EE 569 Homework #1

Prof. C.-C. Kuo

Hrishikesh Hippalgaonkar

hippalga@usc.edu

Table of Contents:

1. Problem 1

1.1 Abstract and Motivation…….. 3
1.2 Approach and Procedure……. 4
1.2.1 Colour-to-Grayscale conversion…… 4
1.2.2 CMY colour space conversion……... 5
1.2.3 Bilinear Interpolation…….. 5
1.3 Experimental Results…… 6
1.3.1 Colour-to-Grayscale conversion…… 6
1.3.2 CMY colour space conversion……... 7
1.3.3 Bilinear Interpolation…….. 8
1.4 Discussion……. 8

2. Problem 2

2.1 Abstract and Motivation…….. 10
2.2 Approach and Procedure……. 11
2.2.1 Histogram Equalization…….. 11
2.2.2 Image Filtering – Creating Oil Painting Effect…… 12
2.2.3 Image Filtering – Creating Film Special Effect…… 13
2.3 Experimental Results…… 14
2.3.1 Histogram Equalization…….. 14
2.3.2 Image Filtering – Creating Oil Painting Effect…… 17
2.3.3 Image Filtering – Creating Film Special Effect…… 20
2.4 Discussion…….. 21

3. Problem 3

3.1 Abstract and Motivation…….. 22
3.2 Approach and Procedure……. 22
3.2.1 Noise removal…….. 22
3.2.2 Patch based Principal Component Analysis……..... 25
3.2.3 Block matching and 3-D (BM3D)……... 25
3.3 Experimental Results…… 27
3.3.1 Colour-to-Grayscale conversion…… 27
3.3.2 Patch based Principal Component Analysis…… 29
3.3.3 Block matching and 3-D (BM3D)……... 30
3.4 Discussion…….. 32

References…… 33

1. Problem 1

1.1 Abstract and Motivation:

Light that does not have any colour is called monochromatic light, and its only attribute is its *intensity* or *luminance*. The range of values of such monochromatic light from black (minimum intensity) to white (maximum intensity) is usually referred to as *gray scale* and such monochromatic images are called as *gray images* or *gray scale images*. Each channel of a colour image itself can be thought of as a gray scale image too, since each channel has a minimum and maximum intensity, regardless of the colour. It is only when all the gray scale channels are combined that we get a colour image.

Many digital image processing algorithms, such as edge detection, don't require the colour information in the image. In such situations the image can be converted to a gray scale image, thus preserving its luminance but removing all the colour information. Consequently, colour to grayscale conversion is frequently used in digital image processing.

Colour space (also called as a colour model or colour system) can be defined as a coordinate system and a subspace (in that system), where each colour can be uniquely identified by a single point. There are numerous colour models in use today, and some of the most common colour models are:

- RGB model – (Red, Green, Blue)
- CMY model – (Cyan, Magenta, Yellow)
- CMYK model – (Cyan, Magenta, Yellow, Black)
- HSL model – (Hue, Saturation, Lightness)

RGB is an additive colour spectrum in which the addition of all the colours results in white colour. It is a colour space for the emittance of light. And for these reasons the RGB model is the most commonly used hardware oriented colour model. It is used for colour monitors, various digital cameras, scanners etc. where the light must be *emitted*.

As opposed to electronic displays, printing works on the principle of absorption and reflection of light. In printing, inks, dyes and toners do not properly represent the emittance of light. For these reasons, the RGB colour space cannot be used for printing processes. Instead, the CMYK colour space must be used. This colour space represents the absorption of light, where black is formed by combining all the primary colours.

Most of the digital image processing work, like photo editing, image enhancement, image restoration, is done in the RGB colour space. But when these photos or images need to be printed, they

must be converted to CMYK colour space. Thus, colour space transformation from RGB to CMYK is an important digital image processing step.

Many image processing algorithms require rotation and/or scaling of images. But the pixel coordinates that are obtained from such transformations may very well be non - integer coordinates. One could simply enlarge the original image or *scale* the image to the required dimensions. But this scaling may result in blocks or patches in the enlarged image. In the bilinear interpolation approach of scaling the image, the intensities at the new pixel locations are calculated using the nearest neighbouring pixels from the un-sized original image.

1.2 Approach and Procedure:

There are three parts in this section. The first part concerns with conversion of colour image to grayscale. The second part deals with conversion of RGB colour space to CMY colour space. And the third part elaborates the procedure for bilinear transformation.

1.2.1 Colour-to-Grayscale Conversion:

Three methods are described in the homework handout for converting a colour image to grayscale.

1. **Lightness method:** In this method, average of the most prominent and least prominent colour of a pixel is calculated.

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

Where ***I*** is the grayscale intensity of the output image.

2. **Average method:** In this method, the output grayscale intensity of a pixel is just the average of the intensity of its three channels.

$$I = \frac{(R + G + B)}{3}$$

3. **Luminosity method:** This method uses a weighted sum of the intensities of the three channels of a pixel.

$$I = 0.21R + 0.72G + 0.07B$$

The raw image is read into an array and its three channels are separated and stored in three different arrays. Then the pixel intensities corresponding to each channel are passed from these arrays to one of the formulas stated above to calculate the new grayscale intensity

1.2.2 CMY Colour Space Conversion:

The conversion of RGB to CMY is quite simple. Again, the image is read into an array and then its channels are separated. The intensities in all the three channels are normalized i.e. each channel intensity is divided by 255 (the maximum possible intensity). This brings each channel intensity in the range of 0 to 1. The conversion from RGB to CMY can be done using the following equations:

$$C = 1 - R$$
$$M = 1 - G$$
$$Y = 1 - B$$

Where ***R, G, B*** are the channel intensities of the original RGB image and ***C, M, Y*** are the CMY intensities of the output image. The CMY intensities are in the range of 0 to 1 and need to be re-normalized by multiplying with 255. This gives the final CMY image.

1.2.3 Bilinear Interpolation:

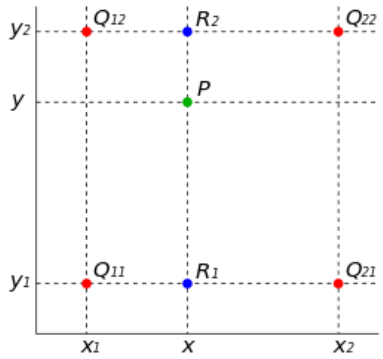


Figure 1: Bilinear Interpolation

The points $Q_{11}, Q_{22}, Q_{12}, Q_{21}$ are the pixels in the original image. Let P be the new pixel in the high-resolution image. The intensity of this pixel is calculated based on its nearest four neighbours from the original. In this case, $Q_{11}, Q_{22}, Q_{12}, Q_{21}$ are the nearest neighbours. Assume the perpendicular distance between each pixel of the original image to be 1 unit. If distance between Q_{12}, R_2 is Δx , then distance between Q_{22}, R_2 is given by $(1 - \Delta x)$. Similarly, if distance between R_2, P is Δy , then distance between R_1, P is given by $(1 - \Delta y)$.

The intensity of the new image is given by

$$I_p = (1 - \Delta x)(1 - \Delta y)I_{12} + (\Delta x)(1 - \Delta y)I_{22} + (1 - \Delta x)(\Delta y)I_{11} + (\Delta x)(\Delta y)I_{21}$$

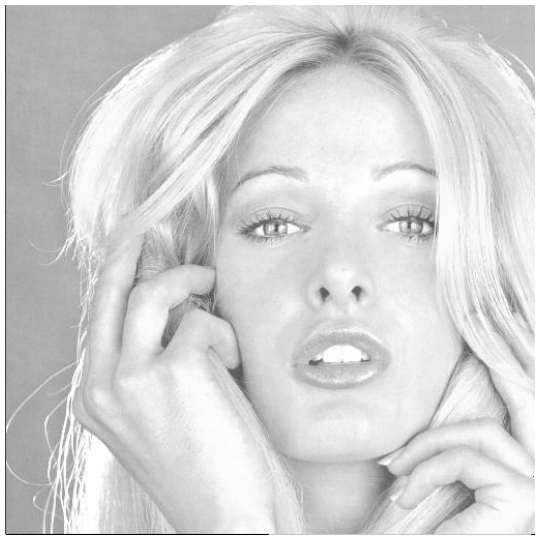
where, $I_{11}, I_{22}, I_{12}, I_{21}$ are the intensity levels of $Q_{11}, Q_{22}, Q_{12}, Q_{21}$ pixels and I_p is the intensity of P

1.3 Experimental Results:

1.3.1 Colour-to-Grayscale conversion:

(a) Grayscale conversion using Average method

(b) Grayscale conversion using Lightness method



(c) Grayscale conversion using Luminosity method

Figure 2: Colour-to-Gray conversion

1.3.2 CMY Colour Space Conversion:

(a) *Bear.raw* image in CMY colour space

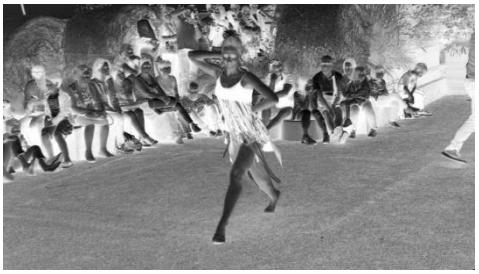
(b) Cyan channel

(c) Magenta channel

(d) Yellow channel

Figure 3: CMY colour space conversion for *Bear.raw* image

(a) *Dance.raw* image in CMY colour space



(b) Cyan channel (b) Magenta channel (b) Yellow channel

Figure 4: CMY colour space conversion for *Dance.raw* image

1.3.3 Bilinear Interpolation:

(a) Original *Airplane.raw* image (512x512) (b) Resized *Airplane.raw* image (650x650)

Figure 5: Bilinear Interpolation of *Airplane.raw* image

1.4 Discussion:

Colour-to-Grayscale conversion: The *Averaging* method is a quick way of converting a colour image to grayscale. But it has it cons – it poorly performs in representing the shades of gray as perceived by humans. The *Luminosity* method takes care of this problem. It takes into consideration the fact that the number of colour sensing cones in human eye for each of the R, G, B colours are not uniform. Thus, the weighted sum of the R, G, B intensities is a good conversion scale. The *Lightness* method can also can also be called as the *desaturation* method. Desaturating an image works by converting an image from RGB colour space to HSL colour space, and then forcing the saturation to zero. Thus, a colour is

converted to *least saturated variant*. A pixel can be desaturated by averaging the maximum of (R, G, B) and minimum of (R, G, B)

CMY Colour Space Conversion: As mentioned earlier, CMYK colour space is mainly used for printing processes. The reason for adding the extra *K (black)* channel is the fact that the *'black'* generated by mixing C, M, Y is quite unsatisfactory. Also using a separate cheap but effective black colour is quite productive, instead of mixing the expensive Cyan, Magenta and Yellow colours to make an ineffective black. Also in CMYK printing, halftoning plays an important part. It helps saves costly colour ink. Moreover, without halftoning, only seven colours can be produced, but with halftoning a continuous range of colours can be generated.

One can observe that often RGB images printed in CMYK don't look the same as they seem on colour displays. This is because the CMYK colour space doesn't cover all the colours in the entire RGB colour space. It can be possible that certain colours of an image displayed on electronic screens cannot be printed on paper. Thus, we observe a change in colours in the printed version of the same image. Most of the time this slight change in colour can be overlooked. Since RGB colour space maps the entire CMYK colour space, it is possible to display CMYK images on a display properly.

Bilinear Interpolation: Interpolation can be defined as calculating new points new data points in the range of given set of known point. Example of a 1-D interpolation is finding a point lying on a line joining two known points. Bilinear interpolation is interpolation done in 2-D. In bilinear interpolation, new points are sampled first in one direction, and then in other direction. Most common application of 2-D interpolation is image resizing (*zooming* and *shrinking). Zooming* can be defined as *over-sampling*, i.e. calculating new data points, while *shrinking* may be defined as *under-sampling*, i.e. removing data points.

There are other forms of image interpolation techniques used for *zooming* and *shrinking*, such as *nearest neighbour interpolation* and *bicubic interpolation*. Bicubic interpolation involves the sixteen nearest neighbours. The intensity value assigned to point $(x, y)$is obtained using equation

$$v(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

where, the sixteen coefficients are determined from the sixteen coefficients in the sixteen unknows that can be written using the sixteen nearest neighbours of point (x, y). Bilinear interpolation performs better than nearest neighbour interpolation but bicubic interpolation performs slightly better than bilinear interpolation. But the computational burden is the greatest for bicubic interpolation, followed by bilinear interpolation. Nearest neighbour interpolation has the least computational burden.

2. Problem 2

2.1 Abstract and Motivation:

The first part of the problem deals with histogram based image enhancement. Consider an image having intensities in the range *[0 (I – 1)]*. The histogram of an image can be defined as a discrete function $h(i_k) = n_k$ where i_k is the *k*th intensity and n_k is the total number of pixels in the image having this specific intensity. Generally, the histogram is normalized by dividing each value by the total number of pixels *N* in the image. Thus, the histogram function can also be represented as:

$$h(i_k) = \frac{n_k}{N}$$

Furthermore, $h(i_k)$ can also be defined as probability of occurrence of intensity i_k in the image. Histogram is a very important statistical tool for the following reasons:

- Histograms are easy to calculate
- Histograms provide a pictorial description of statistical properties of an image

Histograms are used in various image processing applications such as image filtering, image restoration, image enhancement etc. One of the image enhancement methods which uses histograms is the histogram equalization techniques. This technique adjusts the overall contrast of the image. It distributes the intensities of the original image over a larger range of intensities on the histogram. Thus, the areas in the image with low contrast become brighter and areas with high contrast become less bright.

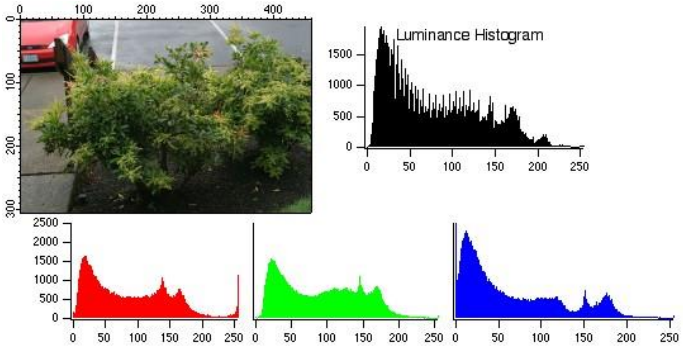


Figure 6: Sample histogram

The second of part problem deals with creating an image filter which renders the image an *oil painting effect.* Two primary features of such an *oil painting effect* are *colour gradients* and *colour intensities.* Real life oil paintings usually have small colour palette and express a low degree of detail as compared to digital images with large colour palette and high degree of detail. Thus, a filter can be implemented to reduce the colour palette and the degree of detail of the input image to give it an *oil painting effect.*

The third part of the problem concerns itself with creating a filter which mimics the transformation applied to a given image (*film special effect)* and then applying this transformation to other images. This can be carried out by the *histogram matching* technique. This technique automatically enhances the histogram of an image to match with the histogram of the other image.

2.2 Approach and Procedure:

There are three parts in this section. The first part concerns with histogram equalization. The second part deals with implementing the *oil painting effect* filter. And the third part elaborates the procedure for implanting *film effect* filter.

2.2.1 Histogram Equalization:

(a) Transfer-function-based histogram equalization:

First, separate the three channels of the input image. Each of the three channels must be equalized separately and then combined back to get the final equalized image. Let the histogram function of the first channel of the image, having pixel intensities in the range of *[0 (I-1)]*, be represented as:

$$h(i_k) = \frac{n_k}{N}$$

i_k is the kth intensity, n_k is the total number of pixels in the first channel having this intensity and N be total number of pixels in the first channel.

Let, $g(i_k)$ be the histogram of the equalized channel. Thus, $g(i_k)$ can be calculated as

$$g(i_k) = T(k) = (R-1)\sum_{i=0}^{k} h(i_k)$$

where, $T(k)$can be defined as a transform which acts on the kth intensity of the input image, and $[0 \ (R-1)]$ is intensity range of the desired equalized image. Thus, the intensity levels of the input image are mapped to new intensity levels, thereby adjusting the contrast of the image and distributing the intensities of the image of a specified range.

The input and output intensities can be thought as continuous random variables U, V, with V defined as

$$V = T(U) = (R-1)\int_{0}^{U} h_U(u)\, du$$

where, h_U is the probability density of the input image. Thus T is the CDF of U.

(b) Cumulative-probability-based histogram equalization:

This method also called as the *Bucket fill algorithm.* In this method, the distribution of the intensities of the input image is such that the number of pixels having an intensity is the same. Thus, the histogram of the output image can be defined as:

$$h(i_k) = \frac{N}{I} \quad \forall\, k \in [0 \;\; (I-1)]$$

where $[0 \;\; (I-1)]$ is the intensity range of the input image and N is the total number of pixels. If N/I is not integer, i.e. total number of pixels N is not divisible by highest intensity I, then N/I is rounded off to the nearest integer of higher value. This results in less number of pixels being assigned to one of the intensities values in the range $[0 \;\; (I-1)]$. The histogram of the output image is a flat line.

2.2.2 Image filtering – Creating oil painting effect:

As mentioned earlier the two main attributes of such an *oil painting effect* are *colour gradients* and *colour intensities.* We need to reduce the colour palette of the input image. This process of reducing range of colour intensities to a smaller set is called as *quantization.* We will be implementing the slightly modified version of histogram equalization algorithm to quantize the range of the input colour palette.

To achieve a 64 colour image, we need to quantize each of the three R, G, B input channels to just four intensity levels. The four intensity levels for each channel will depend on the histogram of the respective channel. The algorithm for selecting the four intensities can be stated as:

1. Select one the three channels.
2. Calculate the histogram of the channel.
3. Divide the histogram in four bins with each bin containing $n = \frac{N}{4}$ pixels; where, N is the total number of pixels in the channel.
4. If N is not divisible by four, round off n to the nearest integer of higher value. However, this would result in one of the four bins to have less number of pixels than the others.
   e.g. The first bin of n = 10000 could extend from the intensities *0* to *80* and few pixels under *81.* Then, the next bin could extend from remaining pixels of *81* to 100 and some pixels under intensity *101*. The next bin could extend from remaining pixels of intensity *101* to intensity *200*, and the last bin could contain the remaining pixels.
5. The mean intensity of each bin is calculated. This is the quantized intensity for each bin. Thus, if *L1, L2, L3, L4* are the quantized intensities for each bin, any pixel with intensity in bin 1 will be reassigned *L1* intensity, any pixel with intensity in bin 2 will be reassigned *L2* intensity and so forth.
6. This process must be repeated for each of three channels.

7. Finally, the three quantized channels should be recombined to obtain the 64 colour quantized image.

To achieve 512 colour image, each of the R, G, B channels must be quantized to eight intensity levels. The remaining process for the quantization process will almost remain the same. The histogram must be divided into eight equal bins instead of four.

The next step deals with reducing the degree of detail in the image. This process can be implemented using a simple statistical filter. Consider a *N*x*N* neighbourhood of a pixel. Select the most frequent colour in the neighbourhood as the representative or that pixel.

2.2.3 Image filtering – Creating Film Special Effect:

As mentioned earlier, the *film special effect* can be implemented using *histogram matching*. In this technique, the histogram of the input image is transformed such that it maps the desired histogram (i.e. *Film.raw* image). This can be implemented in the following steps:

1. For the input image h_x find the cumulative histogram H_X using :

$$H_X[k] = \sum_{i=0}^{k} h_x[i]$$

where,

$$h_x(k) = \frac{n_k}{N}$$

2. Similarly, for the *Film.raw* image h_y find the cumulative histogram H_Y using :

$$H_Y[k] = \sum_{i=0}^{k} h_y[i]$$

where,

$$h_y(k) = \frac{n_k}{N}$$

3. Generate a lookup table, i.e. for each input intensity level u find intensity level v so that $H_X[u]$ best matches $H_Y[v]$,by observing the above two cumulative histograms. This can be done by

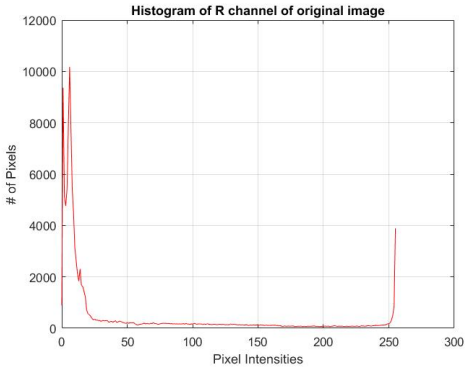
$$LookupTable[u] = \underset{v}{argmin} \{H_X[u] - H_Y[v]\}, \quad \forall v$$

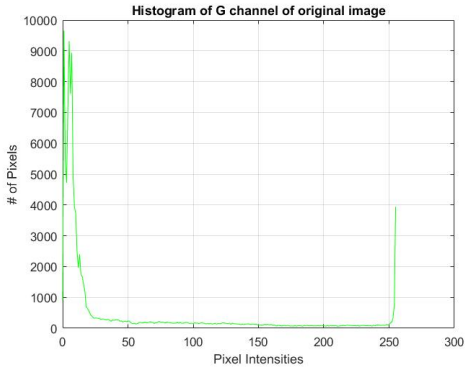
4. Using this *lookup table,* we can map the intensities of the input image to new intensities which will render the *film special effect.*

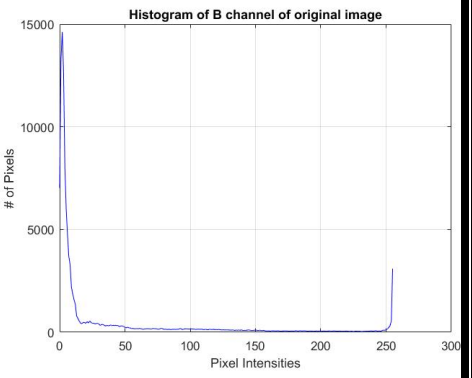
2.3 Experimental Results:

2.3.1 Histogram Equalization:

(a) Original *Desk.raw* image



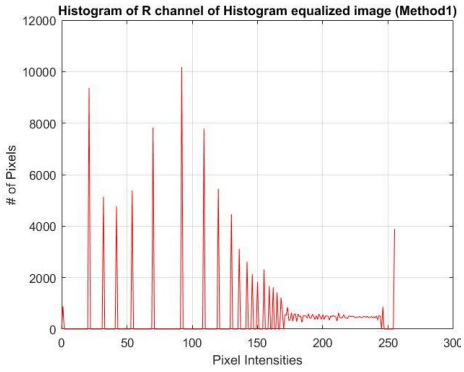




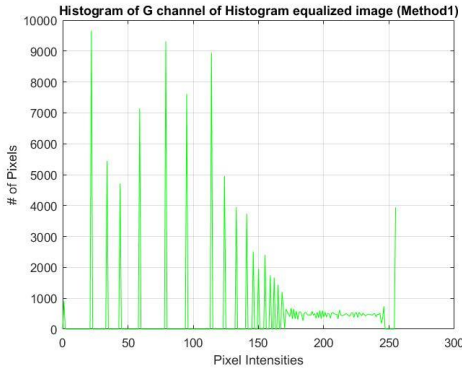
(b) Histogram of R channel (c) Histogram of G channel (d) Histogram of B channel

Figure 7: Histograms of R, G, B channels of the original *Desk.raw* image

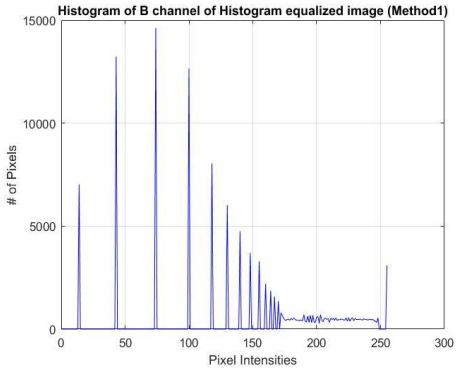
(a) Enhanced image using Method 1 (Transfer function based histogram equalization)



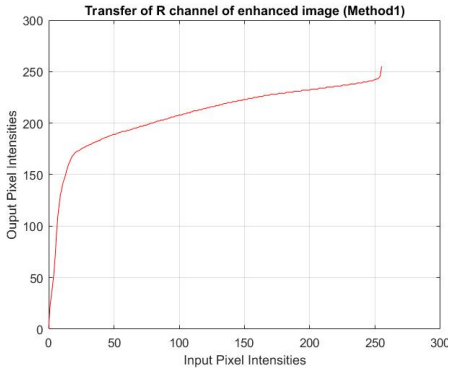
(a) Histogram of R channel of enhanced image



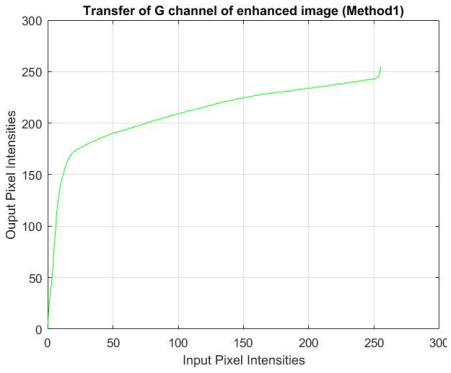
(b) Histogram of G channel of enhanced image



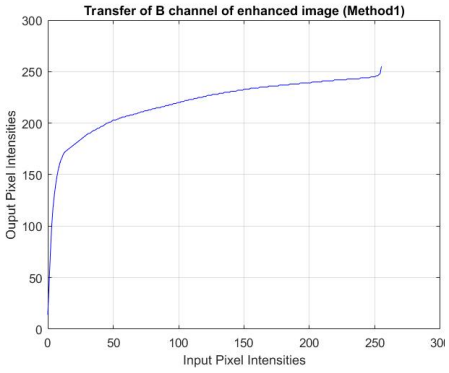
(c) Histogram of B channel of enhanced image



(d) Transfer function of R channel



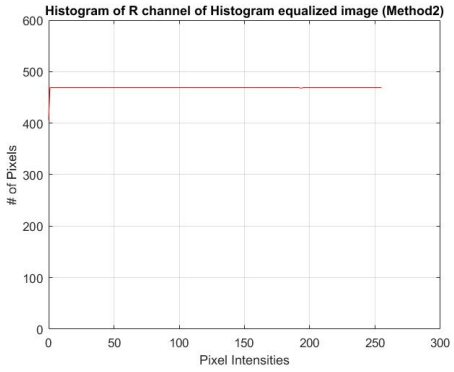
(e) Transfer function of G channel



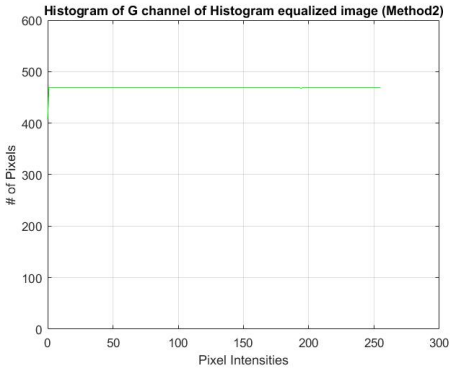
(f) Transfer function of B channel

Figure 8: Enhanced image using transfer function based histogram – equalization

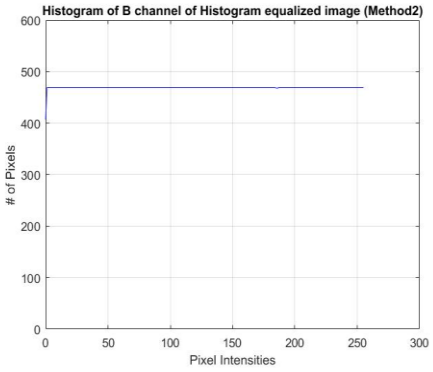
(a) Enhanced image using Method 2 (Transfer function based histogram equalization)



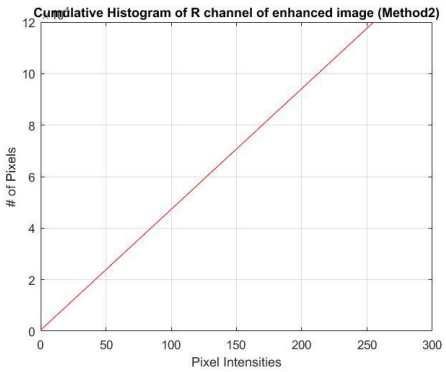
(b) Histogram of R channel of enhanced image



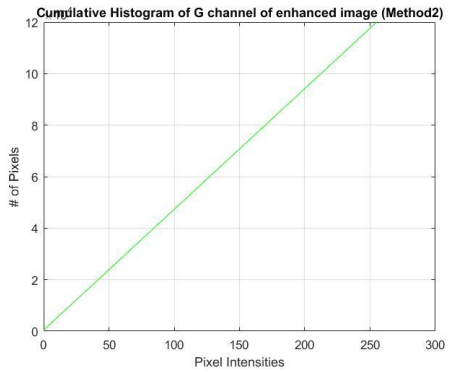
(c) Histogram of R channel of enhanced image



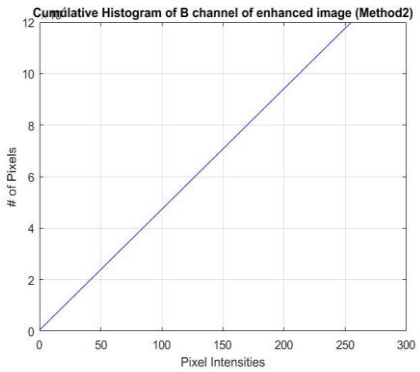
(d) Histogram of R channel of enhanced image



(e) Cumulative histogram of R channel



(f) Cumulative histogram of G channel



(g) Cumulative histogram of B channel

Figure 9: Enhanced image using cumulative-probability-based histogram-equalization

2.3.2 Image Filtering – Creating Oil Painting Effect:

(a) Original *Star_Wars.raw* image

(b) Image quantized to 64 colours

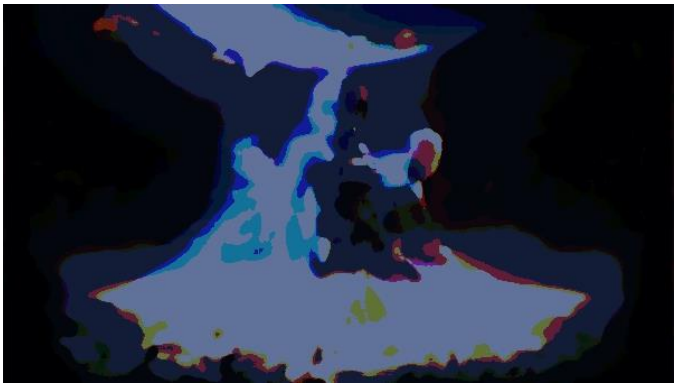
(c) Image quantized to 512 colours

(d) 64 - version image filtered using 3x3 filter

(e) 512 - version image filtered using 3x3 filter

(f) 64 - version image filtered using 5x5 filter

(g) 64 - version image filtered using 7x7 filter



(h) 64 - version image filtered using 9x9 filter

Figure 10: Oil- Painting effect for *Star_Wars.raw* image

(a) Original *Trojans.raw* image

(b) Image quantized to 64 colours

(c) Image quantized to 512 colours

(d) 64 – version filtered using 3x3 filter

(e) 64 – version filtered using 5x5 filter

(f) 64 – version filtered using 7x7 filter

(g) 64 – version filtered using 9x9 filter

(h) 512 – version filtered using 3x3 filter

Figure 11: Oil- Painting effect for *Trojans.raw*

2.3.3 Image Filtering – Creating Film Special Effect:

(a) Original *Girl.raw* image

(b) Final output image after applying the *Film special effect* filter

Figure 12: Image filtering – Creating Film Special Effect

2.4 Discussion:

Histogram Equalization: Upon observing and comparing figure 6 and figure 7, we can say that the transfer-function based approach is slightly better than the cumulative-probability based equalization method. The latter method introduces small patches of odd colour in different section of the image. Histogram equalization works better for images with higher colour depth and lower colour palette. These methods are also useful in improving images which have very high contrast or very low contrast. So, these methods are used mostly in medical imaging applications like X-ray etc. One of the cons of these techniques is that it makes the image patchy and introduces false colours, thus distorting the information. Also, these methods uniformly improve all the intensities in the image. Techniques such as *piecewise stretching* can be used to improve only a range of intensities, leaving all other intensities untouched. For too low contrast images, *log transformation* can be used. *Power law transformation* improves both, too low contrast and too high contrast image, and gives better results than the above two histogram equalization methods.

The above two methods are global equalization techniques, i.e pixels are affected by the transformation based on the intensity distribution of the entire image. But there are cases in which it is necessary to enhance just a small area of the image. Techniques such as *local histogram processing* work by transforming pixel intensities based only on the neighbourhood only.

Image – filtering: Creating Oil Painting Effect: From figure 8, we can observe that the 512 colour version of the *Star_Wars.raw* is better visually than the 64 version colour. Comparing the output images of the 64 colour version for different sizes of filter, we can observe that as the filter size increases, the image looks more like an oil painting. But which output image looks better visually is a matter of opinion. Since the *Stars_Wars.raw* is dominated by black colour, the minute effects of the filter cannot be observed.

As against the *Star_Wars.raw,* the effect of the filter size can be observed in the *Trojan.raw* image quite clearly in figure 9. The effect of the filter size can be seen clearly on the part of the head gear covering the cheeks. As the filter size increases, the details on that part of the headgear diminishes.

There are other ways to implement this *oil painting effect.* Using modern techniques such as *convolutional neural networks,* oil painting effect can be implemented using *style transfer.* A convolutional neural network extracts features from a 'style image' and then adds it to the test image to give it an effect which resembles that of the style image.

Image – filtering: Creating Film Special Effect: It can be observed that the colour space of the original image has been converted from RGB to CMY. Then, this CMY image is mirrored. Finally, we can observe that the histogram of this CMY image is shrunk to get the *Film.raw* image. The *histogram matching* technique provides satisfactory results on the test image for creating the *film special effect*.

For this *film special effect, style transfer* using *convolutional neural networks* may not be able to replicate the filter.

3. Problem 3

3.1 Abstract and Motivation:

Noise removal using point operations is tedious and quite ineffective. Most images features like edges, textures have a highly correlated neighbourhood of pixels. So, to enhance these features and remove noise, point operators do not prove to be helpful. Thus, spatial filters are used to remove noise and enhance image features. When only additive random noise is present in an image, spatial filtering is one the best methods to remove noise

In spatial filters, the two main factors are the neighbourhood of a pixel and a predefined operation which is performed on the neighbourhood pixels. The predefined operation acts on the neighbourhood pixels and produces a new, processed pixel intensity for the pixel under consideration. Different types of spatial filters are used for different types of additive noise.

In recent years, image denoising techniques have really improved and have come along a far way from just spatial filtering or frequency domain filtering. Although these newer paradigms are a computationally sophisticated, they have proven to be very effective.

These new paradigms build on the idea of *self-similarity* that most natural images have. The concept of s*elf-similarity* can be explained as patches (or blocks) of pixels in an image that have a similar neighbourhood, but are not contagious and are located at a distance from each other. And it has been found that processing and denoising such *similar* patches together actually improves the quality of the enhanced image. Initially, these methods were used for image inpainting and texture synthesis. Some these new algorithms are *Non- local means (NLM method), Patch based local PCA, Block matching 3D method (BM3D), BM3D Shape-adaptive PCA* etc.

3.2 Approach and Procedure:

3.2.1 Noise Removal:

In general, linear spatial filtering can defied using the following expression:

$$g(x,y) = \sum_{i=-p}^{p} \sum_{j=-q}^{q} w(i,j)h(x+i,y+j)$$

where, $g(x,y)$ is the output filtered image, $h(x+i,y+j)$ is the input noisy image and $w(i,j)$ are the weights of operation associated with the filter. For mean filter, the weights are $\left(\frac{1}{n}\right)$ where, n is the size of filter. For non-linear filters like median filter, mid-point filter the weights are replaced by order – statistics.

(a) Original Image

(b) Image mixed with different noises

Figure 13: *Lena.raw* image

(a) R channel of Original Image

(b) G channel of Original Image

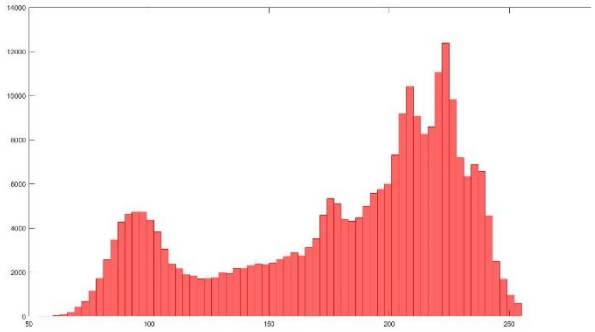
(c) B channel of Original Image

(d) R channel of Noisy Image

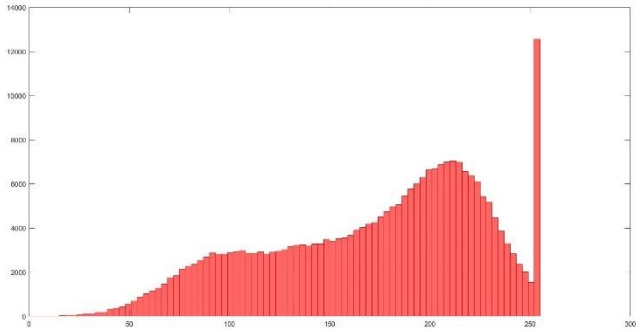
(e) R channel of Noisy Image

(f) R channel of Noisy Image

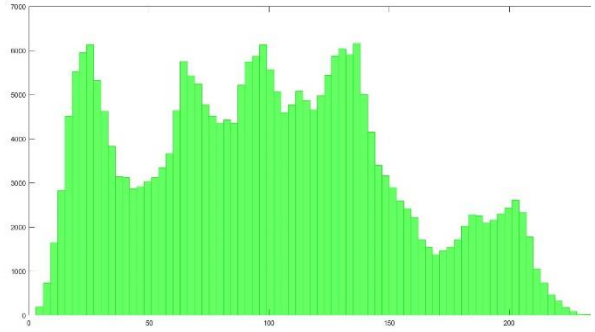
Figure 14: Separated channels for the original and noisy image



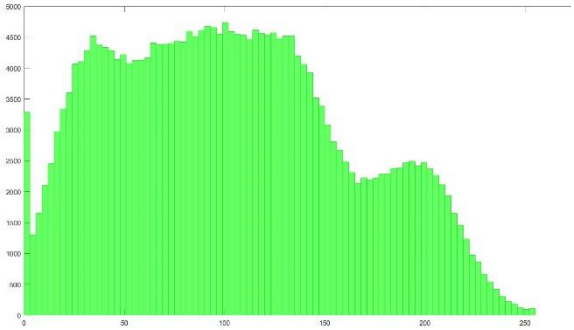
(a) Histogram of R channel of Original Image



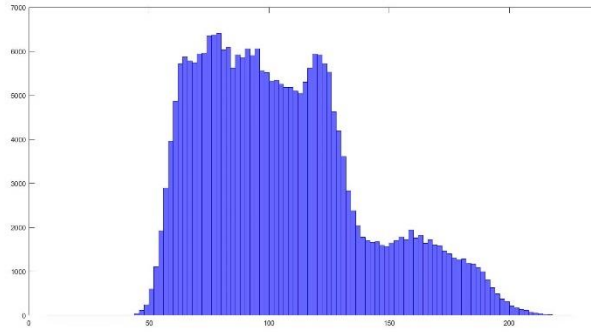
(b) Histogram of R channel of Noisy Image



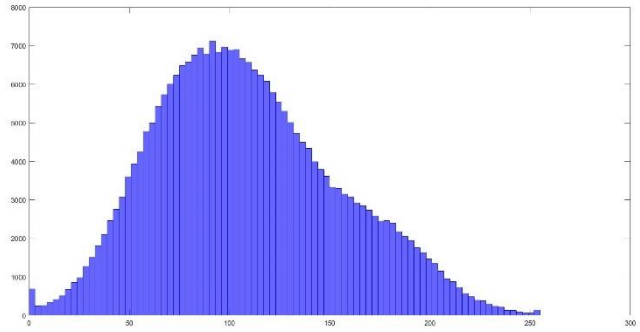
(c) Histogram of G channel of Original Image



(d) Histogram of G channel of Noisy Image



(a) Histogram of B channel of Original Image



(e) Histogram of R channel of Noisy Image

Figure 15. Histograms for each channels of the original and noisy image

As we can observe from the histogram of each plane, the R plane contains mixture of *pepper* noise and *Gaussian* noise, the G plane contains mixture of *salt* noise and *Gaussian* noise, while the B channel contains *Gaussian* noise.

Salt and Pepper noise can be removed using the median filter while the *Gaussian* noise can be filtered using the mean filter.

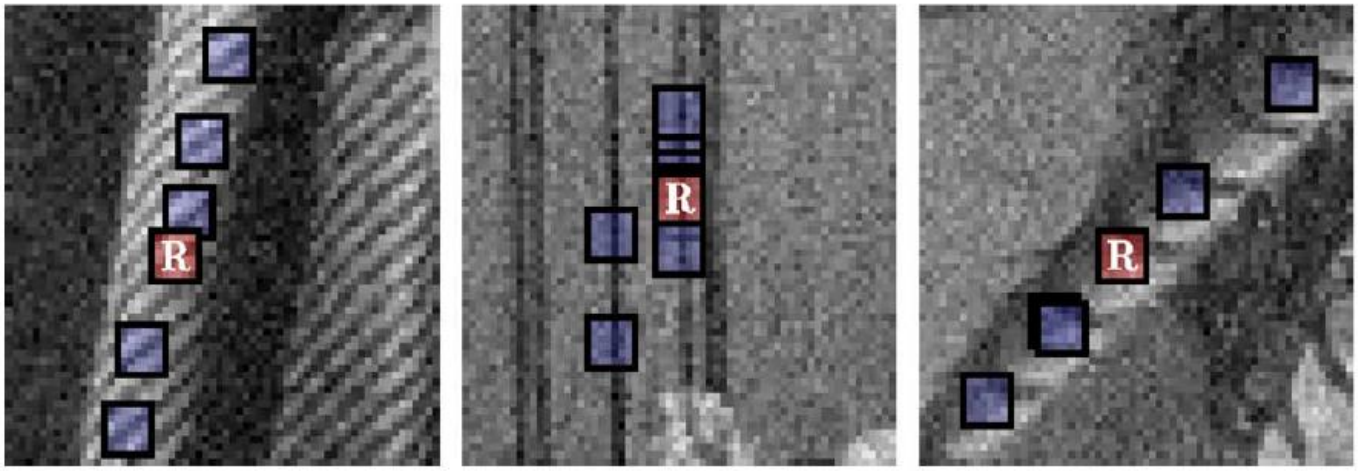
3.2.2 Patch based Principal Component Analysis:

Patch based denoising algorithms work on the idea of patches and capitalizes on the natural *self-similarity* of images. Instead of just considering the nearest neighbourhood of a pixel, other similar patches in the image are found and are used to filter the pixel under consideration

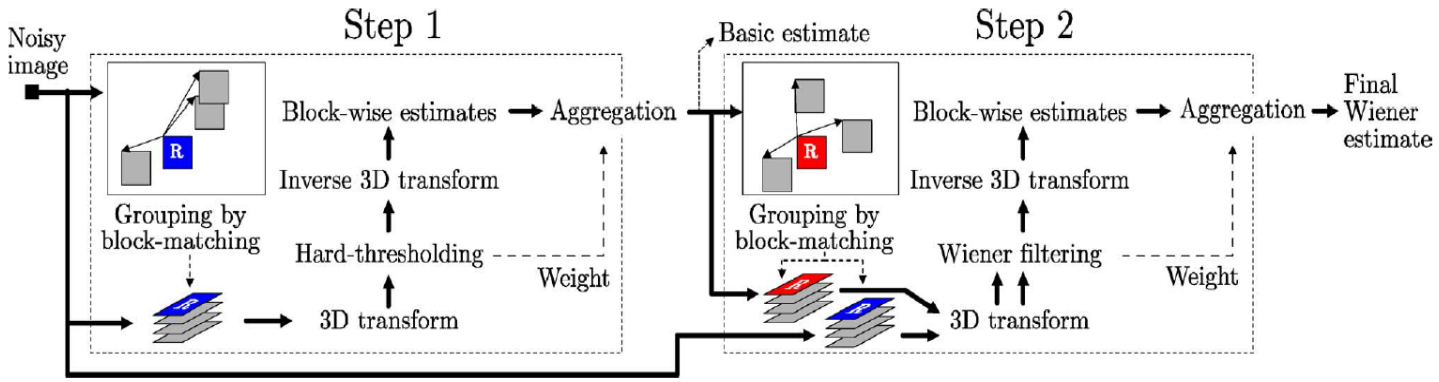
Principal component analysis converts set of possibly correlated data into set of linearly uncorrelated data called principal component. PCA extracts the important information from the dataset and transforms it to the orthogonal principal components. Suppose we have 20 features; in PCA we create 20 new features, where each of the new feature is a combination of the 20 old feature. These new variables are created and ordered in a specific way. The first few principal components are the new features of prime importance.

The Principal Component Analysis consists of the singular value decomposition which yields the eigen vectors (principal component). Patch denoising can be implemented by thresholding to discard the least important principal components. But this results in poor results because noise is uniformly spread in the image. Therefore, it is better to consider all the principal components and instead of thresholding, process each component according to its magnitude.

3.2.3 Block matching and 3-D (BM3D) transform filter:



(a) Grouping blocks from noisy images. Each fragment shows a reference block marked *"R"*



(b) Flowchart of the BM3D denoising algorithm

Figure 16: BM3D Algorithm

- BM3D builds on the idea of patches and it capitalizes, just like *Non- Local Means* filter, on the idea of *self-similarity* that most images have. It introduces *collaborative* filtering
- The red patches are reference patches and the blue ones are some of the patches that are quite like it and can be used to improve the estimate the structure of the reference patch
- Patches like the reference patch are found and they are stacked into a volume (called *group*) that has one dimension more than the image. For example, if the image has three dimensions (i.e. three channels), the dimension of the *group* will be 3+1 = 4
- All patches found in the image are not to be used. There is a threshold for the number of patches to be used and stacked in the *group.* In the MATLAB program, the parameter *tau_match* takes care of this threshold
- The patches are usually sorted by proximity to reference patch along this new artificial dimension
- This denoising exploits intra-patch correlation resulting from smoothness of natural images (this is what all the conventional smoothening or denoising algorithms use). Also, this algorithm exploits the inter-patch correlation resulting from *self-similarity* of natural images. Thus, the idea is to perform this inter-patch and intra-path denoising jointly
- Once the *groups* are assembled, 3D wavelet transform is taken. This changes the domain from spatial to a domain where it is easier to separate the noise. The dimensionality remains the same
- Now, denoise this 'D+1' dimension *group*.
- This is the basis of both Step 1 and Step 2. The denoising filter used in step 1 is a *simple hard thresholding.* When wavelet transform is applied and the domain is changed, the smaller coefficients are the ones that are related to noise are removed by the *hard thresholding.*
- After the denoising is done, inverse wavelet transform is performed to change the *group* back to the spatial image domain.
- The above process is repeated in Step 2, but instead of *hard threshold, Wiener filtering* is used.

3.3 Experimental Results:

3.3.1 Noise Removal:

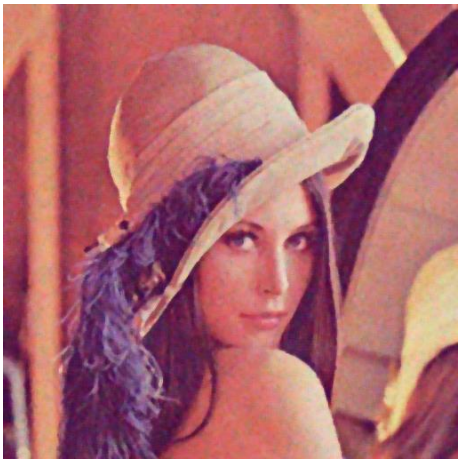


Figure (a)

Figure (b)

Figure (c)

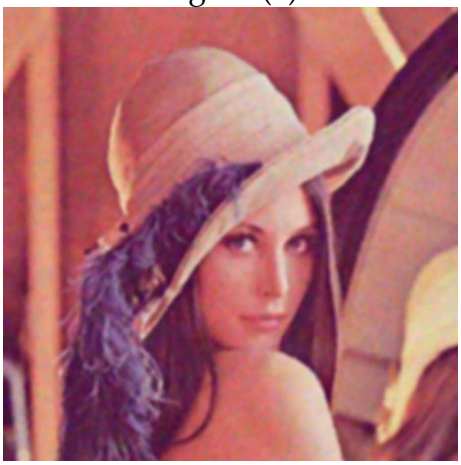


Figure (d)

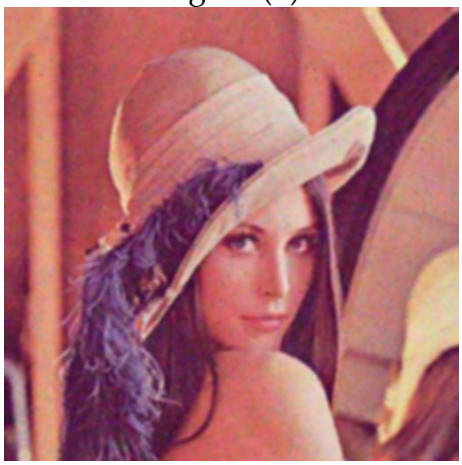


Figure (e)

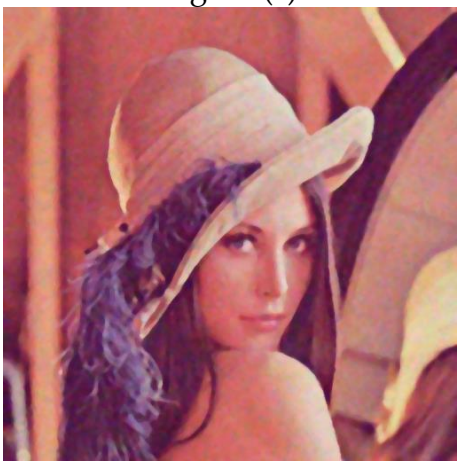


Figure (f)

Figure (g)

Figure (h)

Figure 17: Enhanced Images using filters with different parameters

Figure #	Description of filter	Filter Size	PSNR in R Channel(dB)	PSNR in G Channel(dB)	PSNR in B Channel(dB)	Average PSNR (dB)
a	Median filter for R, G; Mean filter for B	5x5	25.4	27.1	25.3	25.9
b	Median filter for R, G; Mean filter for B	3x3	23.9	26.6	23.9	24.84
c	Median filter for R, G, B; followed by Mean filter for R, G, B	3x3	25.03	27.41	25.55	26
d	Median filter for R, G, B; followed by Mean filter for R, G, B	5x5	25.45	26.24	26.35	26.02
e	Median filter for R, G, B; followed by Mean filter for R, G, B	3x3 for median; 5x5 for mean	25.3	26.52	26.22	26.017
f	Median filter applied twice for R, G, B	5x5	25.9	27.05	26.37	26.44
g	Median filter applied twice for R, G, B	3x3	24.9	27.5	25.06	25.83
h	Median filter applied twice for R, G, B	7x7	25.69	26.05	26.35	26.03

3.3.2 : Patch based Principal Component Analysis:

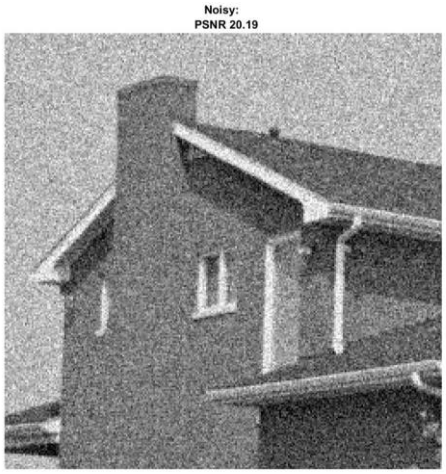


Figure (a) Noisy image with gaussian noise

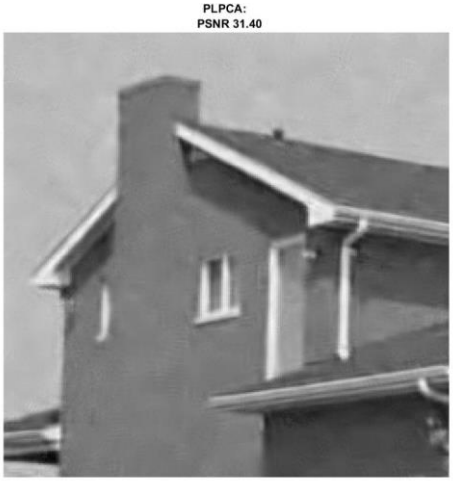


Figure (b)

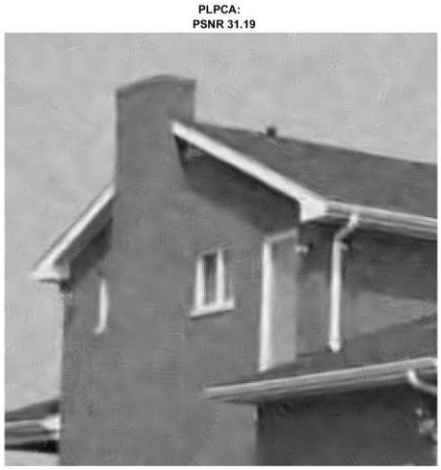


Figure (c)

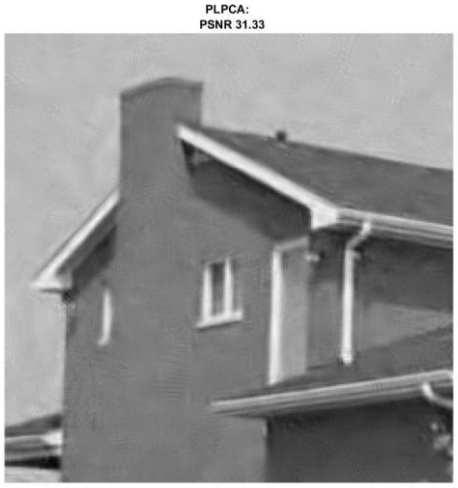


Figure (d)

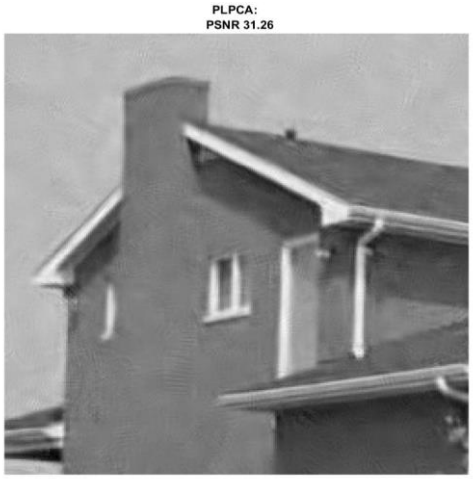


Figure (e)

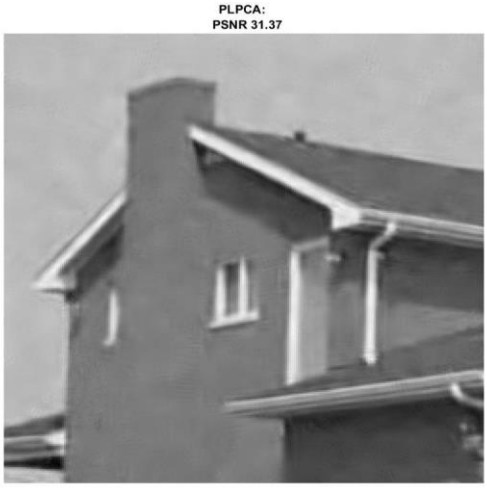
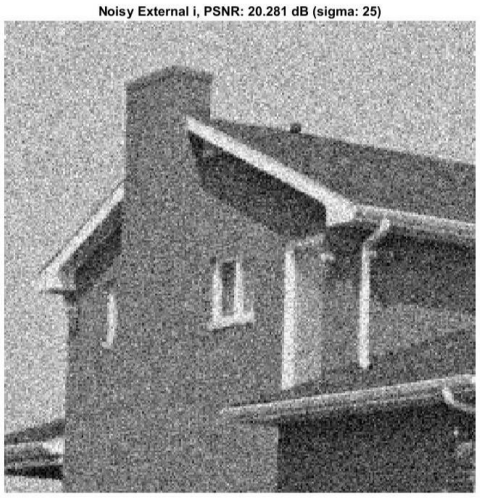


Figure (f)

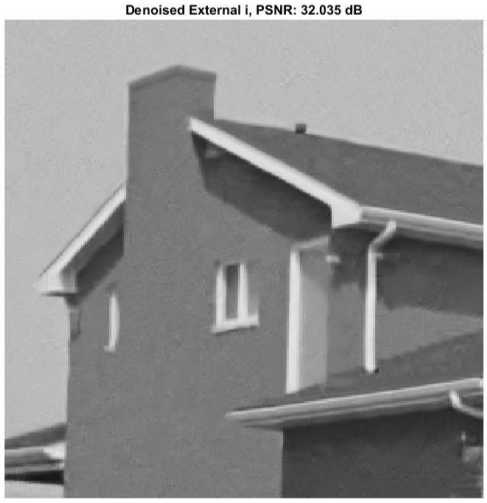
Figure 17: Patch based Principal Component Analysis

Figure number	hW (half size of the searching zone)	hP (half size of the patch)	factor_thr (factor in front of the variance term for hard thresholding the coefficients)	PSNR (dB)
(b)	35	9	2.8	31.4
(c)	12	7	2.75	31.19
(d)	16	9	2.75	31.33
(e)	12	9	2.75	31.26
(f)	25	9	2.95	31.37

3.3.3 : Block matching and 3-D (BM3D) transform filter:



(a) Noisy Image (Gaussian noise $\sigma = 25$)



(b) Denoised Image without using step 2 Wiener filtering

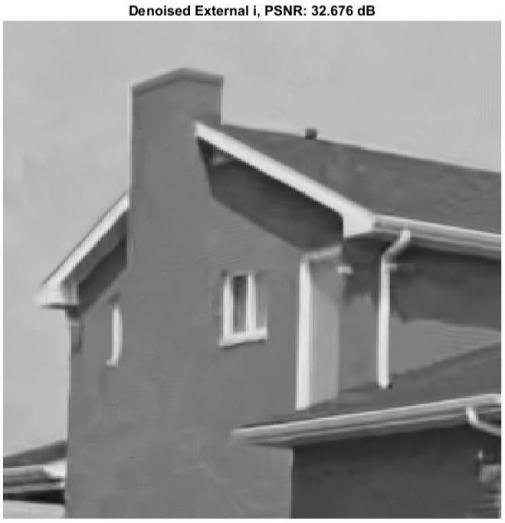


Figure (c)

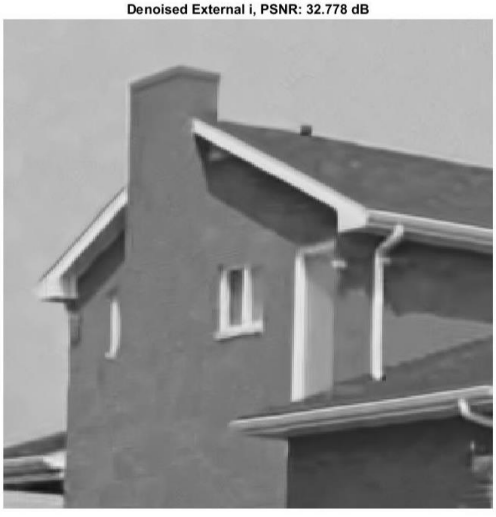


Figure (d)

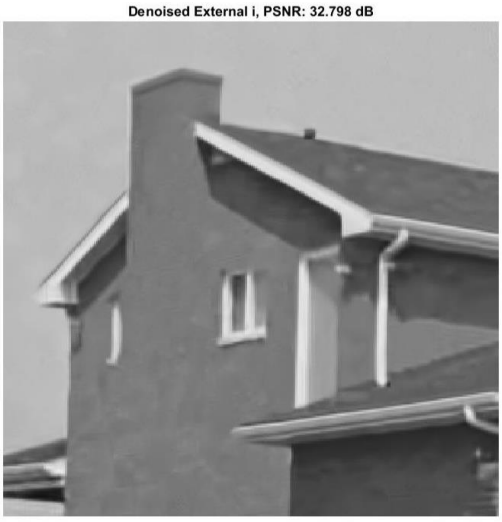


Figure (e)

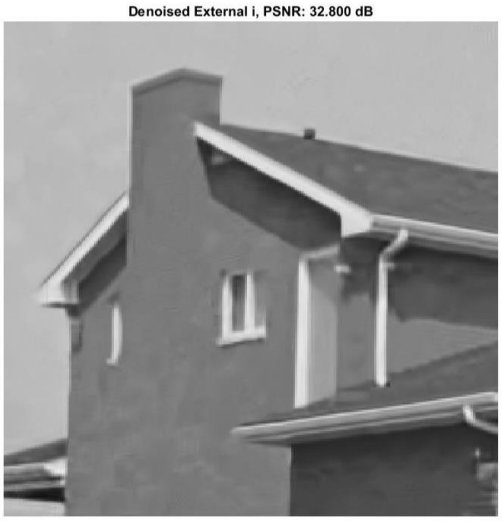


Figure (f)

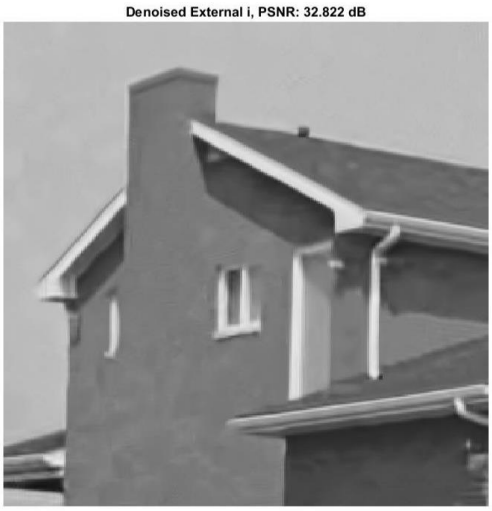


Figure (g)

Figure 18: Block Matching and 3D filter

Figure	Hard Thresholding Parameters			Wiener Parameters			PSNR (dB)
	N1	N2	Ns	N1	N2	Ns	
(c)	8	16	39	8	32	39	32.67
(d)	8	32	99	8	32	99	32.778
(e)	8	32	89	8	32	89	32.798
(f)	8	32	79	8	32	79	32.8
(g)	8	32	75	8	32	75	32.822

N1 is the block – size of thresholding and Wiener filter, N2 is the maximum number of similar blocks and Ns is the length of the search neighbourhood.

3.4 Discussion:

Noise Removal: Median filter performs well in removing both unipolar and bipolar impulse noise. The *max* filter can be used to reduce pepper noise, while *min* filter can be used to reduce salt noise. The mean filter reduces Gaussian noise, but other filters like Bilateral filter, Harmonic mean filter, ContraHarmonic filter perform better in reducing gaussian noise more effectively.

Median filter and similar order-statistic filters must be applied before mean filter. The reason for this cascade is that mean filter distributes the intensities uniformly in the neighbourhood of pixel. Thus, if the mean filter is applied first, it could mask the bipolar noise, making it difficult for the median filter to separate the bipolar noise.

Adaptive filters, Non-local filters, patch based filters perform even better than the above basic spatial. But none of these filters can be used to images that have noise convoluted to the original signal. Such convoluted noise can only be separated using Homomorphic filters

Patch based Principal Component Analysis: Patch based PCA performs better than the simple spatial filtering, mainly because of the patch redundancy property of the images. The patch based PCA transforms the original set of features into new features, where each new feature is a combination of the old features. This makes it easier to separate noise from the image. The thresholding of the principal components helps to remove the noise components

Block matching and 3-D (BM3D) transform filter: BM3D can be defined as both spatial domain filter as well frequency domain filter because, the aggregating of patches in *groups* builds on the spatial redundancy and natural *self-similarity* of images. This is a typical attribute of a spatial filter. The 3D wavelet transform followed by noise removal in the new domain is a typical attribute of a frequency domain filter. It can be observed that without the Wiener filter stage, the PSNR of the filtered image decreases.

References:

1. 'Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering', IEEE transactions on image processing, vol. 16, no. 8
2. 'Image denoising with patch based PCA: local versus global', deledalle, salmon, dalalyan: patch based pca
3. 'Digital Image Processing', Rafael Gonzalez, Richard E. Woods
4. http://fourier.eng.hmc.edu/e161/lectures/contrast_transform/node3.html
5. http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/
6. https://www.math.uci.edu/icamp/courses/math77c/demos/hist_eq.pdf.
7. https://en.wikipedia.org/