation (Source: EE 569 Lecture Slide)

2. Approach and Procedure

Task: Obtaining the CMYK color space from the RGB color space.

Algorithm for Conversion of RGB to CMY: -

Step 1: Read the input raw images ‘’Bear.raw’’ and “Dance.raw” from a file into a 1D array whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array into separate 1D arrays for R, G, B channels respectively.
Step 3: Run a for loop across the image height*image width of the image
Step 3a: Calculate the C,M,Y components of the image and store those in a 1D array.
Step 4: Combine the C, M,Y components in a 1D array which has the following structure C,M,Y,C,M,Y,C,M,Y…..
Step 5: Write the contents of the combined CMY array into a raw file for displaying.

3. Experimental Results

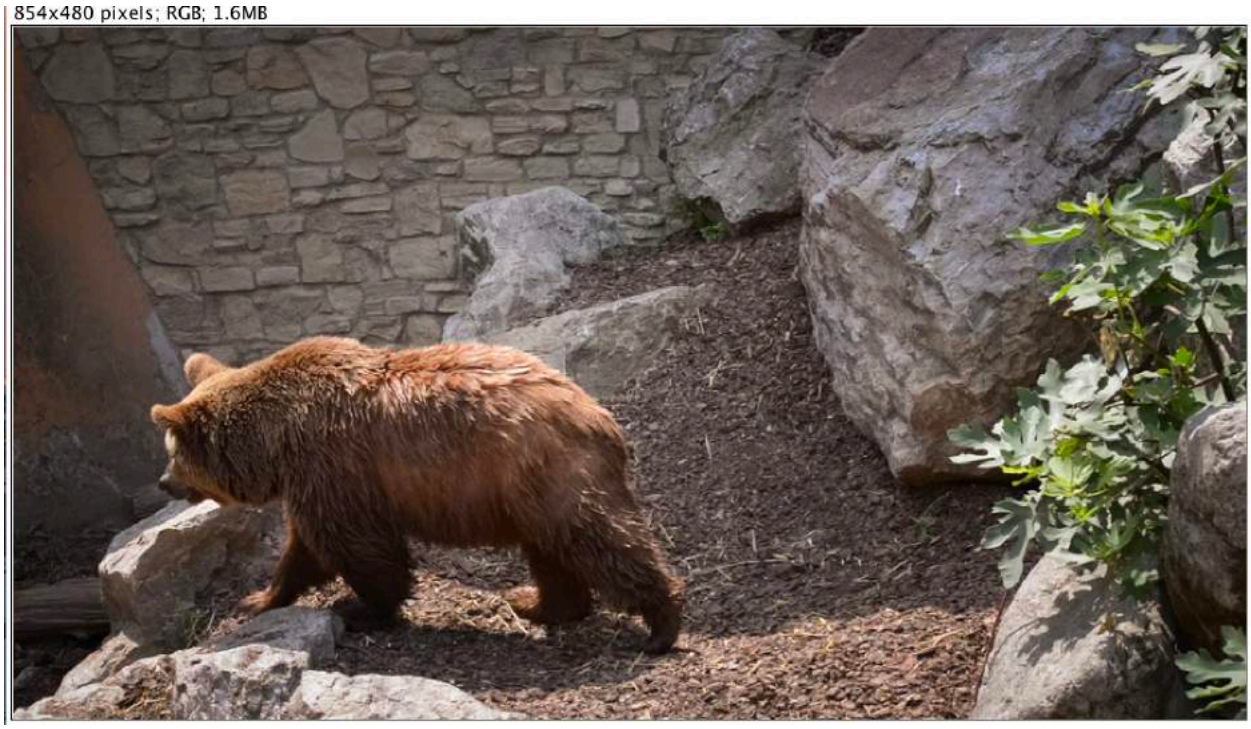


Figure 6: Original Bear Image

Figure 7: CMY representation of Bear image

Figure 8: Cyan(C) component of Bear image

Figure 9: Magenta(M) component of Bear image

Figure 10: Yellow(Y) component of Bear image

Figure 11: Original Dance Image

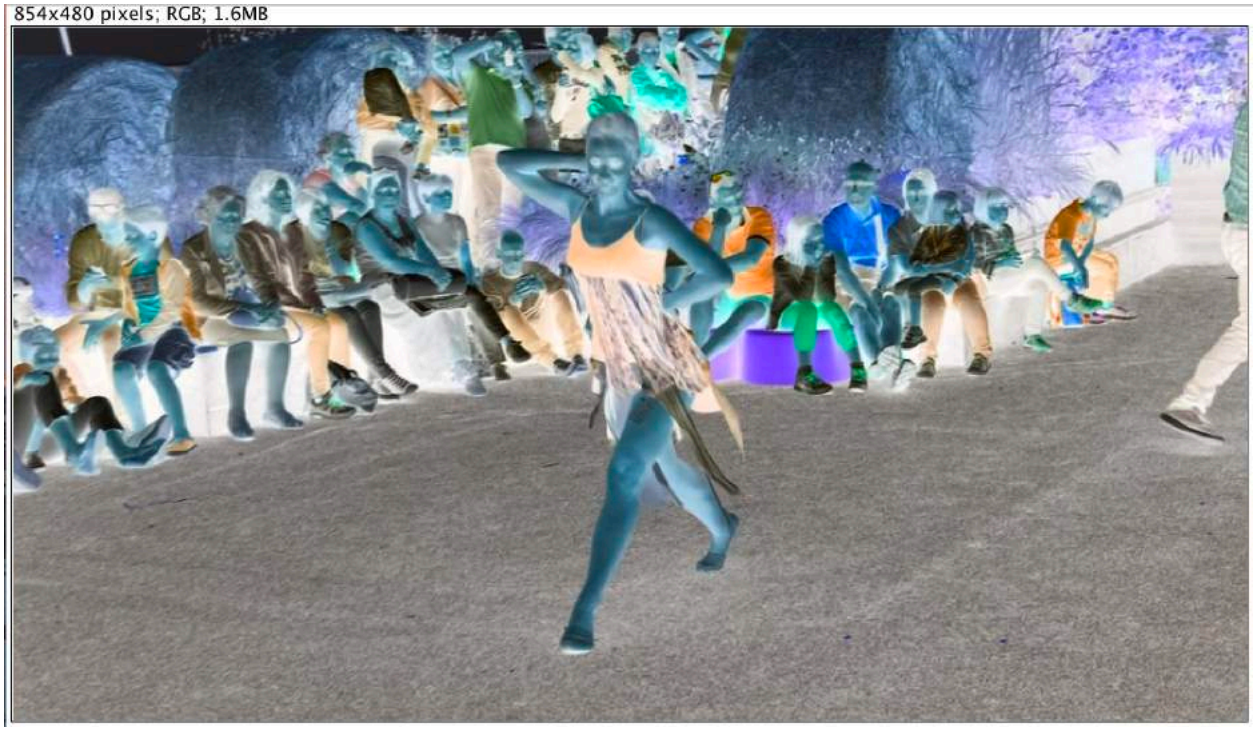


Figure 12: CMY Dance Image

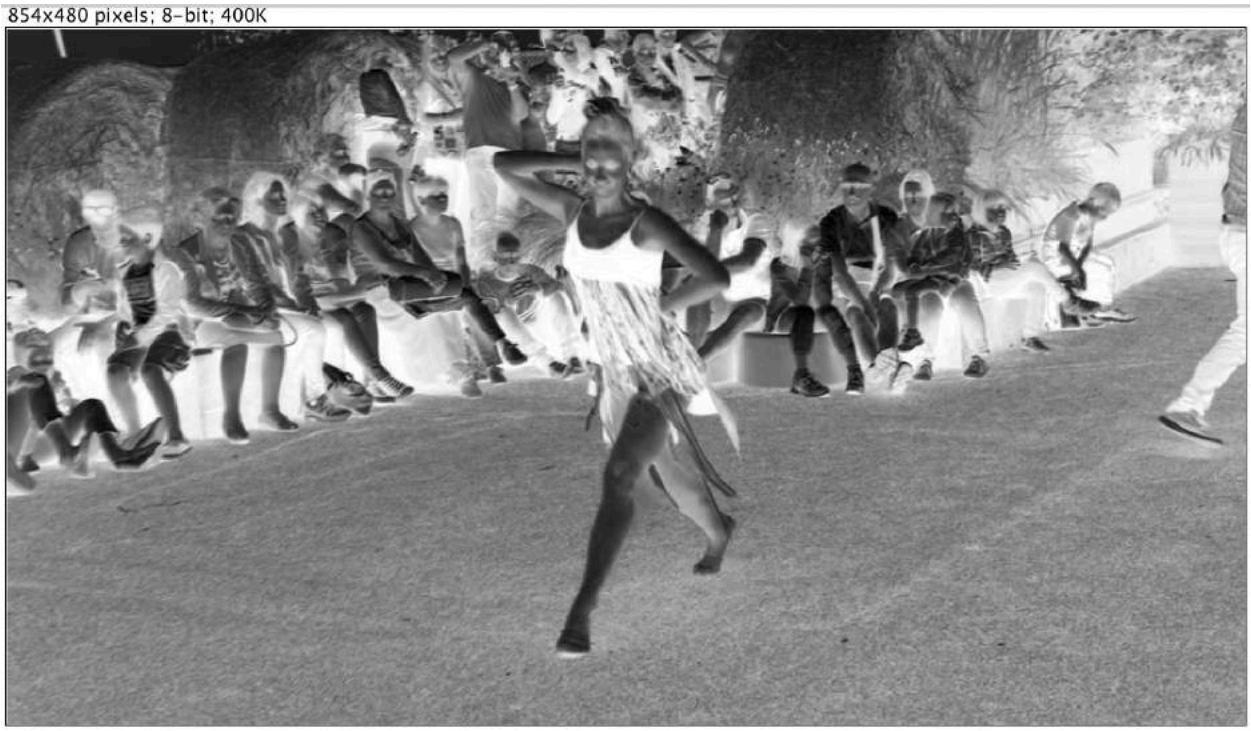


Figure 13: Cyan(C) Component of Dance Image

Figure 14: Magenta(M) Component of Dance Image

Figure 15: Yellow(Y) Component of Dance Image

4. Discussion

From the images, it can be seen that in the CMY(K) representation, the entire image is made of Cyan, Magenta, Yellow and Black colors.

b) Image Resizing via Bilinear Interpolation

1. Abstract and Motivation

The goal of this exercise is to familiarize us with the concept of image resizing as it is widely used in zooming in and zooming out images. Image resizing is nothing but increasing or decreasing the size of a particular image. However, while doing this, the new co-ordinates have to mapped from the old co-ordinates which can be done using the concept of bilinear interpolation.

Task: Resizing(Upsizing) the input Airplane image of size 512*512 into an output image of size 650*650 using the concept of Bilinear Interpolation.

2. Approach and Procedure

Resampling Through Bilinear Interpolation

Let $\mathbf{I}$ be an $R \times C$ image.
We want to resize $\mathbf{I}$ to $R' \times C'$.
Call the new image $\mathbf{J}$.

Let $s_R = R / R'$ and $s_C = C / C'$.

Let $r_f = r' \cdot s_R$ for $r' = 1, \ldots, R'$

and $c_f = c' \cdot s_C$ for $c' = 1, \ldots, C'$.

Let $r = \lfloor r_f \rfloor$ and $c = \lfloor c_f \rfloor$.

Let $\Delta r = r_f - r$ and $\Delta c = c_f - c$.

$$\begin{aligned}\text{Then } \mathbf{J}(r',c') = {} & \mathbf{I}(r,c)\cdot(1-\Delta r)\cdot(1-\Delta c) \\ & + \mathbf{I}(r+1,c)\cdot\Delta r\cdot(1-\Delta c) \\ & + \mathbf{I}(r,c+1)\cdot(1-\Delta r)\cdot\Delta c \\ & + \mathbf{I}(r+1,c+1)\cdot\Delta r\cdot\Delta c.\end{aligned}$$

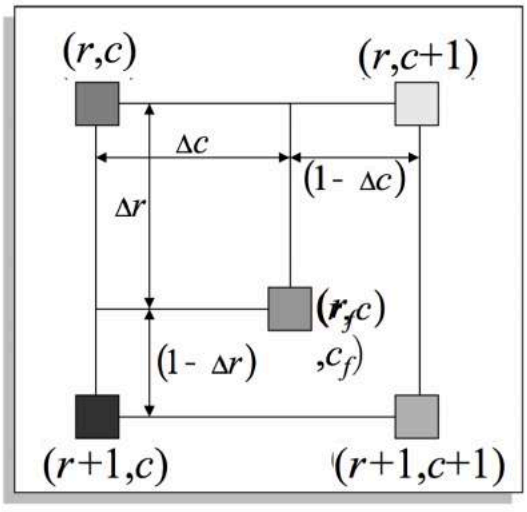


Figure 16: Resampling using Bilinear Interpolation algorithm

(Source: https://ia802707.us.archive.org/23/items/Lectures_on_Image_Processing/EECE_4353_15_Resampling.pdf)

Theoretical Approach: The output image of size 650*650 is constructed from the input image of size 512*512 by considering four adjacent pixels of input image for every pixel of output image.

As seen in the figure 16, the input image I has size 512*512 and the output image J has size 650*650. SR and SC are the resizing ratios with respect to the original and new heights and widths of the image. Rf and Cf are the mapped co-ordinates in the input image to attain a size of R' and C' in the output image. deltaR and deltaC are used to obtain the four neighboring pixels in the input image whose values are used to recreate one output image pixel.

Algorithm for Image Resizing:

Step 1: Read the input raw image ''Airplane.raw'' from a file into a 1D array whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array into separate 1D arrays for R, G, B channels respectively.
Step 3: Run two for loops across the new image height and new image width of the image respectively
- Step 3a: Calculate the resizing ratio and find the mapped pixel (Rf,Cf) by using the formula described in Figure 16.
- Step 3b: Calculate the floored value for each value of Rf and Cf.
- Step 3c: Check if the mapped pixel index exceeds the height and width of the original image.
- Step 3d: Calculate delx and dely which will be used to obtain the four adjacent pixels of the input image to be used for Bilinear Interpolation.
- Step 3e: Find the co-ordinates of J(R',C') of R,G,B channels separately by using the formula mentioned in the Figure 16. This is the step where the concept of Bilinear Interpolation is used.

Step 4: Combine the R,G,B arrays of the resized image into a 1D array which will then be written to a file to be displayed as an output image.

3. Experimental Results

Figure 17: Input Airplane Image of size 512*512

Figure 18: Output Airplane image of size 650*650

4. Discussion

The Bilinear Interpolation is a good algorithm to resize an image, however, it has some drawbacks. If the resizing ratio is increased to a large extent, the image starts to become blur which is an unwanted thing. Hence, other algorithms like Nearest Neighbor Interpolation, Box Sampling can be used to reduce the blurriness of the image. In the box sampling method, a considerable block of input pixels is taken to estimate the value of the output pixel which leads to a large number of input pixels contributing to the output. Hence, the loss of data is less in the Box sampling method. Therefore, such Adaptive Interpolation methods should be used if the resizing has to be done with less loss of data.

Problem 2: Histogram Equalization

a) Histogram Equalization

1. Abstract and motivation

Histogram Equalization becomes a very important concept in image processing as it is used to improve the contrast of the images. The low contrast images are converted into high contrast images by using the histogram Equalization technique.
This method is specifically used in areas wherein the capturing of images can give you dark values. For instance, in areas where X-rays are acquired, the contrast of the x-ray can be low which is very difficult for the eye to recognize. Hence, histogram Equalization becomes an important concept in such cases.
Generally, in a low contrast image, a lot of pixels have their intensity values in the darker range i.e. closer to 0. Therefore, when equalization is done, the number of pixels having their value close to 0 is reduced, whereas the number of pixels having their values close to 255 is increased. This improves the contrast.
In the histogram Equalization techniques, the original intensity of the pixels is manipulated which is reflected in the output image.
Two methods are used for histogram Equalization:
1)Transfer Function based Histogram Equalization
2)Cumulative probability-based Histogram Equalization
A brief discussion of both the methods is given below:

- **Transfer Function Based Histogram Equalization**

In a Transfer Function based Histogram equalization method, the transfer function is generated in a such a way that a lot of pixels in the original image having darker values are mapped to higher intensities. This method uses the concept of finding probability of the pixels and the cumulative distribution function of those pixel intensities.

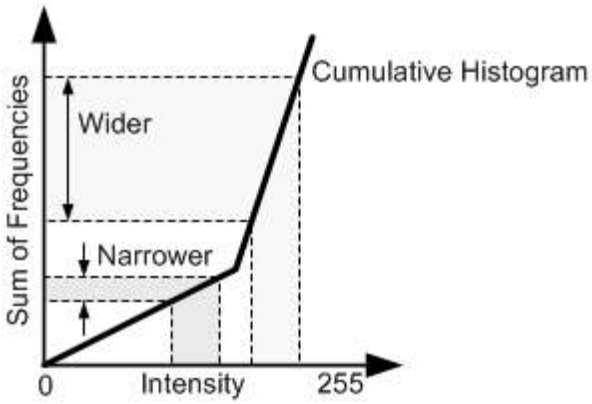


Figure 19: Concept of histogram Equalization in the Transfer function-based method

For instance, in figure 19, the transfer function of the input vs output image is divided into two sections. When the slope of the line is approximately equal to one, the range of intensity values of input image is mapped to a narrow band. But, when the slop of the line is increased above one, a small portion of input image pixel intensities are matched to a larger portion of output intensity values.
The main purpose of using this method is we are going to shrink the range of output image pixel intensities where the pixel density is high and expand the range of intensity values in output image where pixel density of input image is low.

- **Cumulative probability-based Histogram Equalization**

In this type of histogram Equalization, equal number of pixels are assigned a specific intensity. Hence, in this method the entire range of pixel intensities i.e. 0 to 255 is used and hence the equalization method is quite effective.
This method is a good way of improving the contrast as every pixel intensity has equal weight and hence all the intensity values are present in the final image.

2. Approach and Procedure

Task: To equalize the input Desk image by using the transfer function and the cumulative histogram methods and to study the equalized histograms for each method.

Algorithm for Transfer Function based histogram Equalization:

Step 1: Read the input raw image ‘Desk.raw’’ from a file into a 1D array whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array into separate 1D arrays for R, G, B channels respectively.

Step 3: Find the count of each intensity value for the R,G and B channel separately.
Step 4: Find the probability for each intensity value in the R,G,B channel. This is done by dividing each intensity by the total number of pixels in the image.
Step 5: Calculate the cdf values for each channel and multiply the obtained cdf values by 255 to expand the range of intensity values. Then floor the values to the nearest integer.
Step 6: Replace the floored intensity values into the original R channel, G channel, B channel arrays.
Step 7: Combine the R,G,B arrays having the mapped intensity values into a 1D array which will then be written to a file to be displayed as an output image.
Step 8: Move the values of each intensity count for R,G,B channels into a Text file which will be called from Matlab to plot the histograms and the transfer functions.

Algorithm for Cumulative probability-based Histogram Equalization:

Step 1: Read the input raw image 'Desk.raw'' from a file into a 1D array whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array into separate 1D arrays for R, G, B channels respectively.
Step 3: Find the count of each intensity value for the R, G and B channel separately and store the location where that intensity value was present in the R,G,B channels.
Step 4: Calculate the bucket size which is the total number of pixels divided by the number of possible intensities.
Step 5: Traverse till the bucket size for every channel and replace the intensity for every pixel in the bucket with the same intensity.
Step 6: Replace the new intensity values on the locations stored in Step 3 for all the R, G, B channels.
Step 7: Combine the R, G, B arrays having the mapped intensity values into a 1D array which will then be written to a file to be displayed as an output image.
Step 8: Calculate the cumulative values of the pixels for each intensity and print these values into a text file. Read this file from Matlab to plot the Cumulative histograms for each channel.

3. Experimental Results

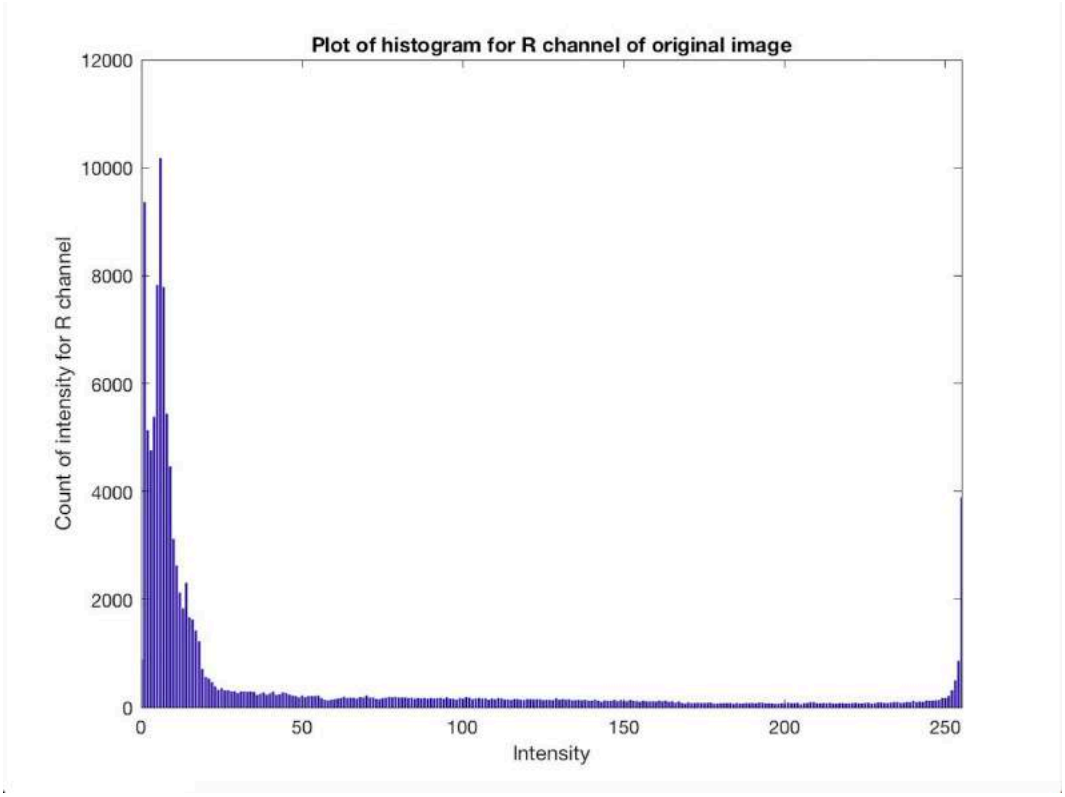


Figure 20: Plot for Histogram of R channel of original image

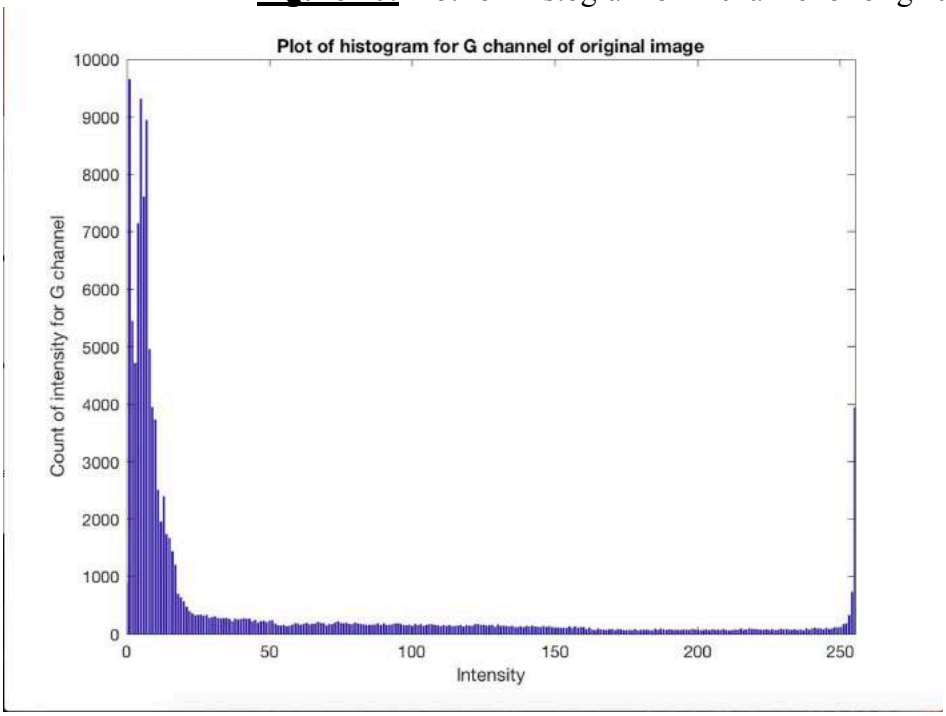


Figure 21: Plot for Histogram of G channel of original image

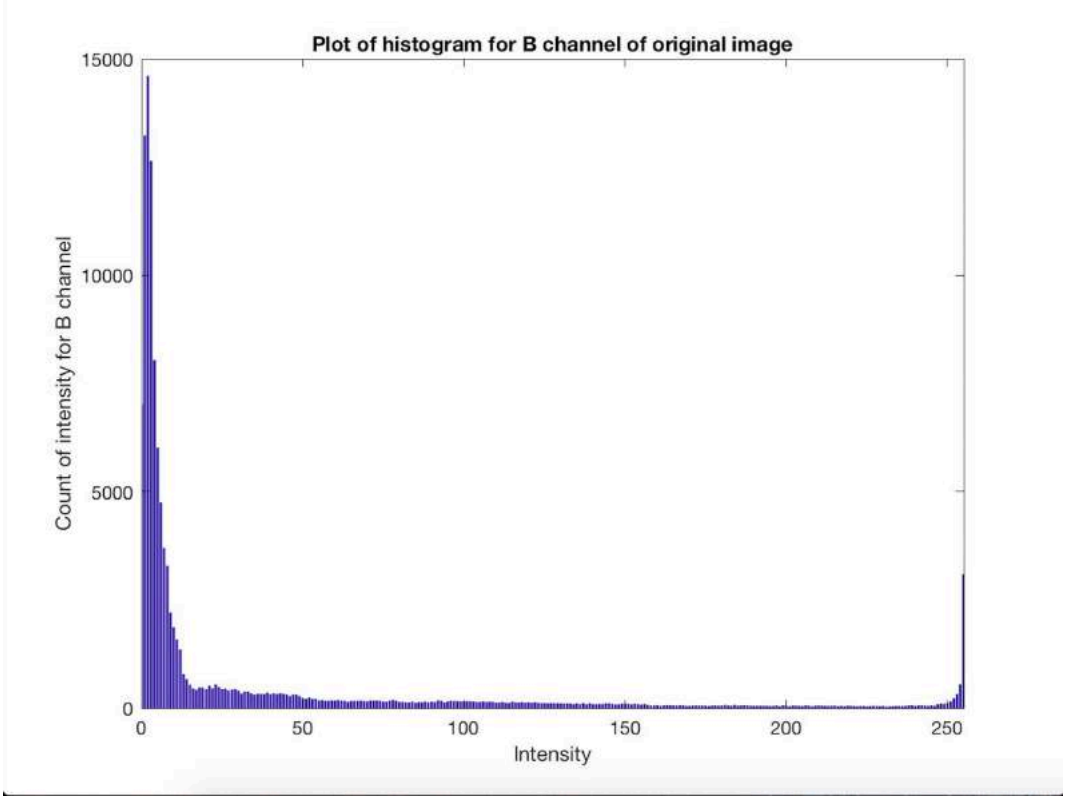


Figure 22: Plot for Histogram of B channel of original image

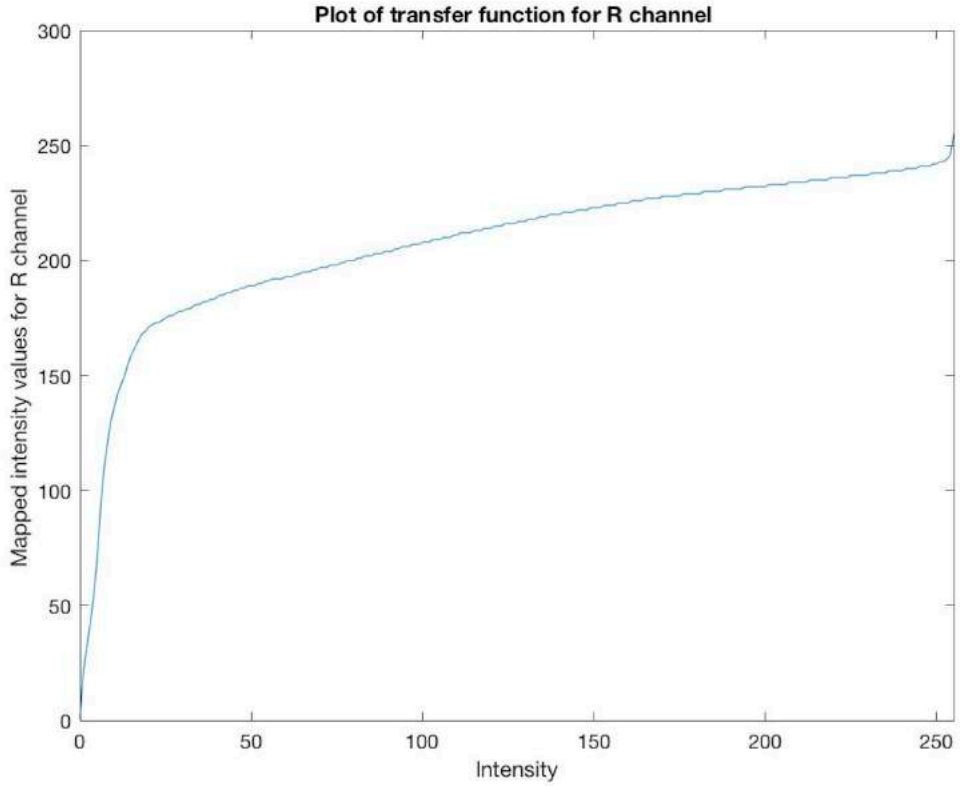


Figure 23: Transfer Function for R channel (Method A)

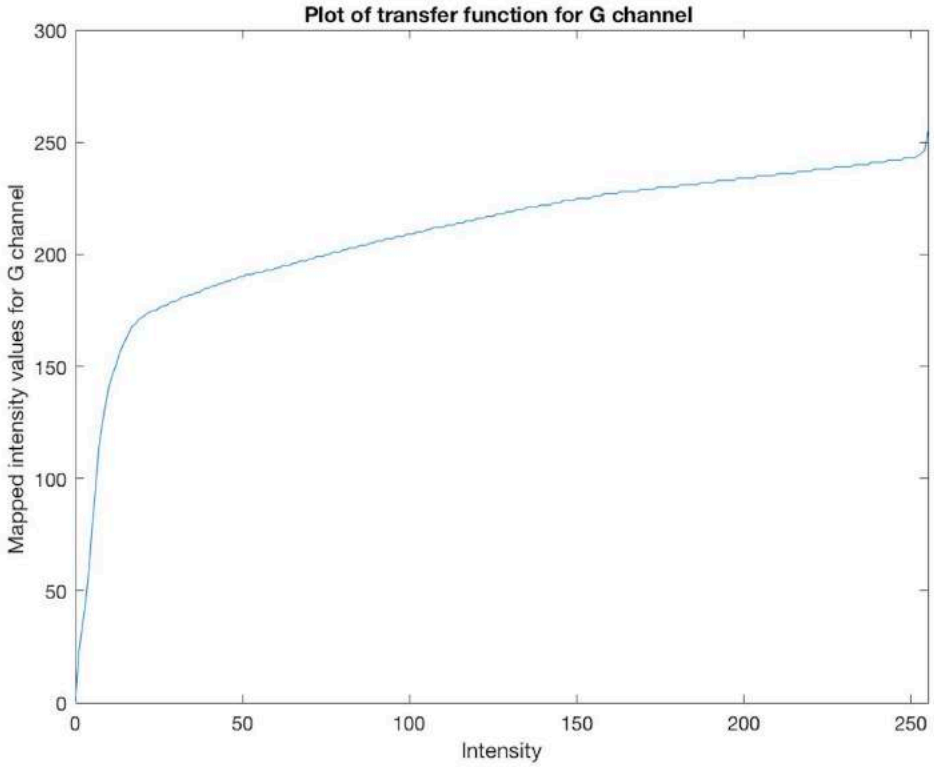


Figure 24: Transfer Function for G channel (Method A)

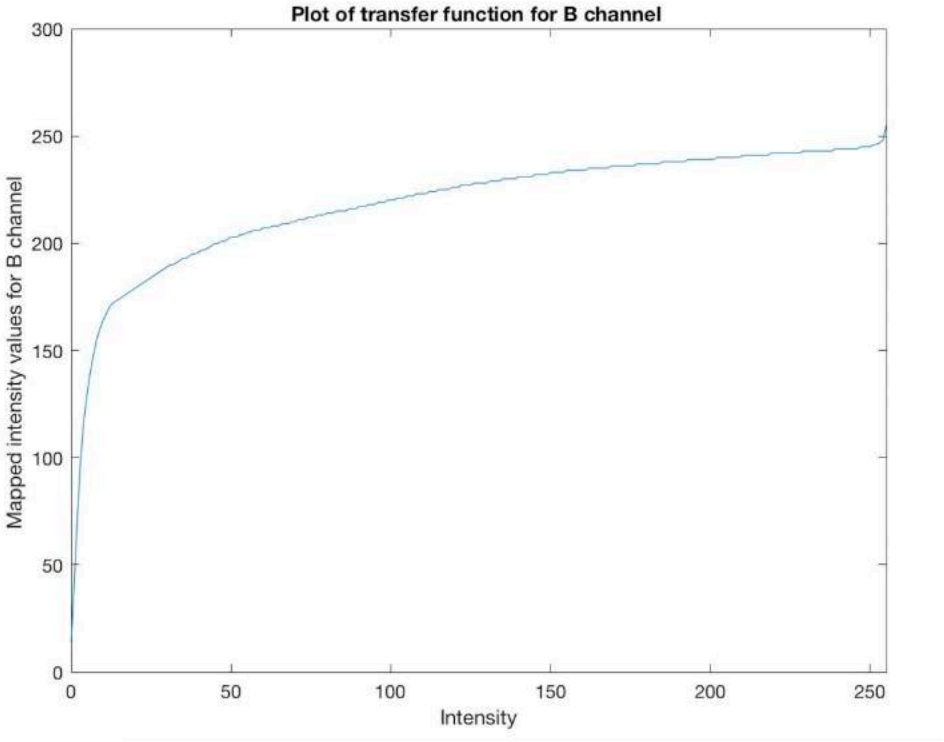


Figure 25: Transfer Function for B channel (Method A)

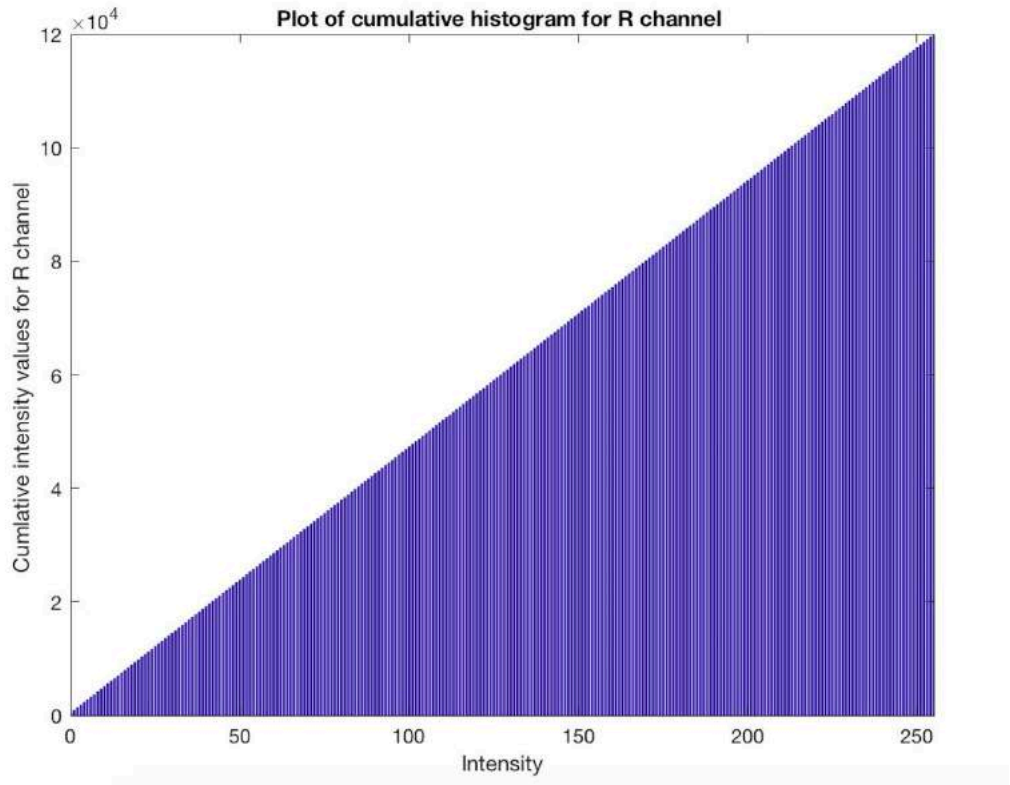


Figure 26: Cumulative Histogram for R channel (Method B)

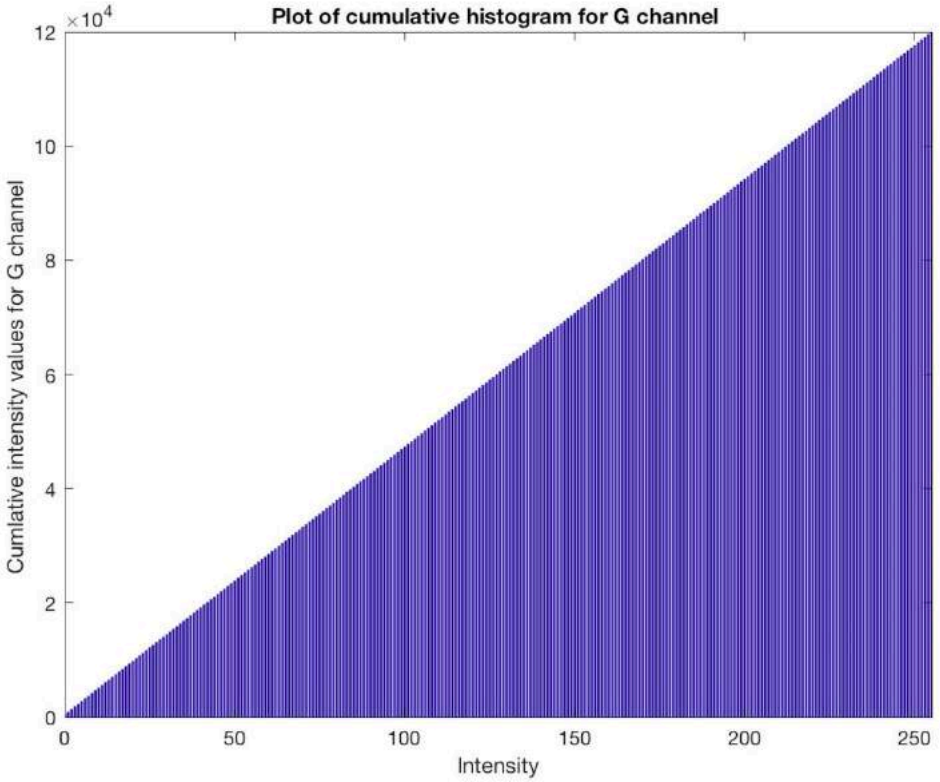


Figure 27: Cumulative Histogram for G channel (Method B)

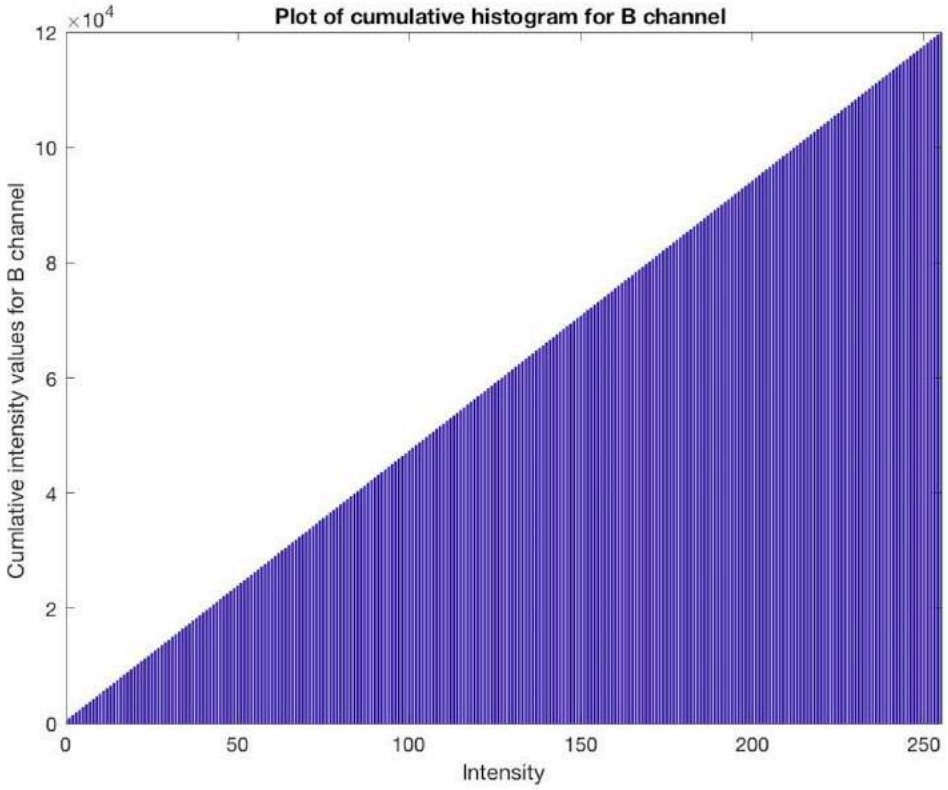


Figure 28: Cumulative Histogram for B channel (Method B)

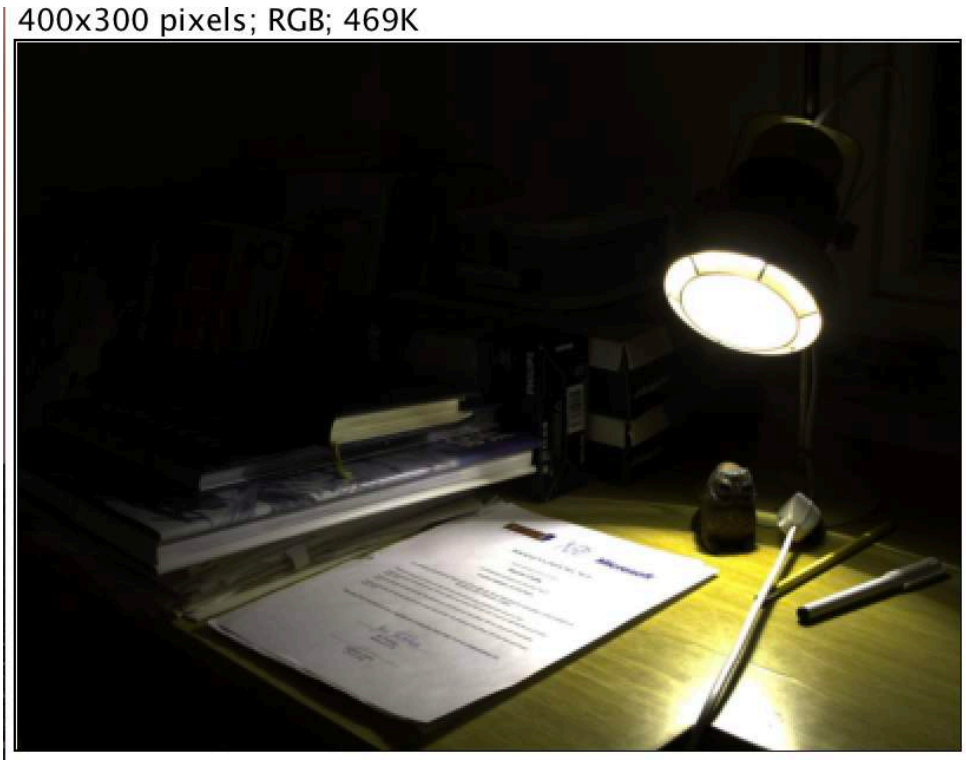


Figure 29: Desk Original Image

Figure 30: Transfer Function Based Histogram Equalized image (Method A)

Figure 31: Cumulative Probability Based Histogram Equalized image (Method B)

4. Discussion

As is visible from the Figures 20,21 and 22, a lot of pixels of the R,G,B channel are concentrated in the lower intensities. That is, a large number of pixels have intensity values closer to 0. Hence, the original desk image has darker portions which has to be removed by using histogram Equalization.
When we look at the figures 23,24 and 25, it can be observed that a larger portion of the intensities of the input image is mapped to the brighter intensity values i.e. close to 255 of the output image. This helps improve the contrast of the output image as a lot of bright portions are included in the image.
However, when we observe the figures 26,27 and 28, we get the transfer function having a slope of 1 as the intensities are distributed uniformly in the entire image. Hence each intensity value has equal number of pixels in it. This helps in contrast improvement of the image as the brighter intensity values also come into existence in the output image.
When we compare figures 30 and 31 which are the equalized images to figure 29 which is the input image, we can observe that the dark portion of the image has been illuminated and all the colors are visible in the output.
However, it can be observed that the enhanced images have some noise which is present in the top left corners which can be removed by using Adaptive Histogram Equalization techniques. When the image into consideration has lot of darker or brighter portions, the adaptive histogram equalization method works the best as it uses transformation function which is derived by taking neighboring pixels into consideration. Hence, the histograms are plotted for distinct sections of the image which are later combined to distribute the lightness in the image.

b) Image Filtering- Creating oil painting effect

1. Abstract and Motivation

Oil painting effect is used to decrease the number of colors in a color palette. Generally, the color palette consists of 256 colors but in the oil painted version of the image, only 64 colors are present. Each of the R, G, B channels are represented using only 4 colors respectively. This effect is obtained by choosing the most frequent color in the neighborhood of a pixel and assigning that value to the all the pixel in that bin. The oil painting method results into a blocky image with less number of colors than the original image and the resultant image looks like an oil painting.

2. Approach and procedure

Task: To get the 64-bit quantized version of the “Star Wars” and “Trojans image” and to assign the most frequent color in a particular pixel’s neighborhood to get the oil painted effect.

Algorithm to generate 64-bit version of the image:

Step 1: Read the input raw image 'Star_Wars.raw'' and "Trojans_720x480" from a file into a 1D array whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array into separate 1D arrays for R, G, B channels respectively.
Step 3: Find the count of each intensity value for the R,G and B channel separately and store the location where that intensity value was present in the R,G,B channels.
Step 4: Calculate the bin size which is the total number of pixels divided by the number of colors to be depicted in the image for only a channel.
Step 5: Traverse till the bin size for every channel and replace the pixel intensity for every pixel in the bucket by the mean of the pixel intensities for that bucket.
Step 6: Replace the new intensity values on the locations stored in Step 3 for all the R,G,B channels.
Step 7: Convert the 1D array having the changed intensities for the R,G,B channel into 2D arrays.
Step 8: Run two nested for loops till the image height and image width of the image
    Step 8a: Run two nested loops to traverse the mask height and width which changes from N=3 to 11
        Step 8a.1: Check if the mask indexes exceed the image height and width of the image.
        Step 8a.1: Store the values of intensities for the masks and keep of count of the number of intensities in every mask. Now find the most frequent intensity value in that particular mask.
        Step 8a.3: Assign the most frequent value calculated in step 8a.1 for the pixels in that mask for R, G, B channels separately.
Step 9: Combine the R, G, B arrays having the changed intensity values into a 1D array which will then be written to a file to be displayed as an output image.

3. Experimental Results

Figure 32: Star Wars original image

Figure 33: Star Wars 64-bit representation

Figure 34: Star Wars N=3 filter

Figure 35: Star Wars N=5 filter

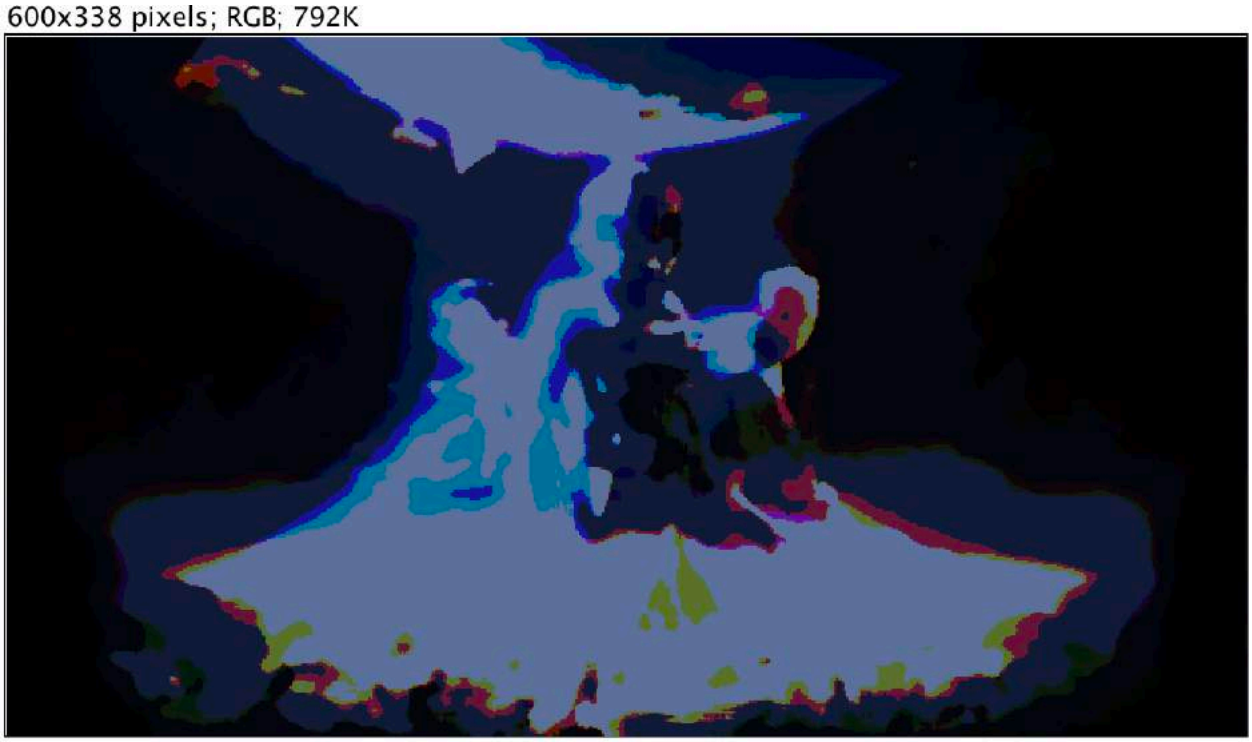


Figure 36: Star Wars N=7 filter

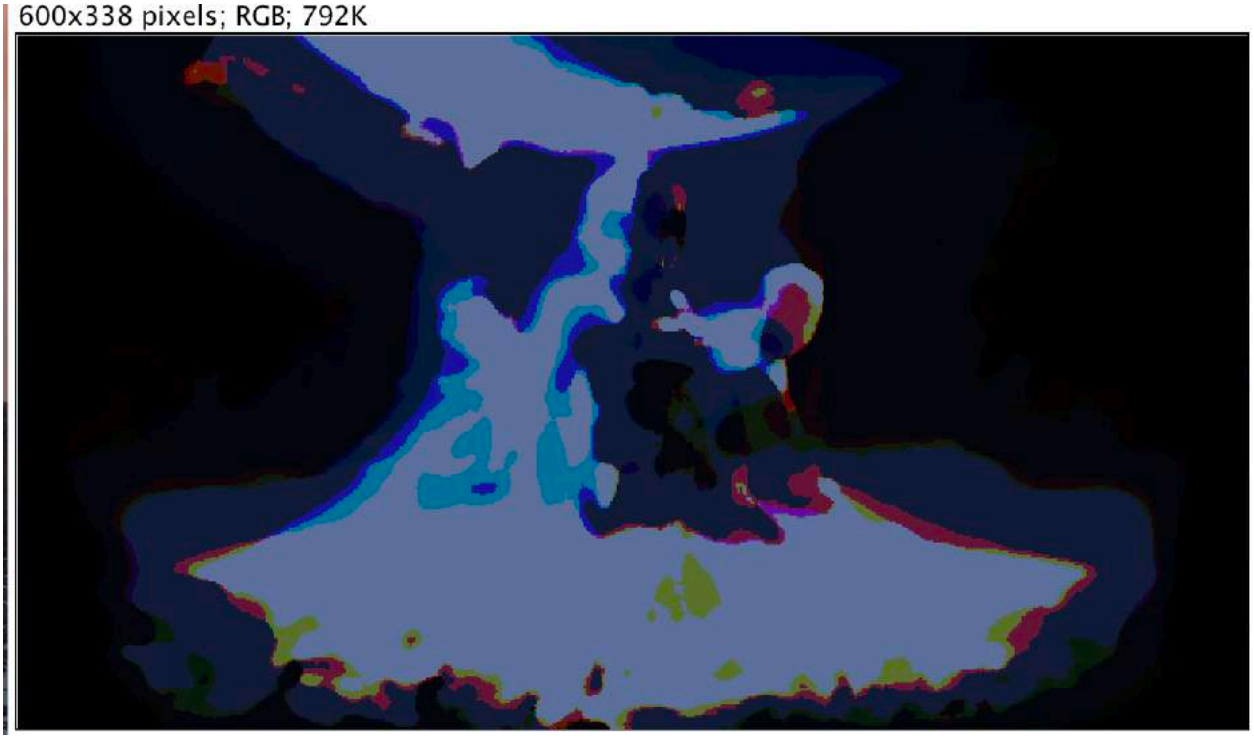


Figure 37: Star Wars N=9 filter

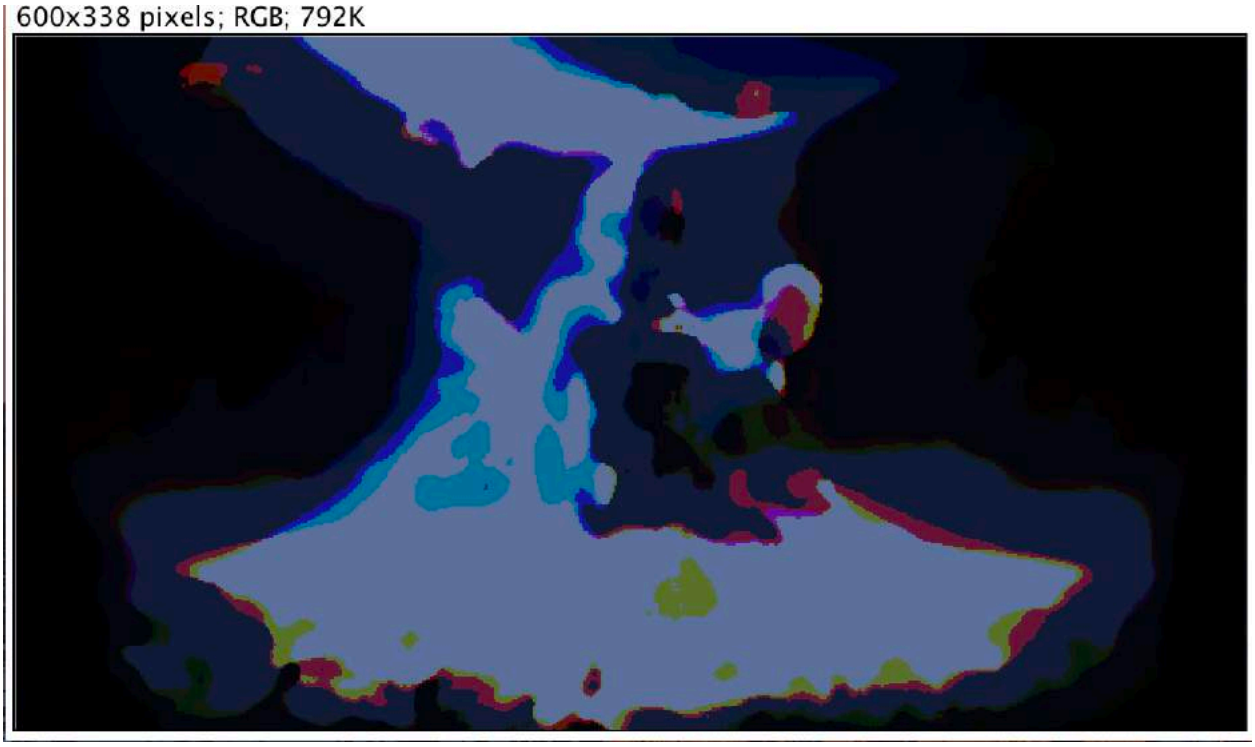


Figure 38: Star Wars N=11 filter

Figure 39: Star Wars 512-bit representation for N=3 filter

Figure 40: Star Wars 512-bit representation for N=5 filter

Figure 41: Trojans Original Image

720x480 pixels; RGB; 1.3MB

Figure 42: Trojans 64-bit Representation

720x480 pixels; RGB; 1.3MB

Figure 43: Trojans Image N=3 filter

Figure 44: Trojans Image N=5 filter

Figure 45: Trojans Image N=7 filter

Figure 46: Trojans Image N=9 filter

Figure 47: Trojans Image N=11 filter

Figure 48: Trojans Image 512-bit representation for N=3 filter

Figure 49: Trojans Image 512-bit representation for N=5 filter

4. Discussion

For the 64-bit representation of the image, each channel has only 4 colors within it. The threshold chosen for the 64-bit representation is the average intensity value within that particular bin. An average of the intensity values in a particular bin is assigned to every pixel of that bin. Hence, only 4 colors are used for each R, G, B channel. The 64-bit representation of both the images can be seen in Figures 33 and 42 and it can be inferred that only certain colors are present in these images.
The oil painting algorithm is implemented for a variety of mask sizes from N=3 to N=11. As the window size increased, the oil painting effect becomes more significant as the increase in window size leads to a loss of edges. Hence, the image becomes smoother as the window size is increased. But, the image starts blurring when the window size becomes too large. This can be seen in figures 38 and 47 as the edges seem to have disappeared to a great extent.
Hence, according to me N=7 is the ideal window size wherein the oil painting effect is significant and at the same time blurring of image also does not take place.
From figures 39,40,48,49 it can be seen that when the images are represented using 512-bits, more number of colors are included and hence the image looks colorful. Therefore, the oil painting effect is more prominent in the 512-bit representation than the 64-bit representation.

c) Image Filtering- Creating Film special effect

1. Abstract and Motivation

The film special effect generated in this question can be examined by using the color channel relationship between the original image and the film effect image. An algorithm is developed on the film effect image and it is applied to the girl image to get the special effect on it.

2. Approach and Procedure

Theoretical Approach:

The equalized histogram of the input image can be found as follows:

$$y = f(x) = \int_0^x p_x(u)du$$

The equalized histogram of the output image can be found as follows:

$$y' = g(z) = \int_0^z p_z(u)du$$

As the input and the output image have same histogram, y=y' and hence the mapping from x to z is given as

$$z = g^{-1}(y') = g^{-1}(y) = g^{-1}(f(x))$$

Hence, the histogram of the original image is compared with the film effect image and the difference between every pixel value is noted. The minimum difference value is selected, and that particular intensity is chosen to be the intensity value. In this way mapping of pixel intensities is done.

Task: To design an algorithm for the film special effect by comparing the original and the film effect image and applying the algorithm on the girl image.

Algorithm for film special Effect:

Step 1: Read the input raw image "Original.raw'', "Film.raw", "Girl.raw" from three files into a 1D arrays whose dimensions are specified by image height, image width and Bytesperpixel.
Step 2: Convert the 1D image arrays into separate 1D arrays for R, G, B channels respectively.
Step 3: Mirror and perform color inversion on the girl and the original image.
Step 4: Calculate the count of each intensity value for the R, G, B channels of the film image and the original image.
Step 5: Find the probability for each intensity value in the R, G, B channel. This is done by dividing each intensity by the total number of pixels in the image.
Step 6: Calculate the cdf values for each channel of the original and film effect image.
Step 7: Find the difference between the cdf values of the original image with the film effect image for every intensity.
    Step 7a: Find the minimum difference and store that intensity value having the minimum difference in an array. Repeat this for G, B channels as well.
Step 8: Assign the values of these new intensities in the original R, G, B channel arrays.
Step 9: Assign the values of the array stored in Step 7a to the R, G, B components of the girl image.
Step 10: Combine the R, G, B arrays for the film effect image and the girl image having the new intensity values into a 1D array which will then be written to a file to be displayed as an output image.

3. Experimental Results

Figure 50: Original Image

Figure 51: Given Film Effect Image

Figure 52: The film effect image after applying the algorithm

Figure 53: Original girl image

Figure 54: Film effect on girl image

The histograms for the original image, film effect image and girl image are also included.

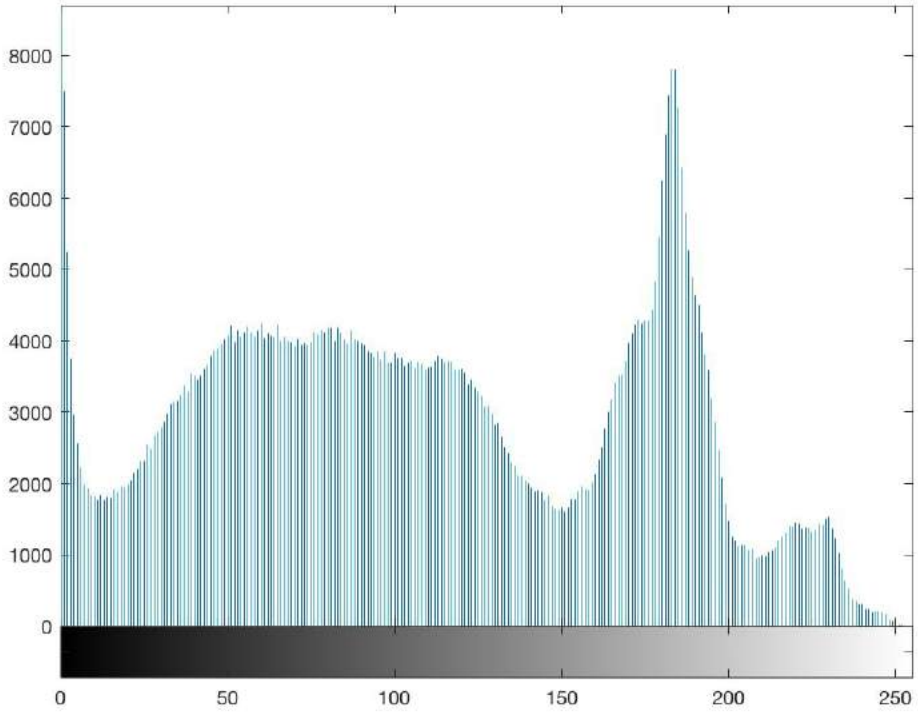


Figure 55: Original image histogram for R channel

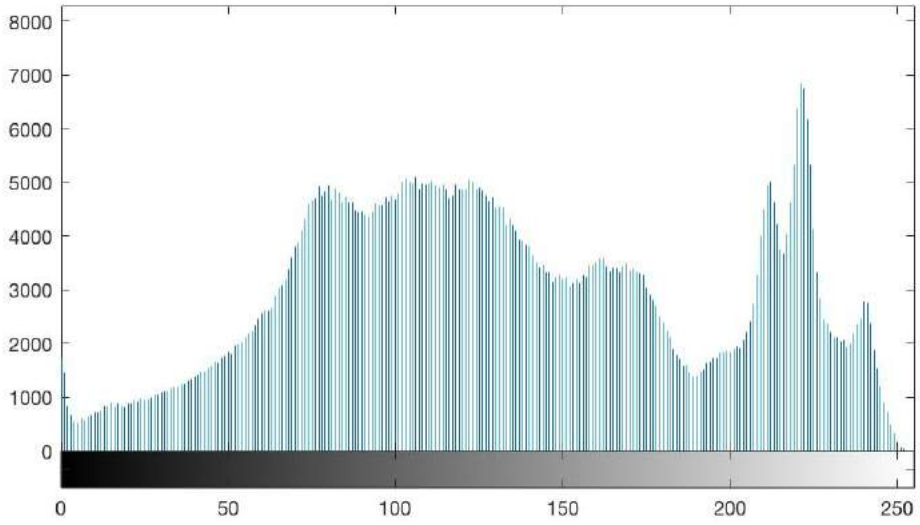


Figure 56: Original image histogram for G channel

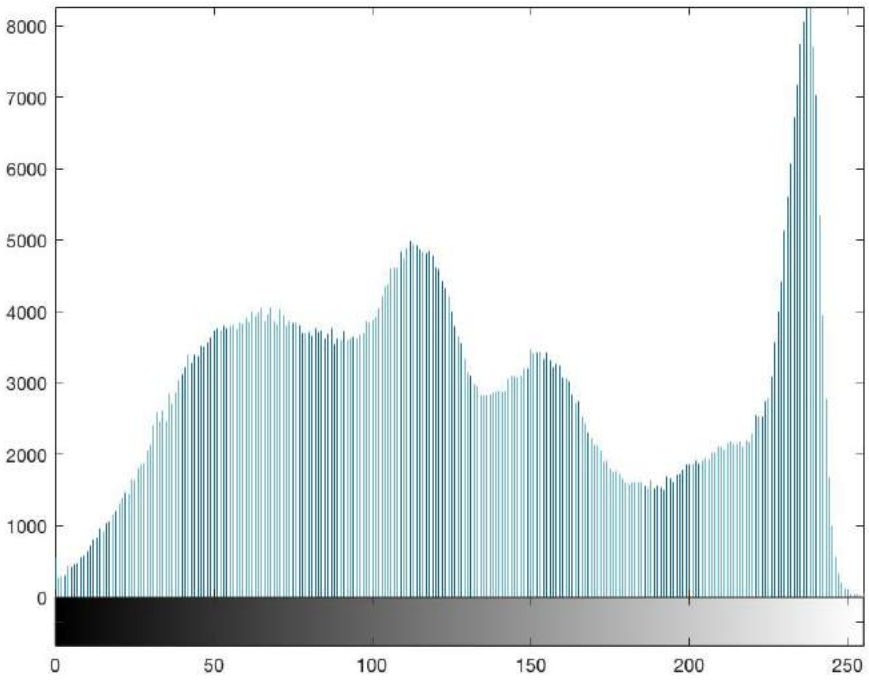


Figure 57: Original image histogram for B channel

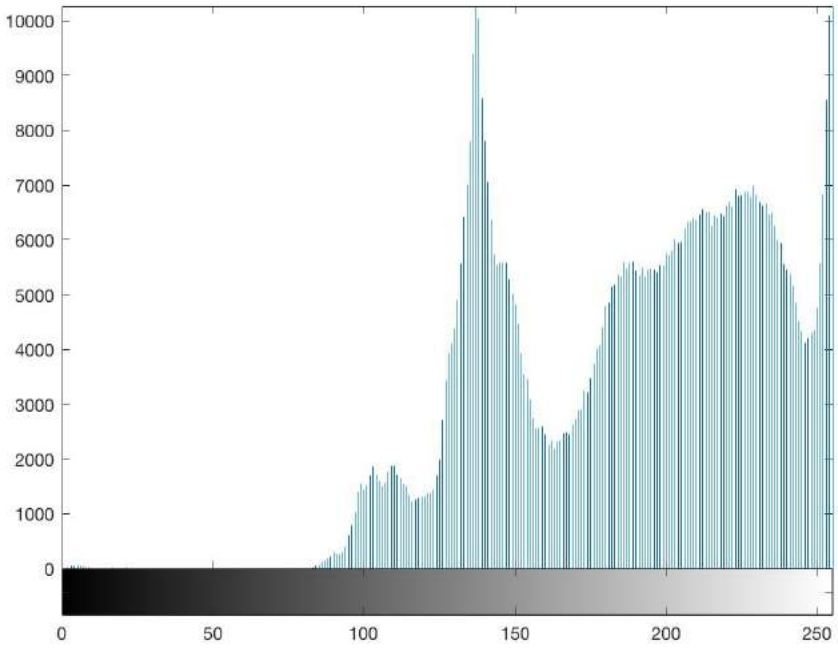


Figure 58: Film image histogram for R channel

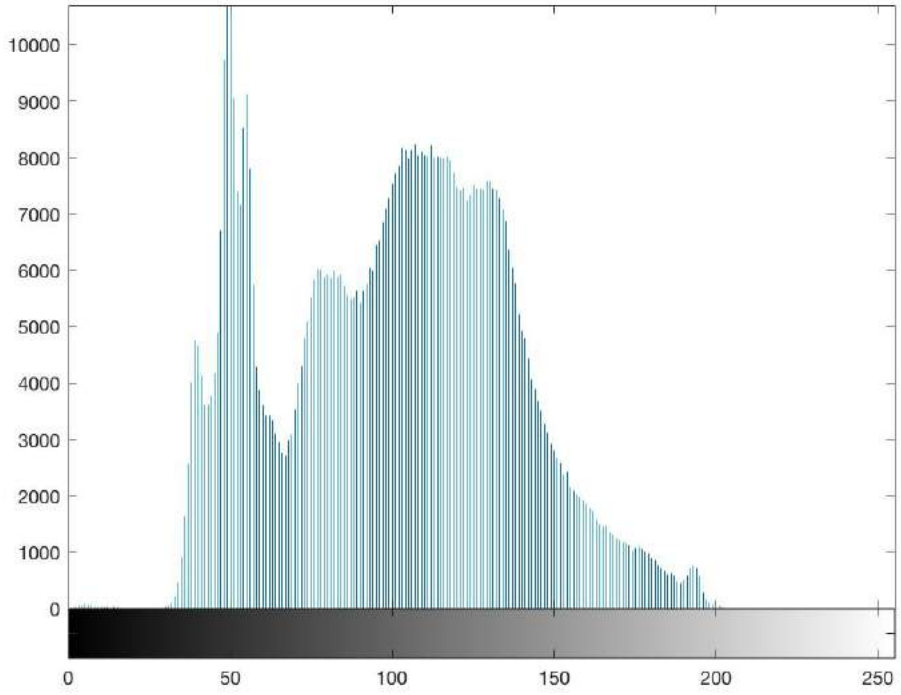


Figure 59: Film image histogram for G channel

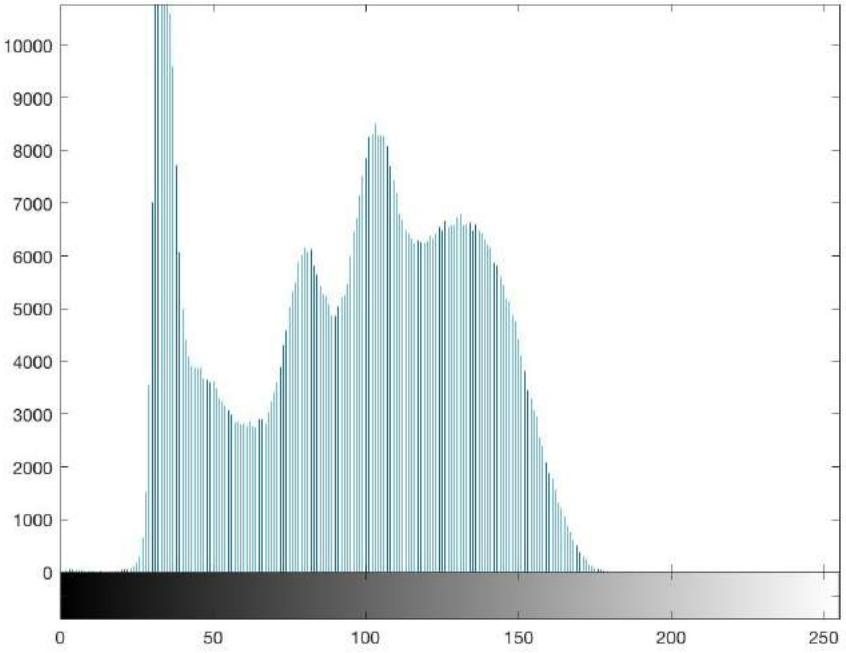


Figure 60: Film image histogram for B channel

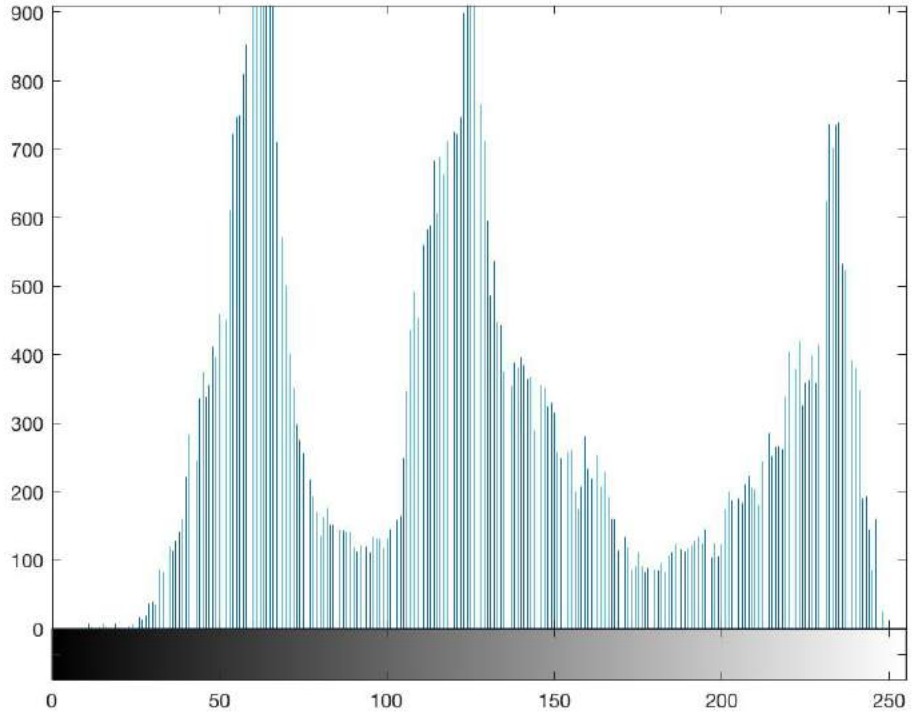


Figure 61: Girl image histogram for R channel

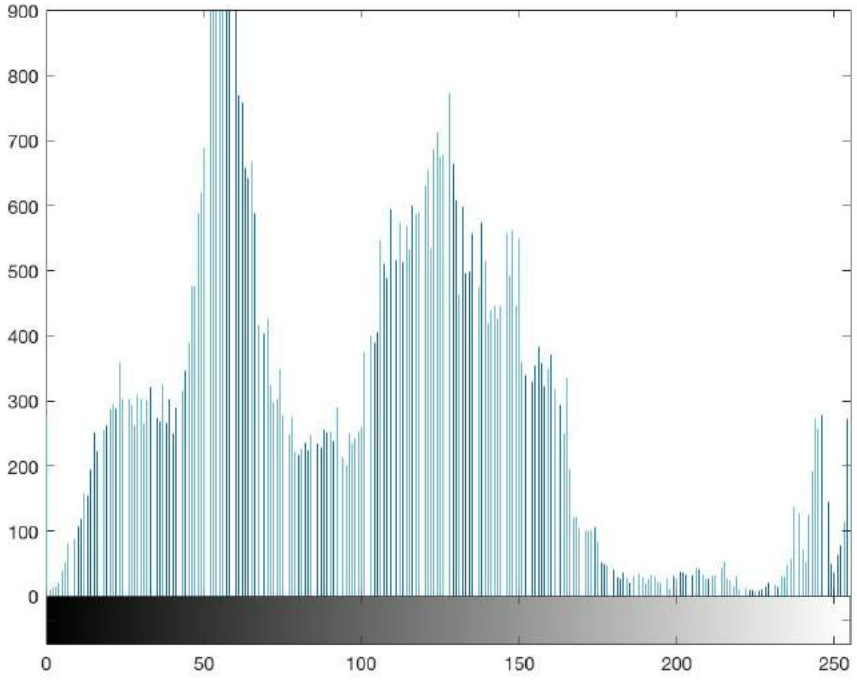


Figure 62: Girl image histogram for G channel

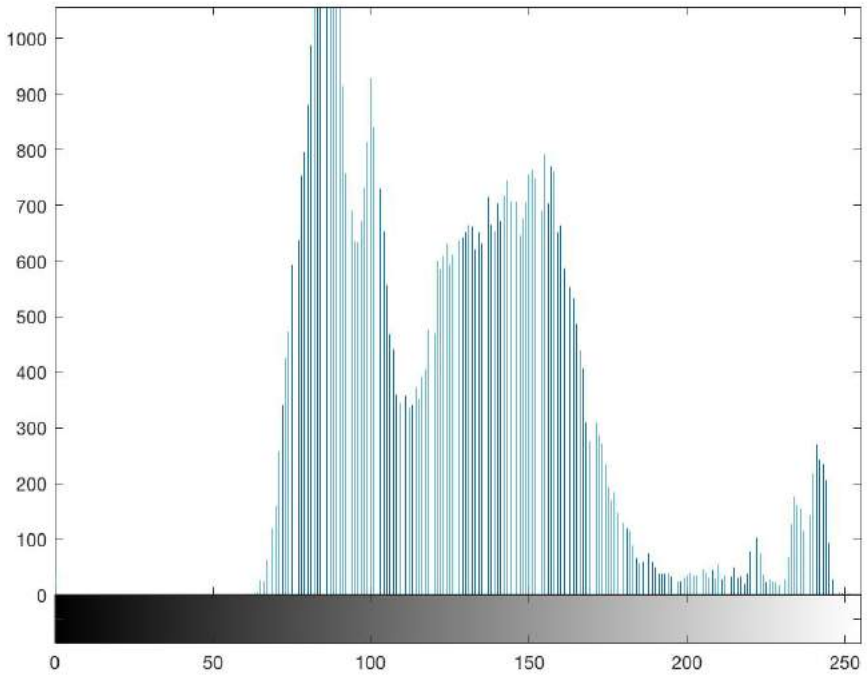


Figure 63: Girl image histogram for B channel

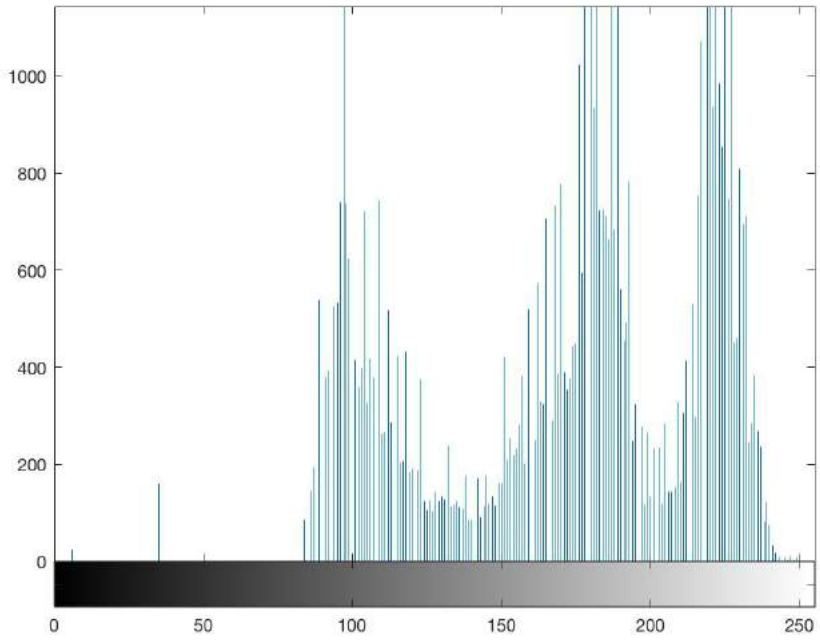


Figure 64: Girl special effect image histogram for R channel

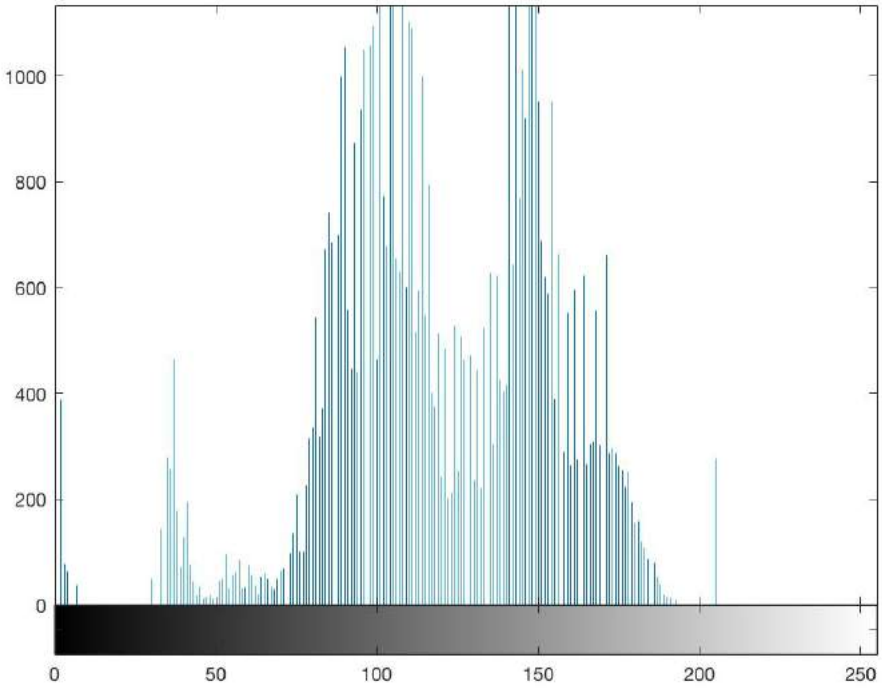


Figure 65: Girl special effect image histogram for G channel

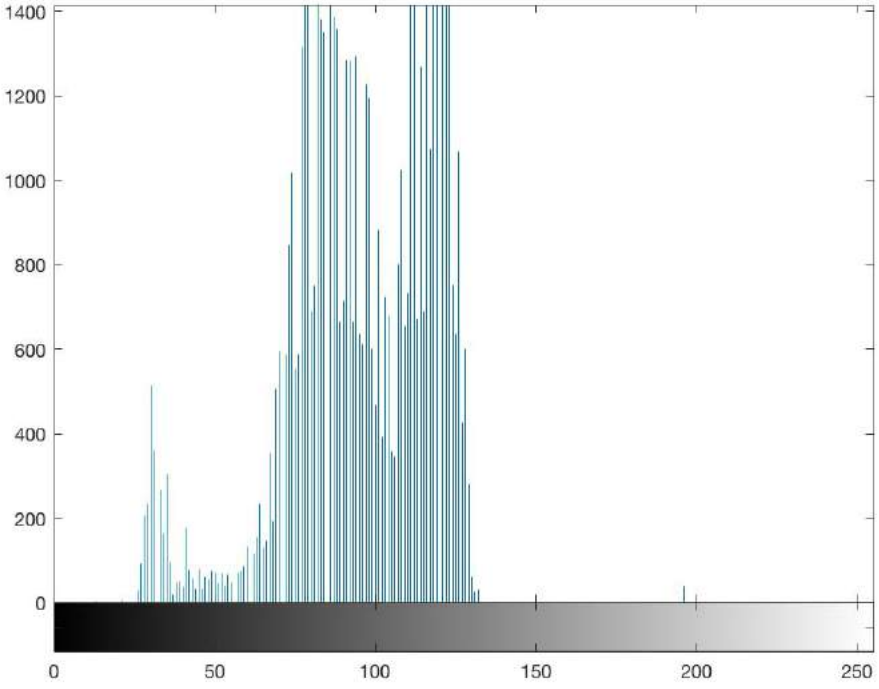


Figure 66: Girl special effect image histogram for B channel

4. Discussion

As seen from the figures 51 and 52, the film effect image generated by the algorithm and the one originally given are the same. Hence, the designed algorithm can be extended for the girl image to get the desired special effect. The special effect image for the girl is shown in figure 54.

On comparing the histograms for the original and the film effect image for each channel, it can be inferred that the histogram in the film effect image is a shrinked version of the original image. By studying this shrinking effect, the shrinking for the girl image histograms can be done.
As seen in figures 64,65 and 66, the histograms of the girl image are shrinked in comparison with the histograms of the original girl image for R, G, B channels given in figures 61,62,63.
It can also be observed that the film effect image histogram is also mirrored with respect to the original image histogram. Hence mirroring and color inversion of the girl image is done prior to applying the special effect algorithm.

Problem 3: Noise Removal

a) Mix noise in color image

1. Abstract and Motivation

Noise is a random phenomenon which is present anywhere in nature. In image processing techniques, in the data acquisition phase, due to imperfect sensors a lot of noise gets added to the system. Due to this noise in the image, a loss of important information takes place. Hence, it becomes extremely important to deal with the noise and to develop algorithms that can help reduce the noise content in the image.
There are different types of noise present in the image, primarily impulse noise and random noise. The random noise can have structure similar to uniform pdf or Gaussian pdf whereas the impulse noise has a structure of salt and pepper where salt is the noise present at 255 intensity and pepper is the noise present at 0 intensity.
Various filters are used to curb the noise namely Non-local means, BM3D, Adaptive non-local means, Weiner filter etc. Every method has its own advantages and disadvantages. However, in recent times the BM3D and Weiner filter have proved to be most effective.

Task: The main purpose of this exercise is to familiarize us with the basic 2 types of noise present in the image and to develop filters to eliminate that noise. We are also expected to cascade filters in different order as well as to change the window sizes used for denoising.

2. Approach and Procedure

The primary step in the denoising algorithm is to identify the types of noise present in each channel.

Figure 67: Original Lena Image

Figure 68: Noisy Lena Image

Figure 69: Noisy Lena image for R,G,B channels respectively.

When the image in Figure 68 is compared with the image in Figure 67, it is fairly obvious that image is corrupted by noise. The goal of this exercise is to reduce this noise prevalent in the noisy image to the maximum extent.

After observing figure 69, a rough guess can be made about the presence of the type of noise in the noisy image. It is clearly visible that there is salt and pepper noise in all the channels of the image. This is indicated by the dots visible in the R,G,B channel images. The only inference that has to be made is about the presence of Gaussian noise.

This can be done by studying the histograms of each channel separately. The histograms for the R, G, B channel are attached here.

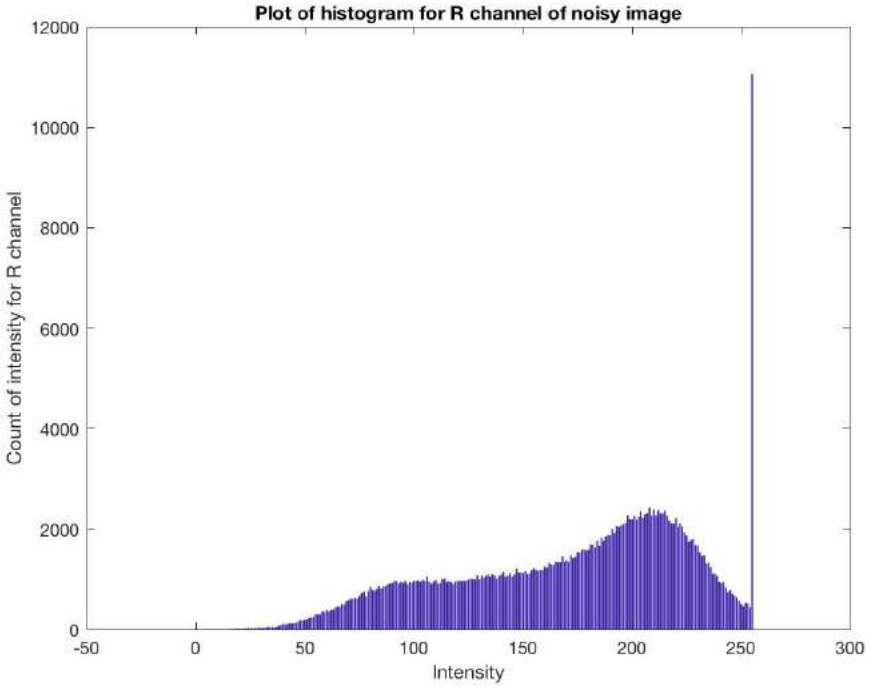


Figure 70: Histogram for R channel of noisy Lena image

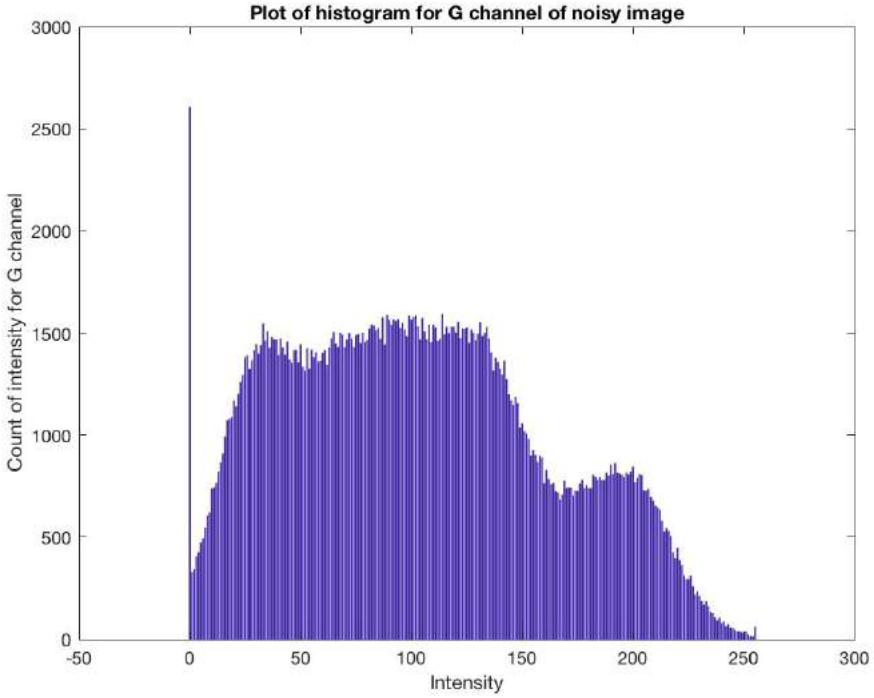


Figure 71: Histogram for G channel of noisy Lena image

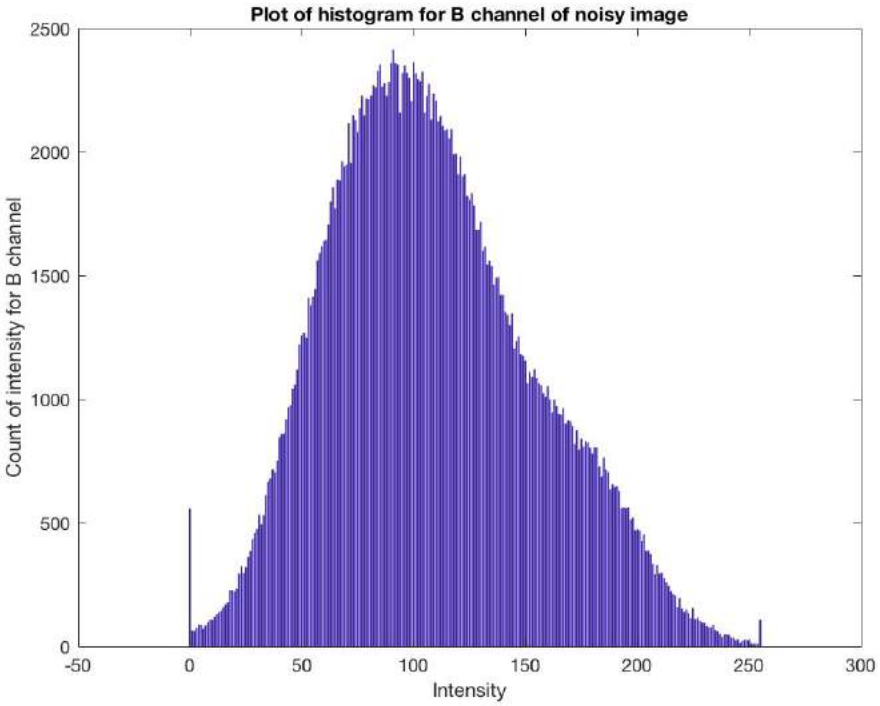


Figure 72: Histogram for B channel of noisy Lena image

As it can be seen from the histograms, the R channel has salt noise, the G channel has pepper noise and the B channel has salt and pepper noise. Considering the Gaussian noise, B channel definitely has Gaussian noise, but it's hard to infer that about the R and G channels. Hence, the outputs for the R and G channel without application of the Gaussian filter have to be studied.

To be remove the salt and pepper noise, various filter like median, Outlier filters can be used. I have chosen the **median filter** to remove the **salt and pepper noise**. To remove the random noise, Gaussian filter, Non-local means filter, mean filter etc. can be used. I have chosen the **mean filter** to remove the **Gaussian noise**.

To design a window size to be used to removal of noise, padding of the pixels have to be done. There are various methods available for padding the rows and columns of the image, I have chosen the method of pixel Replication.

The method for pixel replication is given here.

```
AA|ABCDEFGH|HH
AA|ABCDEFGH|HH
--------------
AA|ABCDEFGH|HH
II|IJKLMNOP|PP
QQ|QRSTUVWX|XX
YY|YZabcdef|ff
gg|ghijklmn|nn
oo|opqrstuv|vv
--------------
oo|opqrstuv|vv
oo|opqrstuv|vv
```

Figure 73: Pixel Replication Algorithm
(Source: http://www-cs.engr.ccny.cuny.edu/~wolberg/cs470/hw/hw2_pad.txt)

As mentioned earlier, the two filters to be used for removal of noise are mean and median filter.

Mean Filter: The concept used in a mean filter is to replace the central pixel in the image by the mean of its surrounding pixels. The surrounding pixels can be included by specifying the window size which will vary from N=3,5,7,9,11 in our case.

So, for a window of size 3x3, the mean filter will be as follows,

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Median Filter: In a Median filter, the central pixel in the window is replaced by the median of all the surrounding pixels in that window. A sorting algorithm has to be used to sort the pixel intensities in the increasing order and calculate the median from it.

Algorithm for Denoising:

Step 1: Read the input raw image 'Lena.raw'' and "Lena_mix.raw" from a file into a 1D arrays whose dimensions are specified by image height, image width and Bytesperpixel for the image.
Step 2: Convert the 1D array of both the images into separate 1D arrays for R, G, B channels respectively.

Step 3: Find the count of each intensity value for the R,G and B channel of the noisy image separately and export these values into a Text file.
Step 4: Convert the 1D array in Step 2 to 2D array for ease in masking procedure.
Step 5: Run two nested for loops which will traverse the image width and height of the image in order to design the median filter

Step 5a: Run two nested for loops to include the desired mask pixels and check if the mask index is exceeding the image height and width indexes.
Step 5b: Sort the array by using a sorting algorithm for the mask and assign the median value intensity of the mask to the central pixel of the mask. Repeat this procedure for G and B channels.

Step 6: Run two nested for loops which will traverse the image width and height of the image in order to design the mean filter

Step 6a: Run two nested for loops to include the desired mask pixels and check if the mask index is exceeding the image height and width indexes.
Step 6b: Find the sum of all intensity values in the mask and find the mean of them. Replace the central pixel by this mean intensity value. Repeat this procedure for G and B channels.

Step 7: Combine the R,G,B arrays having the mapped intensity values into a 1D array which will then be written to a file to be displayed as an output image.
Step 8: Calculate the PSNR values for R,G,B channels separately and also calculate the PSNR values for the combined image. This will give us the metric to test out denoising algorithm.
Step 9: Print the histogram of the three channels by importing the text file mentioned in Step 3 through Matlab.

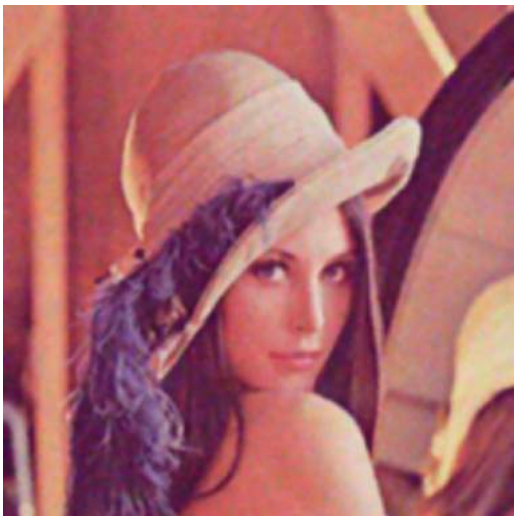
3. Experimental Results

Figure 74: Original Lena Image

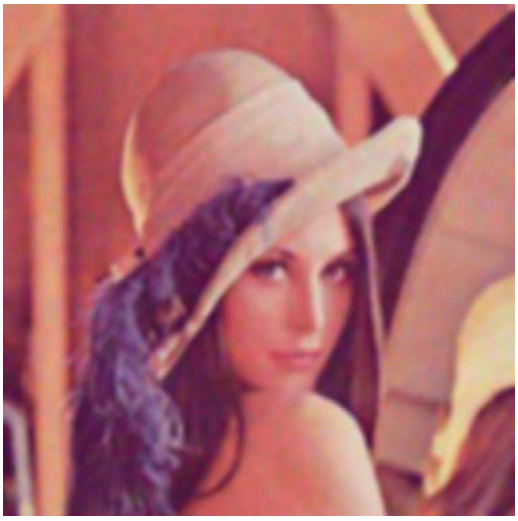
Figure 75: Noisy Lena image

The effect of different window sizes on the image is demonstrated as follows

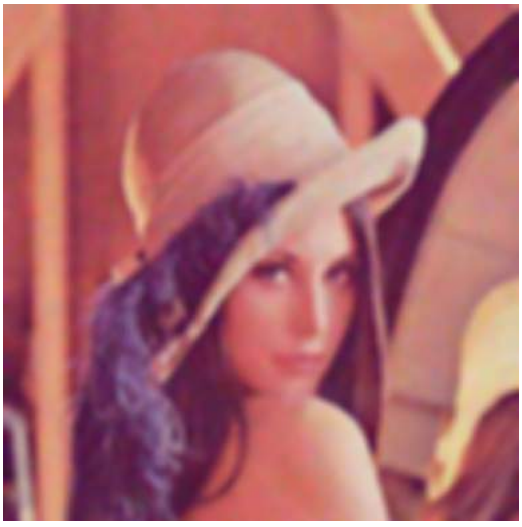
N=3



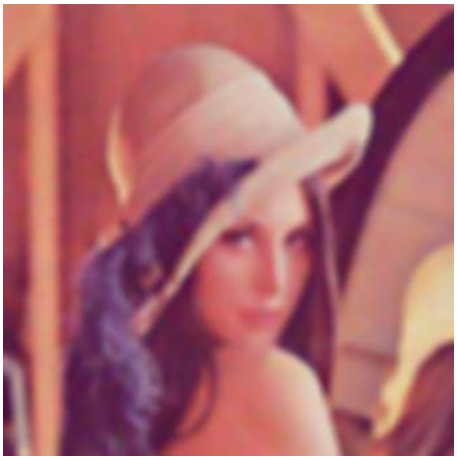
N=5



N=7



N=9



N=11

Figure 76: Effect of different window sizes on the image

The mean and the median filters can be cascaded in any order. The cascading of the filters is shown below:

Figure 77: Median+Mean

Figure 78: Mean+Median

The difference between both the images is quite visible. Figure 77 is clearer than Figure 78 which indicates that the noise is removed in a better way in Figure 77. The Median filter should be applied first to remove the salt and pepper noise and then the mean filter should be applied to remove the Gaussian noise.

As mentioned earlier, it's very hard to identify whether Gaussian noise is present in the R and G channels along with the impulse noise.
Hence, I have included images for the R and G channels after application of mean filter and without application of mean filter.

Figure 79: Original noisy R Channel

Figure 80: R channel after applying median filter

Figure 81: R channel after applying median+mean filter

Figure 82: Original noisy G channel

Figure 83: G channel after applying median filter

Figure 84: G channel after applying median+mean filter

As seen from the images, after application of the median and mean filter, the noise is removed in a better way. Hence, a combination of mean and median filter should be used in all the channels.

Next metric to evaluate the performance of the denoising algorithm is PSNR in dB.

Type of Filter	**R channel**	**G channel**	**B channel**	**All channels together**
Median+Mean 3x3	25.0456	27.4275	25.5607	25.9045
Only Median	23.945	26.6779	25.9536	24.6834
Only Mean	23.9934	22.8281	23.9382	23.2219
Mean+Median	23.8895	23.8593	25.1022	24.2446
Median for R, G and Median+Mean for B 3x3	23.945	26.6779	25.5607	25.2514
Median+Mean 5x5 for R Median+Mean 3x3 for G,B	25.4657	27.4275	25.5607	26.0013
Median+Mean 5x5	25.4657	26.2536	26.3652	26.0166
Median for R,G and Median+Mean for B 5x5	**25.4488**	**27.1172**	**26.3652**	**26.2644**

As seen from the table, the best combination is applying the median filter with window size=5 for R,G,B channels first and then the mean filter with window size=5 only for B channel. This combination gives the maximum PSNR value.

4. Discussion

Answer 1 a)
Yes, all the channels have different noises. The B channel has Gaussian and salt and pepper noise to a very great extent, whereas the R and G channels have salt and pepper noise which is dominating. However, they also have small content of Gaussian noise which should be removed by using the mean filter. This is visible from the histograms in Figure 70,71,72 and Figures 79 to 84.

Answer 1 b)
As all the channels have different intensity of noise in them, all of them should be dealt with separately. Hence, each channel should be applied the median and mean filters separately.

Answer 1 c)
As mentioned earlier, the images have salt and pepper and Gaussian noise in them. Hence, the median filter should be used to remove the salt and pepper noise and the mean filter should be used to remove the Gaussian noise.

Answer 1 d)
The filters should not be cascaded as the best combination is obtained by using the median filter first and then the mean filter. This is visible from the Figures 77 and 78, as the figure 77 which has median filter first and mean filter next has better noise removing capabilities than the other combination. This is actually quite intuitive as when the mean filter will be applied first, it will average out the noise and finding the median won't help in reducing the noise completely. However, cascading by changing window sizes can be done to get the most effective combination. The best combination is given for window size=5.

Answer 1 e)
The effect of different window sizes on the noisy image can be seen in figure 76. When the window size is increased, the noise content of the image definitely reduces, but the image starts to become blurry as well. For N=3, the image has a lot of noise but is sharp. On the contrary for N=11, the noise content is less but the image does not have visible edges. Also, as N is increased the computational time is also increased.

Answer 2)
According to me, the best performance by the denoising algorithm is given for window size N=5 and the combination of median filter for R,G channels first and then the median and mean filter only for B channel. This combination gives a very high PSNR and the reduction in noise content is also quite visible.
However, using the above method does not remove noise completely. The image has some blurry parts and hence this is not the optimal algorithm.

Also, the **SSIM Index** should be used along with PSNR to test the performance of our denoising algorithm. SSIM Index takes into account the structure of the image and hence is more effective. Using other filters like **Non-Local Mean, Adaptive Non-Local Mean, BM3D, Weiner filter** will reduce the noise content in the image to a great extent.

b) Principal Component Analysis (PCA)

1. Abstract and Motivation

Answer 1)

Principal Component Analysis is basically a dimensionality reduction algorithm in which the axes having the maximum variance are preserved whereas the axes having minimum variance are neglected. PCA is mainly used in high-dimensional dataset to reduce the dimensions of the dataset but is also used in image processing as a denoising algorithm.
Signal data that is present in the image has a structure attached to it and hence is considered to be having high variance whereas the noise does not have any structure and hence has low variance. Therefore, the main purpose of PCA as a denoising algorithm is to predict the principal axes that contain the maximum variance and discard the axes with minimum variance. It is quite intuitive that the largest variance signal will be unaffected by noise and hence if we try to reconstruct the original data using only the principal components, only signal data will remain, and noise will be suppressed. The main goal of PCA denoising algorithm is to find the optimal principal axes.

Components in the PCA denoising algorithm are the principal axes having maximum variance which can be used to reconstruct the data. Hence, optimal bases have to be chosen so as the loss of information is not too huge.
The patch size of the image is used to describe the number of pixels that are to be considered as a time.

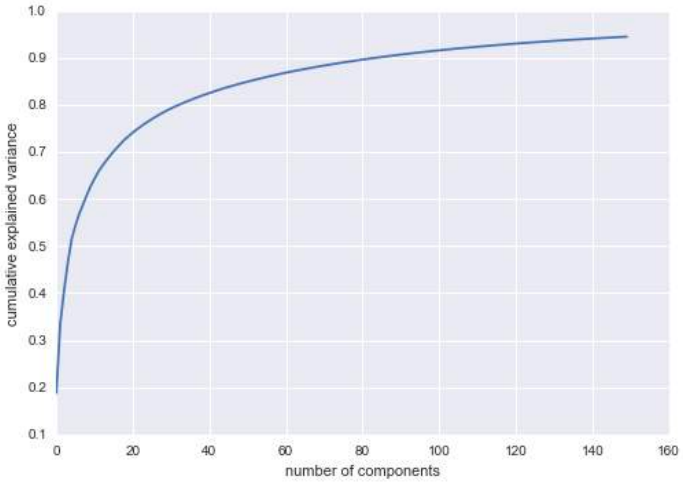


Figure 85: choosing the number of components
(Source: https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html)

For instance, in the above figure it can be seen that 10 components contribute to around 75% of the variance whereas rest 80% of the components contribute to remaining 25% of the variance. Hence, the first 10 components can be chosen to reduce the dimension of the data. Also, suppose in an image the patch size is defined to be 8x8, it has a total of 64 pixels that will be considered, and a window will be applied on them.

PCA can be applied in patches such that the patches having minimum variance can be grouped together. Hence, in this way PCA can be applied to the patches having similar structure so that reconstruction won't be a problem. The patches are grouped locally in Local PCA rather than globally, so we have to mention the patch size as well as the window size while running a PLPCA based algorithm. Sliding windows are used to collect the patches.

The optimum value of the Signal variance to noise variance is found to be **2.5-3**.

2. Approach and Procedure

Task: To implement the PCA code in Matlab and test by changing some parameters such as patch size, number of components etc.

I changed the window size and the patch size for different values to get the optimum value.

3. Experimental Results

A table for all the parameters and the PSNR values for them has been prepared [1]

Patch size(hP)	**Window size(hW)**	**PSNR in dB**
7	11	31.15
9	11	31.21
11	11	31.06
13	11	30.79
9	13	31.27
9	15	31.29
9	17	31.31
9	19	31.34
9	**21**	**31.37**

As seen from the table, whenever the window size is increased, the PSNR increases as a large number of similar patches of pixels are included in the window. Hence, the dimensionality reduction happens effectively, and the noise content is suppressed.

The best **PSNR** value of **31.37dB** was obtained for the combination of **patch size=9 and window size=21.**

The noisy image(sigma=25) and the denoised images by using PCA are included herewith.

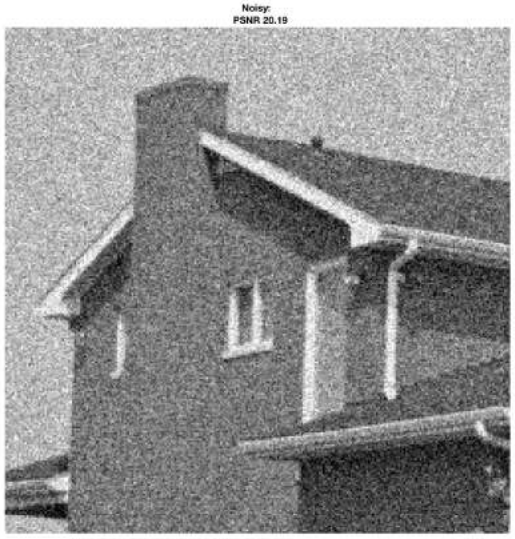


Figure 86: Noisy House image

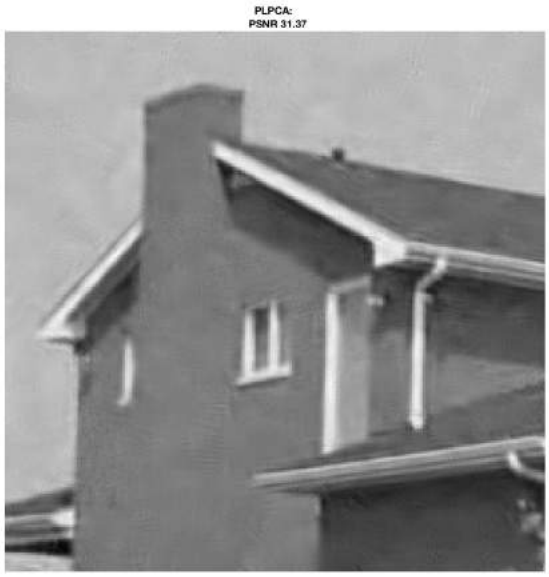


Figure 87: Denoised House image by using PCA

Answer 4)

Also, the algorithm developed in Question 3a was run on the House image and the PSNR values were calculated for different window sizes.

Method Used	PSNR Value
PCA	31.37
Median+Mean filter	28.1554

Hence, it is quite evident that the PCA filtering approach works better than the conventional mean and median filter approach as in the PCA large window sizes can be defined. By using mean and median filters, the window size can't be too large as the image starts to become blur when the window size is increased to a large extent. Also, in PCA the similarity of group of pixels is considered which will help is denoising rather than using mean filter method where the noise is averaged.

Figure 88: Denoised image using Median+Mean filtering

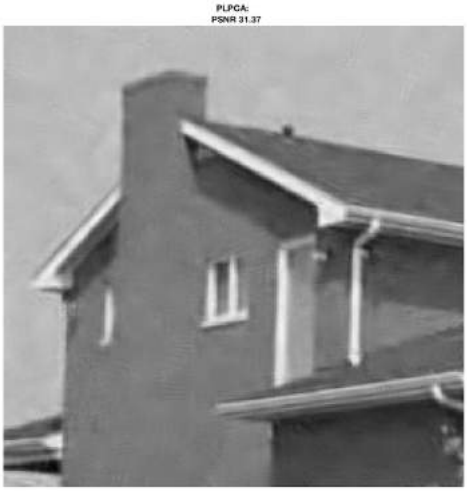


Figure 89: Denoised image using PCA method

4. Discussion

The Principal Component Analysis method is more effective than many other denoising methods as in PCA the reduction of noise takes place using the structural forms of the data.
However, there are some disadvantages of using the PCA approach as overfitting of data takes place and also the computational complexity increases. Hence, other methods such as BM3D and Weiner filter are used to reduce noise in the image.

c) Block Matching and 3D (BM3D) transform Filter

1. Abstract and motivation

BM3D algorithm is used mainly to denoise a particular image. This algorithm is based on enhanced sparse representation of a particular image in transform domain. BM3D uses the same concept as that of PCA as the pixels in the BM3D algorithm are segregated into patched depending on the similarity of the pixels in a particular patch. Three steps are involved to carry out the BM3D algorithm.

2. Approach and Procedure

Task: To implement the BM3D denoising algorithm on the "House.raw" image and change the parameters of the code to get the best denoising result.

Algorithm for BM3D: [2]
Step 1: Grouping
In this step, the entire image is traversed and the 2D fragments of the image are transformed into 3D arrays. While traversing through the image, structural similarity is seen between the pixels which help on grouping the pixels into patches. This is done by stacking all the patches one above the other, which results into a 3D array. This step allows us to enable use of higher dimensional filtering of each group.

Step 2: Grouping by Matching
After grouping the image into patches, the patches are compared with each other to see if there is any similarity between the patches. This step uses the concept of 'K-means clustering' and 'Vector Quantization'. In the K-means clustering method, the distances between the patches is found out and if the distances between the patches is less than the specified threshold, they are grouped together as clusters. Minimum distances between the patches indicate that there is a high similarity in them. Distance is just one metric to test the similarity between patches. Other metrics can also be used to group the patches by matching. This is a recursive procedure as every patch needs to be compared with every other patch as a lot of similarities can be there between them. When the similarity between the patches is examined the average of that value is taken and assigned to every pixel in that patch. For instance, if one patch has a similarity of 40% with one other patch and a similarity of 70% with the other patch, the average of these similarity values is taken and assigned to the pixels.

Step 3: Collaborative Filtering
Collaborative filtering is a very novel concept which is used in BM3D algorithm. This step consists of three other steps as follows:

a) 3D transformation of 3D groups
b) Shrinkage of transform spectrum
c) Inverse 3D transformation

When the clustering is done for the patches, estimates are provided for each of them. If identical patched are present in the cluster, then the average of the estimates comes out to be good. But if non-identical patches are clustered together, the average of estimate is bad. Hence, identical patched should be combined together ideally. The number of blocks in a particular group is related to the residual variance which is in turn related to the estimates. Hence appropriate blocks should be selected for estimating the thresholds. The Discrete Cosine Transform (DCT) converts the image from spatial domain to Fourier domain. Collaborative hard thresholding is done by using transform coefficients whereas the approximated values are gained using inverse 3D transform. Estimates are obtained for the images.

3. Experimental Results

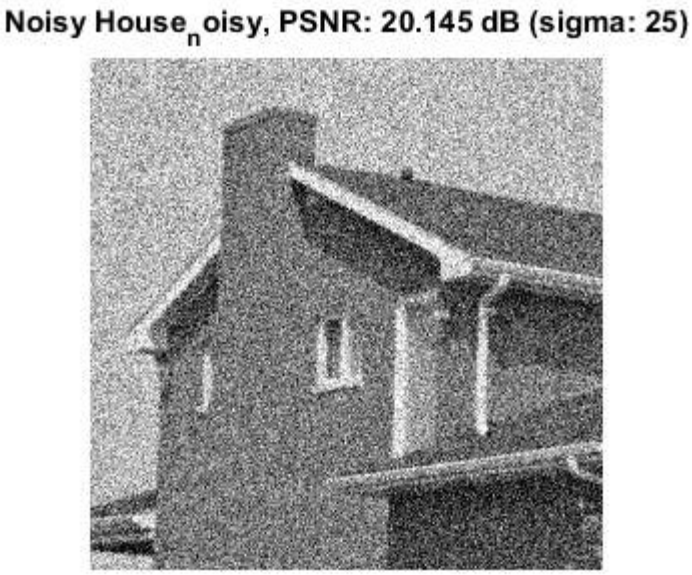


Figure 90: Noisy House image

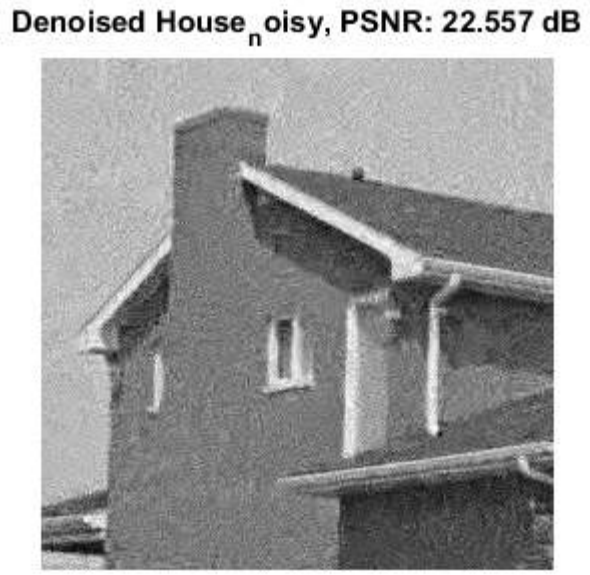


Figure 91: Denoised House image using BM3D algorithm

The BM3D code was run by changing certain parameters to find the optimum parameter.
A table of the observations is being attached.

Lambda 3D	Beta	NStep	N2	Beta-Weiner	NStep Weiner	PSNR without Step 2	PSNR with step 2
2.7	8	3	16	8	3	22.032	22.017
2.5	8	3	16	8	3	22.330	22.445
2.5	4	2	64	4	2	**22.566**	**22.549**

4. Discussion

BM3D has a lot of advantages as it has a very fast and efficient realization which can be achieved by reducing the number of blocks, reducing complexity of grouping etc.
The disadvantages are that in extremely noisy images, it's very hard to eliminate noise and also this algorithm can lead to blurring of images.

Answer 2)

If we use step 2, it gives us a basic estimate of the values, which helps in grouping by block matching. Block matching is used instead of K-means clustering as in K-means the similarities of the blocks are observed rather than observing the edges.

Answer 3)

BM3D is a spatial domain as well as frequency domain filter as this algorithm used Weiner filter which works in the frequency domain.

REFERENCES

[1] Deledalle, Charles-Alban, Joseph Salmon, and Arnak S. Dalalyan. "Image denoising with patch based PCA: local versus global." BMVC. Vol. 81. 2011.

[2] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3-d transform- domain collaborative filtering,” Image Processing, IEEE Transactions on, vol. 16, no. 8, pp. 2080–2095, 2007.

[3] http://josephsalmon.eu/code/index_codes.php?page=PatchPCA_denoising)

[4] http://www.cs.tut.fi/~foi/GCF-BM3D/