EE 569 HOMEWORK 1

GUNJAN JHAWAR
gunjanjh@usc.edu

Contents

1. Problem 1

1.1 Abstract and Motivation

Color spaces define the complete subset of colors in the visible spectrum. Various color spaces exist because they present color information in ways that make certain calculations more convenient or because they provide a way to identify colors that is more intuitive. For example, the RGB color space defines a color as the percentages of red, green, and blue hues mixed together. Other color models describe colors by their hue (green), saturation (dark green), and luminance, or intensity. [1]

To validate the above statement, we see the advantage of using CMY color space in color printers that deposit color in comparison to use RGB. CMY uses subtractive color mixing to reproduce the desired color.

In this problem, we implement different methods: Lightness, Average and Luminosity to convert RGB to Grayscale Image. Also, we convert RGB images to Cyan, Magenta and Yellow channel.

Lightness Method: It averages the most prominent and least prominent colors:(max (R, G, B) + min (R, G,B))/2.

Average Method: It averages the pixel value for all three-color channel: (R, G, B)/3.

Luminosity Method: It performs the weighted average of the pixel values using the formula: 0.21R + 0.72G +0.07B.

Resizing or scaling is the ability to crop images digitally renders the flexibility to remove unnecessary contents from the image and to change the visual appearance of an image, to alter the quantity of information stored in a scene representation, or as a low-level preprocessor in multi-stage image processing chain which operates on features of a scale. The scale operator performs a geometric transformation which can be used to shrink or zoom the size of an image.

When an image needs to be scaled up, each pixel of the original image needs to be moved in a certain direction based on the scale constant. However, when scaling up an image by a non-integral scale factor, there are pixels that are not assigned appropriate pixel values. In this case, those pixel values should be assigned appropriate RGB or grayscale values so that the output image does not have non-valued pixels. [2]

Bilinear interpolation can be used where perfect image transformation with pixel matching is impossible, so that one can calculate and assign appropriate intensity values to pixels.

In this problem, we implement bilinear interpolation to resize an image.

1.2 Approach and Procedures

There are two parts in this section. The methodology applied for color space transformation is composed of two parts: A. Color Space to Gray Scale image conversion and B. Conversion of RGB to CMY color space which is described in 1.2.1 and later the image resizing is done using bilinear interpolation which is explained in 1.2.2.

1.2.1 Color Space Transformation

The raw image with RGB values is stored in a vector container. The pixel value and index are fetched from the vector and computation is done on it and the new pixel value is stored in the output image's vector. Initially the output image is a black monochrome image.

A. RGB to Gray Scale Conversion:

Grayscale conversion is done using three methods:

1. Lightness Method: The pixel value of Red, Green and Blue color channel for every index is fetched using:

$$\text{Index} = \text{row} \times \text{imageWidth} \times \text{bytesPerPixel} + \text{column} \times \text{bytesPerPixel} + \text{color channel}$$

and the maximum of value among the red, green and blue channel is found and similarly is minimum value found. There average value is the new value assigned to the vector for that index.

2. Average Method: The pixel value of Red, Green and Blue color channel for every index is fetched using:

$$\text{Index} = \text{row} \times \text{imageWidth} \times \text{bytesPerPixel} + \text{column} \times \text{bytesPerPixel} + \text{color channel}$$

and the value of red, green and blue channel is found and their average is done.

There average value is the new value assigned to the vector for that index.

3. Luminosity Method: The pixel value of Red, Green and Blue color channel for every index is fetched using:

$$\text{Index} = \text{row} \times \text{imageWidth} \times \text{bytesPerPixel} + \text{column} \times \text{bytesPerPixel} + \text{color channel}$$

and the value of red, green and blue channel is found and their weighted average is done using formula mentioned in abstract is used. There average value is the new value assigned to the vector for that index.

B. RGB to CMY Color Space:

The raw input image with RGB values is stored in a vector container. Firstly, the RGB values are normalized from range [0,1] by dividing by 255. Then the normalized values are subtracted from 1. To assign them back to [0,255] range, they are multiplied by 255. The convert pixel values are in CMY(K) color space. The computation is done individually for each color space. Cyan, magenta and yellow channel is found. The resulting pixel value is stored in the output image's vector.

1.2.2 Image Resizing via Bilinear Interpolation

The idea behind the bilinear interpolation is to first convert the 512*512 image to 1*1 and then resize it 650*650. This is done directly by using a scaling factor (512.0/650.0). Computing every row and column in terms of scaling factor. It gives us the scaled version of every location. Then computing the neighboring locations by adding 1 to the row and column value as we scaled down every location in the range [0,1]. Then computing x difference which is the distance between the two x locations we found. Similarly, finding the y difference. Now, we have all parameters required to use the formula:

$$newPixelValue = (1 - \Delta x)(1 - \Delta y)value(x, y, colorChannel) + (\Delta x)(1 - \Delta y) value(x, y1, colorChannel) + (1 - \Delta x)(\Delta y)value(x1, y, colorChannel) + (\Delta x)(\Delta y)value(x1, y1, colorChannel)$$

1.3 Experimental Results

- **RGB to Gray Scale Conversion:**

Figure 1 Lightness, Average and Luminosity Method

- **RGB to CMY Conversion:**

Figure 2 Cyan, Magenta and Yellow channel on Bear. Raw

Figure 3 Cyan, Magenta and Yellow on Dance. Raw

- **Image Resizing using Bilinear Interpolation:**

*Figure 4 650*650 converted to 512*512 on Airplane. Raw*

1.4 Discussion

The result for the following are described as below:

- **RGB to Gray Scale Conversion:** The average method does not take into consideration that the three colors come from different wavelength and have different contribution in an image but it considers equal contribution of all colors. The result is that the image turns out to be a black image rather than grayscale. The lightness method performs better than average method as it computes using the channel which contributes the maximum and minimum as balances it out. So, it reduces the contrast. The luminosity method takes in consideration that red has more wavelength and green color has less wavelength which in turns gives soothing effect to the eyes. It increases the contribution of green and decreases the contribution of red and blue. The results are shown in Figure 1.

- **RGB to CMY color Space:** Cyan, Magenta and Yellow channel images are shown in Figure 2 and 3 for Bear. Raw and Dance. Raw respectively.

- **Image resizing using Bilinear Interpolation:** Resized and original image is shown in Figure 4 for Airplane. Raw.

2. Problem 2

2.1 Abstract and Motivation

Histogram is a representation of distribution of colors in an image. In case of digital images, it represents the number of pixels that colors have in color range. In transfer function, the objective is to find a transfer function that maps the original pixel distribution to a desired distribution. In cumulative probability based approach, it distributes the pixels equally into 256 bins.

In this problem, we see how these methods allow us to change the pixel value distribution to span a wider range. It in turns helps in contrast enhancement of the image.

In general, images contain 256 intensity levels associated with every channel. If we reduce the number of color intensity level it creates an oil painting effect out of it which creates visually impressive image.

In this problem, we create oil painting effect on the image.

Film effect is created on the image by inverting and scaling the histogram.

2.2 Approach and Procedures

There are three parts in this section. The methodology of Histogram equalization is done using two methods: Transfer function based histogram equalization and Cumulative based histogram equalization which is described in 2.2.1, Image filtering: Creating oil paint effect explained in 2.2.2 and Image filtering: Creating film effect explained in 2.2.3.

2.2.1 Histogram Equalization

Firstly, we plot the histogram of red, green and black channels of original image. The intensity value is represented as x-axis and number of pixels as y-axis. To do this, we create a vector for each color channel. We count the number of times we see a pixel value in the image for that channel. We store the count of pixel value to the index which is as same as the pixel value.

The information of the histogram is stored in a .csv file whose name is asked from the user.

In the transfer function approach, aim is to we use gray levels so that histogram is uniform. The histogram computed in the earlier step is used. We calculate the probability density function of pixel values. We compute the probability of each pixel value. Using the pdf, we compute the cumulative probability density and store it in another vector. To re-normalize it we multiply it by 255.

In cumulative probability based approach, we create a vector that store the pixel location whose value start from 0 to 255. It is done by comparison of pixel value with respect to the pixel value at an index. So, the vector has address based on sorted pixel values. Now, the pixel value is distributed uniformly into bin size $(imageWidth * imageHeight * bytesPerPixel/256)$ which is assigned to the address in the sorted vector.

All equalized channels are concatenated together to form the enhanced image.

2.2.2 Image filtering: Creating oil paint effect

In Step 1, using the histogram computed for the original image, the index value in the histogram is saved in a vector when the total number of count of pixel value reaches $(imageWidth \times imageHeight)/4$. Using the index value stored in the vector, we take the average of the highest and the lowest pixel value and assign it to the value of the bin. Thus, each bin is equalized and 4 bins are created. Thus, converting the input image to an image containing only 64 colors. This computation is also performed when the input image has 512 colors. So, the bin size will become $imageWidth \times imageHeight)/8$.

In Step 2, a window is created which takes the neighboring pixel value of the pixel for whom computation is performed. It uses a kernel $(N \times N)$spans the neighboring values and store them in a vector. Afterwards, the mode of the pixel values is taken and assigned as the new pixel value. This computation is done for all locations in an image. The computation is performed for different values of kernel (N).

2.2.3 Image filtering: Creating film special effect

To achieve film special effect, the following algorithm is used:

a) The pixel value is fetched from (first row, last column) to (first row, first column).
b) Subsequently, values are fetched recursively for each row in the same manner as a). This results in an inverted image.
c) After we get an inverted image, we transform the RGB values to CMY color space. This is done by dividing the pixel value by 255 then subtracting them by 1. Now, bringing them back to the same scale by multiplying the values by 255. We get the image inverted and in CMY color space. But, it is still not the film effect that we want.
d) The next step is to alter the histogram of separate color channels to match the histogram of film effect image.
e) We derive functions for them to map the input histogram to the output histogram.
f) Using the functions, we derived, we insert the input pixel values to get the desired output values.

We get the output image as desired.

2.3 Experimental Results:

- **Histogram Equalization**

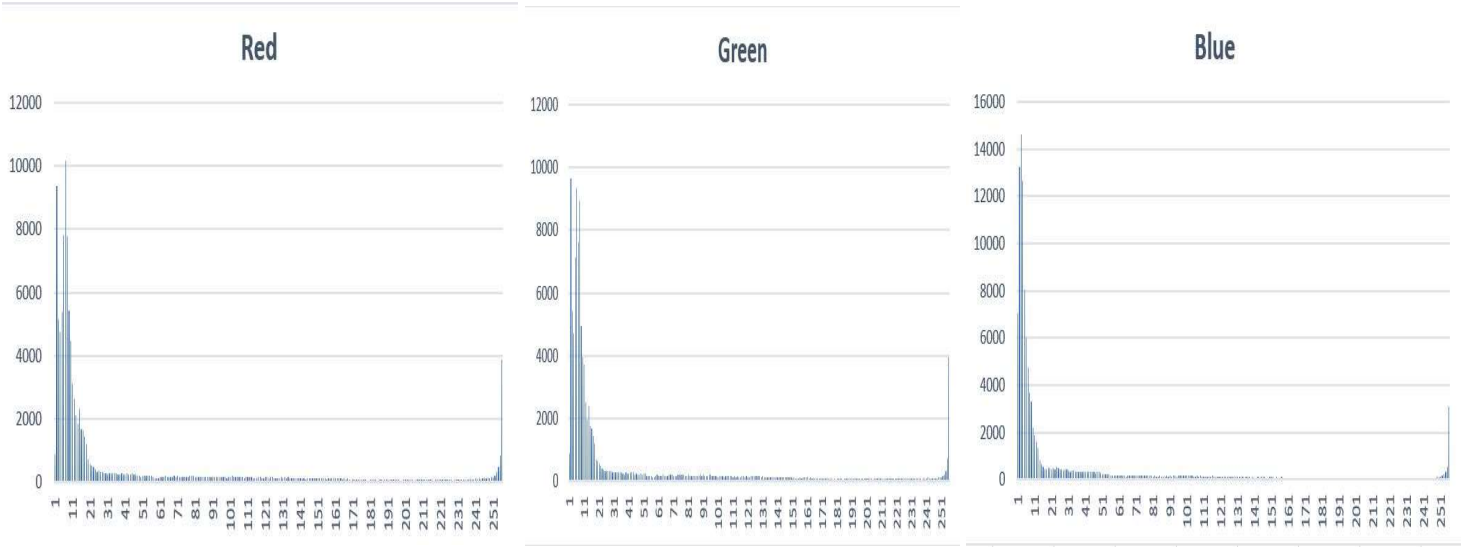

Figure 5 Histogram of each channel in original image

- **Transfer function based Histogram Equalization**

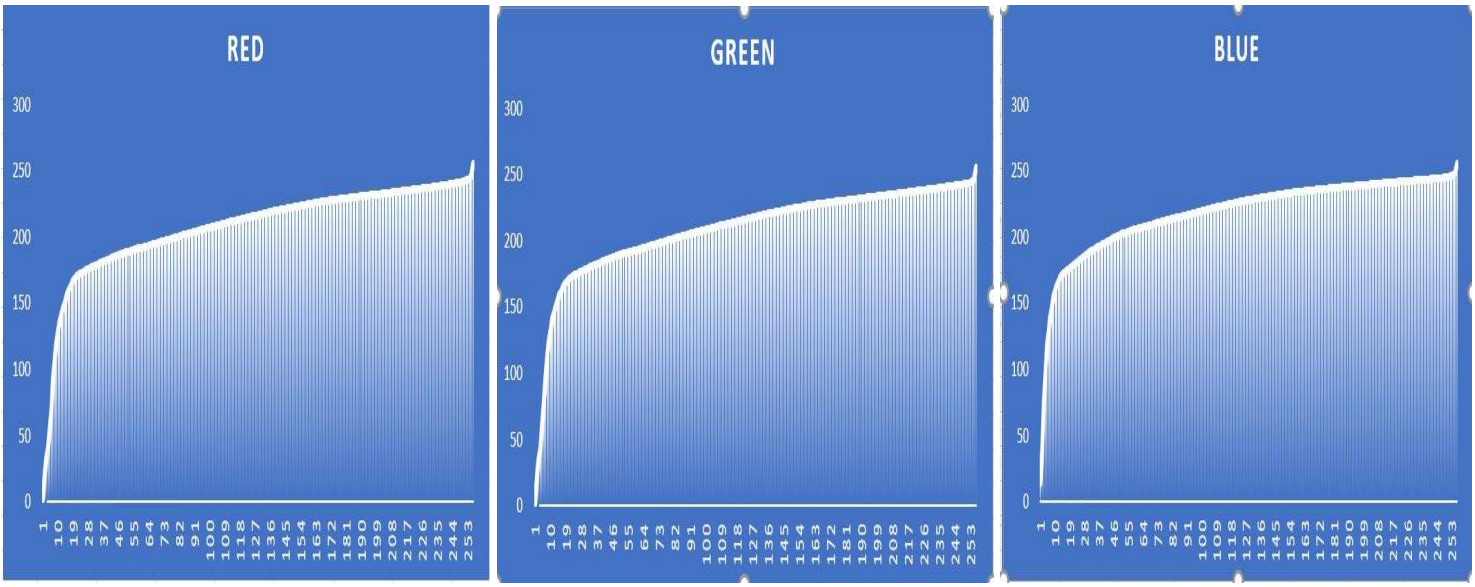

Figure 6 Transfer function based histogram equalization

Figure 7 Enhanced image from Transfer function based histogram equalization

- **Cumulative Probability based Histogram Equalization**

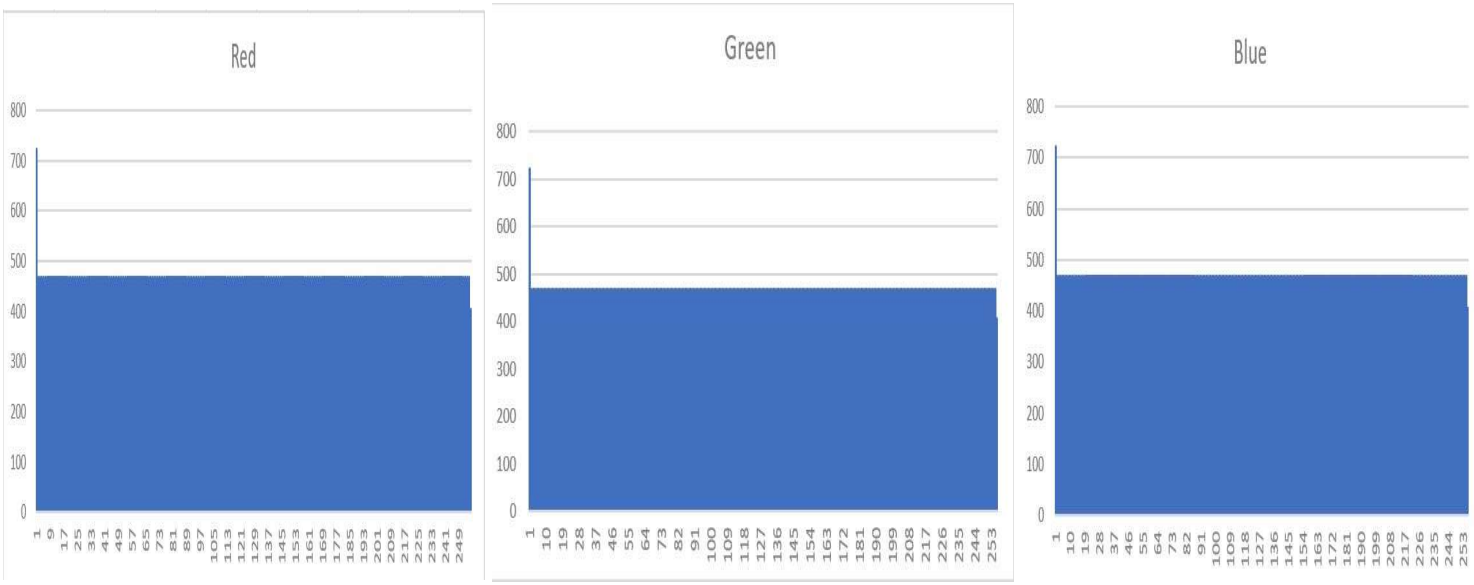

Figure 8 Histogram of each channel for Cumulative probability based histogram equalization

Figure 9 Enhanced image for Cumulative probability based histogram equalization

- **Image Filtering: Creating oil paint effect**

Figure 10 Step 1: 64 Color version of Star_Wars.raw and Trojans.raw

Figure 11 Step 2: N = 3 on Star_Wars.raw

Figure 12 Step 2: N = 5 of Star_Wars.raw

Figure 13 Step 2: N = 7 of Star_Wars.raw

Figure 14 Step 2: 11 of Star_Wars.raw

Figure 15 Step 2: N = 3 of Trojans.raw

Figure 16 Step 2: N = 5 of Trojans.raw

Figure 17 Step 2: N = 7of Trojans.raw

Figure 18 Step 2: N = 9 of Trojans.raw

Figure 19 Step 3 on StarWars.raw

Figure 20 Step 3 on Trojans. Raw

Figure 21 Film effect on girl. Raw

2.4Discussion

The result is described below:

Histogram Equalization: We observe the histogram of the original image to be more concentrated towards 0 which is correct as the image is darker.

In transfer function based histogram equalization, the objective is to improve the range of the pixel values by uniformly distributing them. It is computationally faster. But, we see in the output image that pixels are not utilized. Hence the image is pixelated.

In cumulative function based histogram equalization, all pixel values are accustomed to follow a desired distribution. The output histogram is smooth. The disadvantage of this method is that the mapping is not consistent. It might not

be desirable the way that the pixel values are distributed in the output image. Thus, causing imbalance in the color of the output image. Also, the implementation of the method is slightly expensive as it needs to store pixel location in terms of pixel value.

Both methods don't improve the overall quality of the image. There are other histogram equalization methods that are more precise and taken into consideration different parameters described in [3] like adaptive histogram equalization which uses different part of image computes several histograms, perform computation individually on them and then enhances the image.

Image filtering: Creating oil paint effect:

1) The image consists of 64 colors which implies that each color channel consists of 4 colors. The threshold for each bin is determined by total number of color per channel is required. It decides the number of pixel values in each bin. As soon as the bin reaches total pixel equal to (imageWidth×imageHeight)/4, it shifts to the next bin. In every bin, the pixel value is the average of all pixel value's in the bin.

2) Different value of N = 3,5,7,9 and 11 are used. We observe that as the value of N becomes larger, the details and edges in the image become less and the oil painting effect is dominant. It is because in large window there are large values with the same representation color. Smaller value of N like 3,5 give better result.

3) When we use 512 colors instead of 64 colors, we observe that our image has more details and more colors. It is because rather than 4, 8 colors are assigned to each color channel which gives it more color set to assign.

Image filtering: Creating film effect:

We observe the desired image in Figure 21. We observe that how we can alter histogram of each channel to get functions that maps our input image to output image like a particular example.

3. Problem 3

3.1 Abstract and Motivation

Noise is random variation of brightness or color information in images usually an aspect of electronic noise. It can be produced by the sensor and circuitry of a scanner or digital camera. [4] They deteriorate the image quality. Their removal is essential to maintain the quality of the image and to make them usable for any task. The performance of each filter or set of filters will be based on the Peak Signal to Noise Ratio (PSNR). Lower PSNR values indicate presence of high noise content

In this problem, we study how noise is present in an image. Implementation of techniques for noise removal like implementing mean filter, median filter, cascading them together is done. Also, quantitatively measuring the performance of the denoising algorithm by parameters like PSNR.

Principal component analysis (PCA) is a dimensionality reduction algorithm, yet it can also be used as a tool for noise filtering.
In this problem, it is implemented as a noise filtering method. More information about the method is here. [5].

BM3D is a state-of-the-art denoising algorithm. In this problem, its implementation is explained and performed. More information about the method is here. [6].

3.2Approach and Procedures

There are three parts in this section. The methodology of mix noise in color image done is described in 3.2.1, Principal Component Analysis is explained in 3.2.2 and Block matching and 3D Transform filter is explained in 3.2.3.

3.2.1 Mix noise in color image

The process of applying filter to de- noise the image starts with creating a window of neighboring pixels. The window traverses through each pixel, inputs the neighboring pixel value and computes their mean/ median depending on what filter is used. The output image is composed by the mean/median of neighboring values for that pixel.

Peak signal to noise ratio is computed to assess the performance of the denoising algorithm. Using the formula:

$$PSNR(db) = 10 \log \left(\frac{MAX^2}{MSE} \right)$$

3.2.2 Principal Component Analysis

The procedure is described in this source. [7]
Why PCA can be used to reduce noise?

1) It creates an orthogonal dictionary by learning the orthogonal basis by performing PCA directly on the noisy image and decomposes the noisy patch in the basis. These patches can be global, local and hierarchical. It de noises these patch by zeroing the small coefficient. These orthogonal dictionaries based method performs competitively with respect to the state of art algorithm.

   Component refers to the principal axes of patches extracted from the noisy image. An orthogonal linear transformation projects the data into k dimensions of highest variance. The high eigenvalues signify the high variance. The eigenvectors corresponding to those eigenvalues are selected as the principal component.
   Parameters:
   Factor for thresholding:factor_thr
   Half size of the searching zone: hW
   Shift parameter of searching windows:
   Delta: Step Size

2) Patch Based Local PCA (PLPCA):

- The algorithm performs several PCA's on subsets of patches which have less variation among them. A sliding window is calculated with a step size of Ws-1/2= delta.
- Each sliding window provides a candidate which estimates the true noise free patch. For every pixel, there are multiple candidates as they become a part of several overlapping patches.
- The final patch estimate is the uniform average of candidate estimates for all pixels.

Advantages: The sliding step lessens the computation time by a factor delta^2

Disadvantages: Sliding windows are overlapping which causes redundancy. Also, PCA is computed several times which increases computation time.

3.2.3 Block matching and 3D Transform Filter(BM3D)

BM3D algorithm consists of grouping and collaborative filtering achieved by block matching and shrinkage in 3D domain respectively. The input image which is noisy is processed successively which extracts reference blocks. For each of the reference blocks, similar blocks to it are found and stacked together in 3D array. Then collaborative filtering of the block is done and returned to the original position. As, there are multiple estimates for each pixel in the image. The resulting image is the aggregates of all the estimate.

It is carried in form of two steps:

Basic Estimate:
1) Block wise estimate: For each block: Grouping: Blocks similar to the current one is found and stack together in 3D array.

Collaborative hard thresholding: 3D transform is applied to the formed group in previous step, attenuating the noise by hard thresholding the coefficients. Then inverting the 3D transform to produce estimates for all grouped blocks. Then returning the block with them estimates to their original position.

Aggregation: As we will have block wise estimates that will be overlapping, the value in the true image will be its weighted average.

Final estimate:
Using basic estimate performing improved grouping and collaborative wiener filtering.

Grouping: Finding block locations like the one in basic estimate. Using its locations to form 2 groups, one from noisy and other from basic estimate.

Collaborative wiener filtering: The two groups that we formed in previous step performing 3D transform on it. Performing wiener filtering on the noisy group using energy spectrum of the basic one. Applying 3D inverse transform on the blocks to find the filtered coefficients and return the estimates.

Aggregation: Computing the final estimate by weighting average all local estimates.

b)
We use block matching because grouping by block matching is improved if we use basic estimate instead of noisy image.
Basic estimate as reference for Weiner filtering is more accurate than simple hard thresholding of 3D spectrum of noisy data.

3.3Experimental Results:

Filter	Size(N×N)	Color Channel PSNR(R,G,B)
Mean	3×3	21.4173
		22.3654
		23.3938
Mean	5×5	20.7458
		22.7612
		23.9814
Mean	9×9	18.787
		21.3787

		22.8413
Median	9×9	19.5582
		22.9809
		23.3123
Median	3×3	22.7665
		23.8866
		22.3758
Median	5×5	21.8482
		24.8399
		23.9899
Mean	3×3	22.7672
Median	3×3	23.8862
		22.3761
Median	3×3	21.4316
Mean	3×3	22.3711
		23.4015
Median	3×3	22.7673
Median	3×3	23.8863
		22.3763
		23.5004
Median	3×3	25.6229
Median	5×5	24.5848
Mean	3×3	22.008
Mean	5×5	23.2374
		24.6046
Mean	3×3	21.4306
Mean	3×3	22.3711
		23.4014

Table 1 Mean and Median filter along with PSNR Value

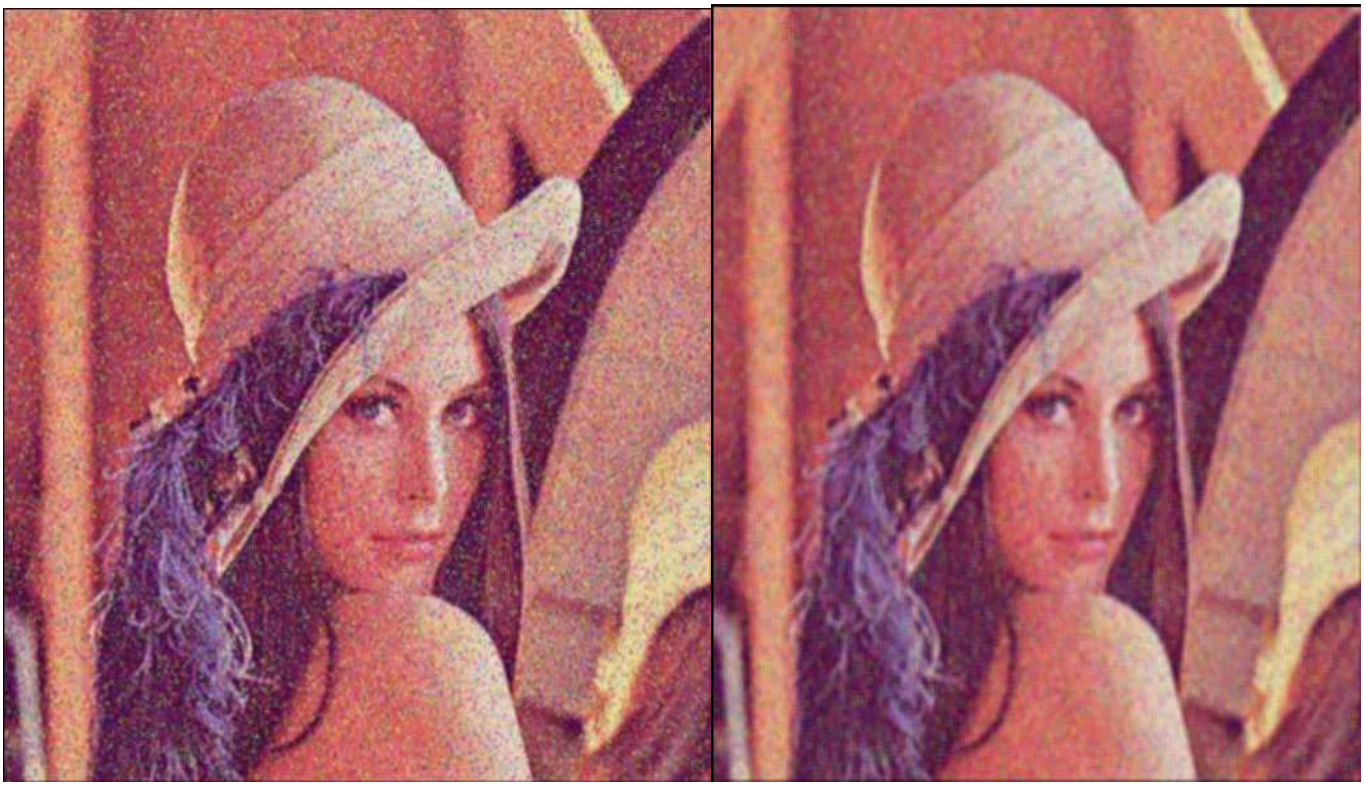

Figure 22 Mean filter of N =3 and N=5

Figure 23 Median filter of N =3 and N=5

Figure 24 Mean with Mean with N =3 and Median with Median filter of N =3

Figure 25 Median with Mean filter of N =3

PSNR	Time	Sigma	delta	hW	Thresholding for variance
39.48db	2.47s	5	7	8	2.5
35.78db	1.74s	10	7	10	2.5
32.42db	1.94s	20	8	11	2.75
31.28db	1.62s	25	8	15	2.75
29.03db	4.32s	40	12	20	3.00

PSNR	Time	Sigma	delta	hW	Thresholding for variance
21.85db	1.50s	25	6	10	1
24.59db	2.04s	25	7	8	1.5
28.19db	1.75s	25	6	11	2
30.49db	1.84s	25	7	9	2.5
31.35db	1.89s	25	8	13	3

Table 2 PCA with different parameters, Yellow block is the best result

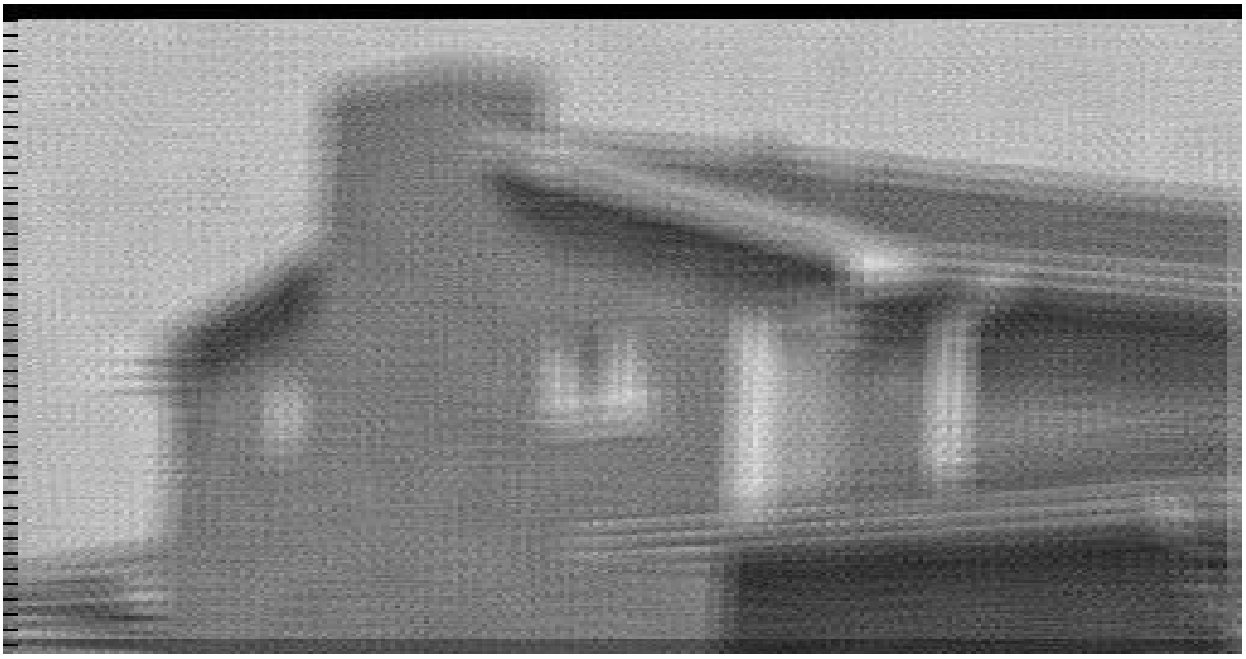

Figure 26 Median then mean filter on House_noisy.raw

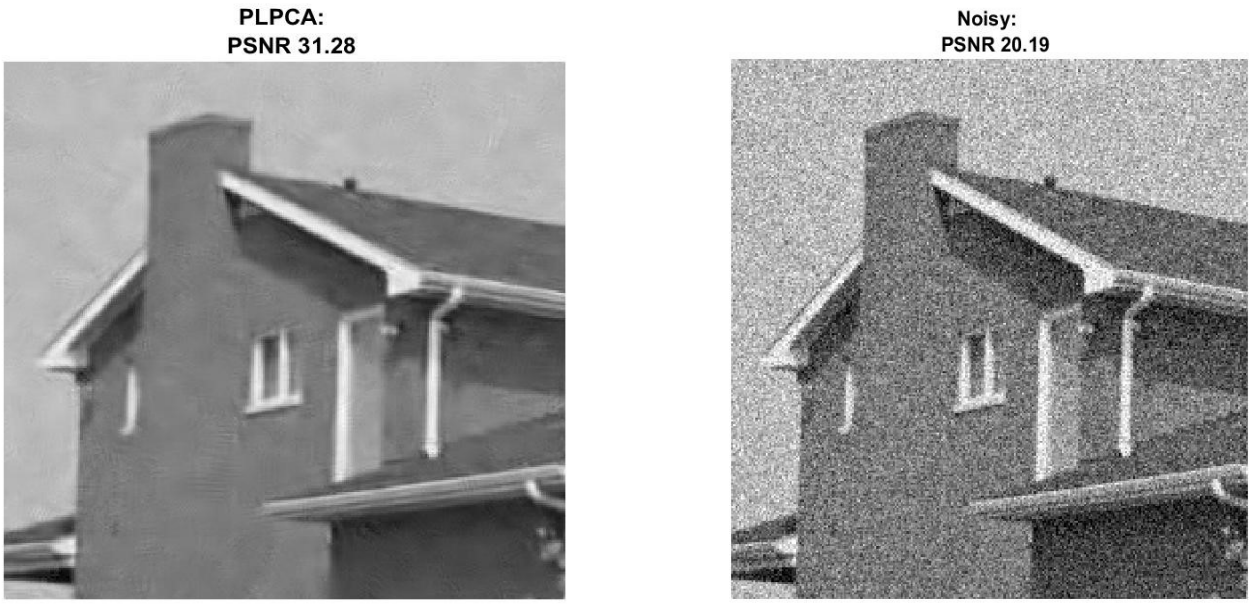

Figure 27 PCA on House_noisy.raw

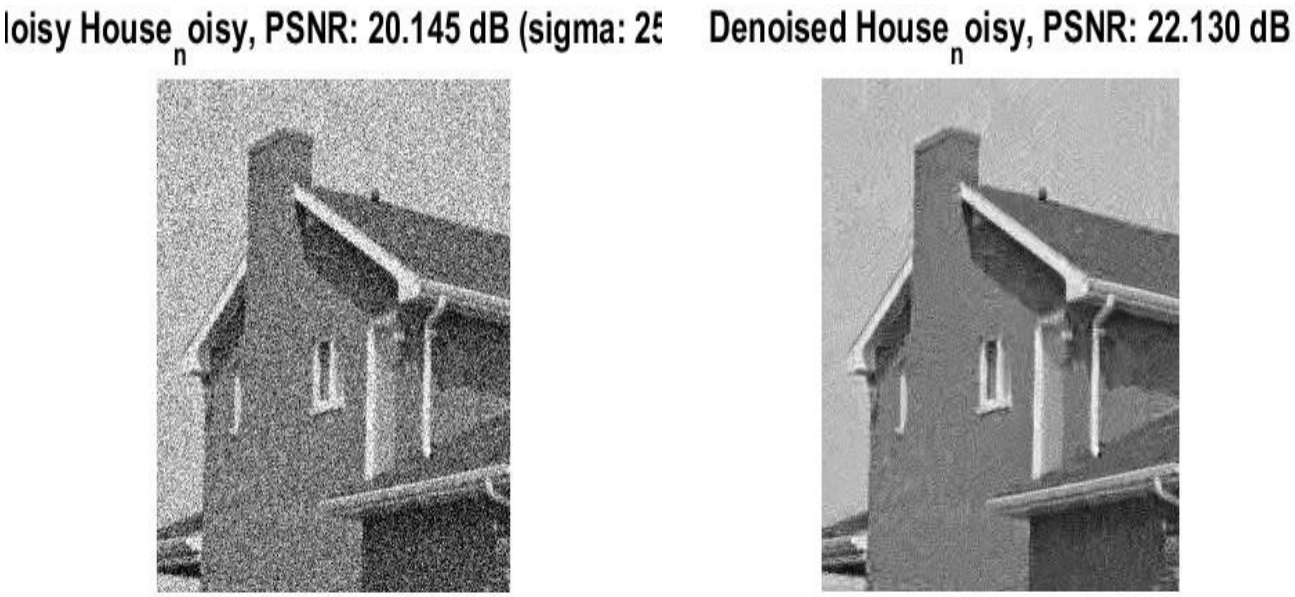

Figure 28 BM3D on House_noisy.raw

3.4Discussion

a)

1.

(1) No, different channels have different noise type. This is proven by them getting different PSNR values.

(2) Yes, filtering should be performed individually as different channels have different noise type. Like red channel is filled with salt noise, green channel with pepper noise and blue channel with salt and pepper noise. The image also contains gaussian and random noise.

(3) Mean and Median filters were used to remove noise.

(4) Yes, filters can be cascaded. Median filters are most popular in eliminating salt and pepper noise. Mean filter is good at removing gaussian noise. If used together, they can help in de- noising the image.

(5) Higher window size give works well but they add a blurring effect to the image. Window size alters the sharpness of the image.

2.

(1) The best method is Median filter with window size of 3 cascaded with mean filter of window size 3. Its result is shown in Figure. It performs well as Median filter works well to remove salt and pepper noise and mean filter removes the gaussian noise not affected by the outliers caused by salt and pepper noise.

(2) The filter does not remove random noise that is present in the image.

(3) Filter like Non- Local Mean can be cascaded with median to remove the random

noise present in the image.

b)

(3) Best parameters for sigma = 25 are marked as yellow in Table. We observe that:

As delta, shift parameter of searching windows, increases our time is increased.

The size of searching zone and threshold helps to make the PSNR better. But large size of searching zone contributes to blurring of the image.

(4) We observe that using the best-found filter in (a) indeed gives blur image for House_noisy.raw.

PCA considers, the maximum variation basis and the small variations are automatically ignored resulting in image denoising. It makes it easier to implement.

PCA transformation matrix trains from local window of the image. It helps in taking account details locally.

It takes care of the subjective details of the image than just quantitatively removing noise as done in median/ mean filter.

c)

(a) The value of sigma decides the block size. If sigma is less than or equal to 40, small block sizes like 8 are used. If sigma is greater than 40, larger block sizes like 11/12 are used.

(b) The block matching takes most of the time of BM3D so if we use it, the algorithm takes more time but also produces better result as compared to when it is not used.

(d) PCA produces highly structed image patterns but it is more computationally expensive as compared to BM3D.

PCA relies on data which has maximum variance while BM3D relies on data which is highly correlated.

4 References

[1]https://www.mathworks.com/help/images/understanding-color-spaces-and-color-space-conversion.html

[2] https://en.wikipedia.org/wiki/Bilinear_interpolation#Application_in_image_processing

[3] http://ijsart.com/Content/PDFDocuments/IJSARTV2I532236359873139316759 67.pdf

[4] https://en.wikipedia.org/wiki/Image_noise.

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

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

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