EE569 – Homework 1 – Feb 4, 2018

A Report on

Basic Image Manipulation

Histogram Equalization

Noise Removal

All the codes were implemented in C++

except for Problem 3 parts B and C where I have used MATLAB code available online

Submitted by:

ANURAG SYAL

MASTERS IN ELECTRICAL ENGINEERING

MULTIMEDIA AND CREATIVE TECHNOLOGIES

USC ID: 9954-6583-64

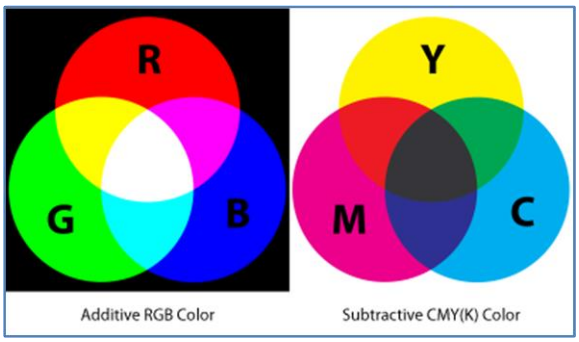
Problem 1: Basic Image Manipulation

Part A: Color Space Manipulation – Grayscale, RGB to CMY
Part B: Bilinear Interpolation

Abstract and Motivation

Color space manipulation is a basic operation of image processing. A color image generally has 3 components. These components can be Red-Green-Blue (RGB), Cyan-Magenta-Yellow (CMY), YUV, etc. Sometimes a color image also has a 4th component which is the Alpha or transparency channel.

Since humans are more perceptive to luminosity than color, it becomes essential to have a function which can change the color space from RGB to Grayscale. Also, when an image is captured from a camera, it is generally in the RGB space. But while printing it, we need to change the space from RGB to CMY because most of the printers operate in that space. Converting to Grayscale also helps us analyzing the intensity level of each channel. Which means, darker regions of a channel's image indicate more intensity of that channel in those areas.



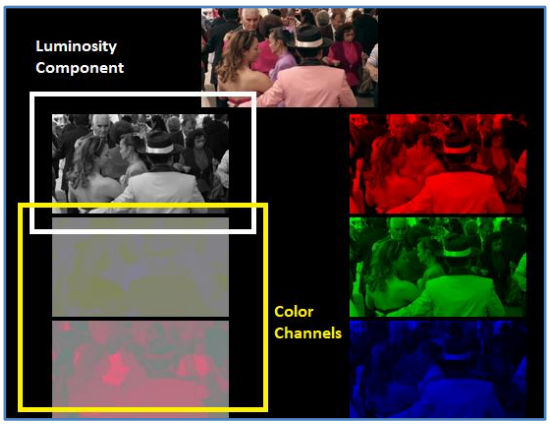


Figure 1: a) RGB and CMY Spaces b) RGB and YUV

Bilinear Interpolation is a technique to resize an image. When resizing an image, sometimes the pixels are left empty. Now in normal scaling algorithms the pixels are copied over. This produces several artefacts in the image and the image overall looks blocky. On the other hand, using Bilinear Interpolation, one can fill the empty pixels by interpolating or superimposing the values of nearly pixels.

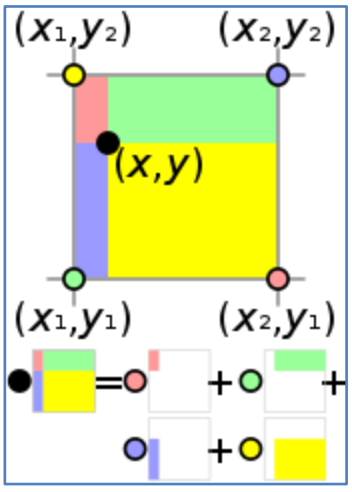


Figure 2: Example of Bilinear Interpolation

Approach and Procedures

RGB to Grayscale

There are 3 types of grayscale operations as described below:

a) Lightness Method -> Pixel Value = [max (R, G, B) + min (R, G, B)]/2
b) Averaging Method -> Pixel Value = [R + G + B]/3
c) Luminosity Method -> Pixel Value = 0.21*R + 0.72*G + 0.07*B

For implementing the above functions, a program in C++ was implemented. Following is the pseudocode for the algorithm:

1. User enters the image dimensions
2. User also enters the method by which grayscale operation must be performed
3. Read the image byte-by-byte in an array
4. Store each channel in separate vectors using array above
5. Grayscale operations are performed on the channel-vectors and a single channel vector is created which stores the grayscale values
6. Result vector is written to a file

RGB to CMY

There are 3 types of grayscale operations as described below:

Pixel Value C = 255*(1 - R)
Pixel Value M = 255*(1 - G)
Pixel Value Y = 255*(1 - B)
R, G and B are normalized values in the range of (0 - 1)

For implementing the above functions, a program in C++ was implemented. Following is the pseudocode for the algorithm:

1. User enters the image dimensions
2. Read the image byte-by-byte in an array
3. Store each channel in separate vectors using array above
4. RGB to CMY operation is performed on individual channel-vectors and three new channels for each of C, M and Y are created
5. Result vector is written to a file

Bilinear Interpolation

For implementing the above functions, a program in C++ was implemented. Following is the pseudocode for the algorithm:

1. User enters the image dimensions
2. Read the image byte-by-byte in an array
3. Store each channel in separate vectors using array above
4. New pixels are generated using Bilinear Interpolation
5. Result vector is written to a file

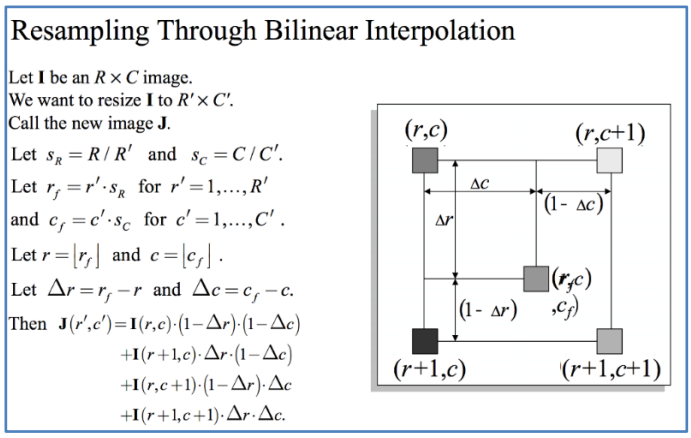


Figure 3: The maths behind Bilinear Interpolation (Source: Ryan Phan)

Results and Discussion

RGB to Grayscale

Figure 4: Tiffany - Grayscale – a) Lightness b) Average c) Luminosity

1) Image produced by **Lightness** method appears washed out. The shine of the cheeks, hands nails are not visible clearly. The whole face appears a bit greyish. The hair near the forehead appear dull.
2) Image produced by **Average** method appears the best among the 3. In comparison with the features of the image produced by Lightness method are somewhat preserved. The cheek lines near the mouth and the appearance of the chin are more pronounced. The hands are shinier too.
3) Image produced by **Luminosity** is the darkest image among all. While the features are little bit more preserved than Lightness method, the overall darkness of the image makes it tough to judge the image. The shadows in the image are more distinctly visible than all the other methods.

RGB to CMY

Bear.raw

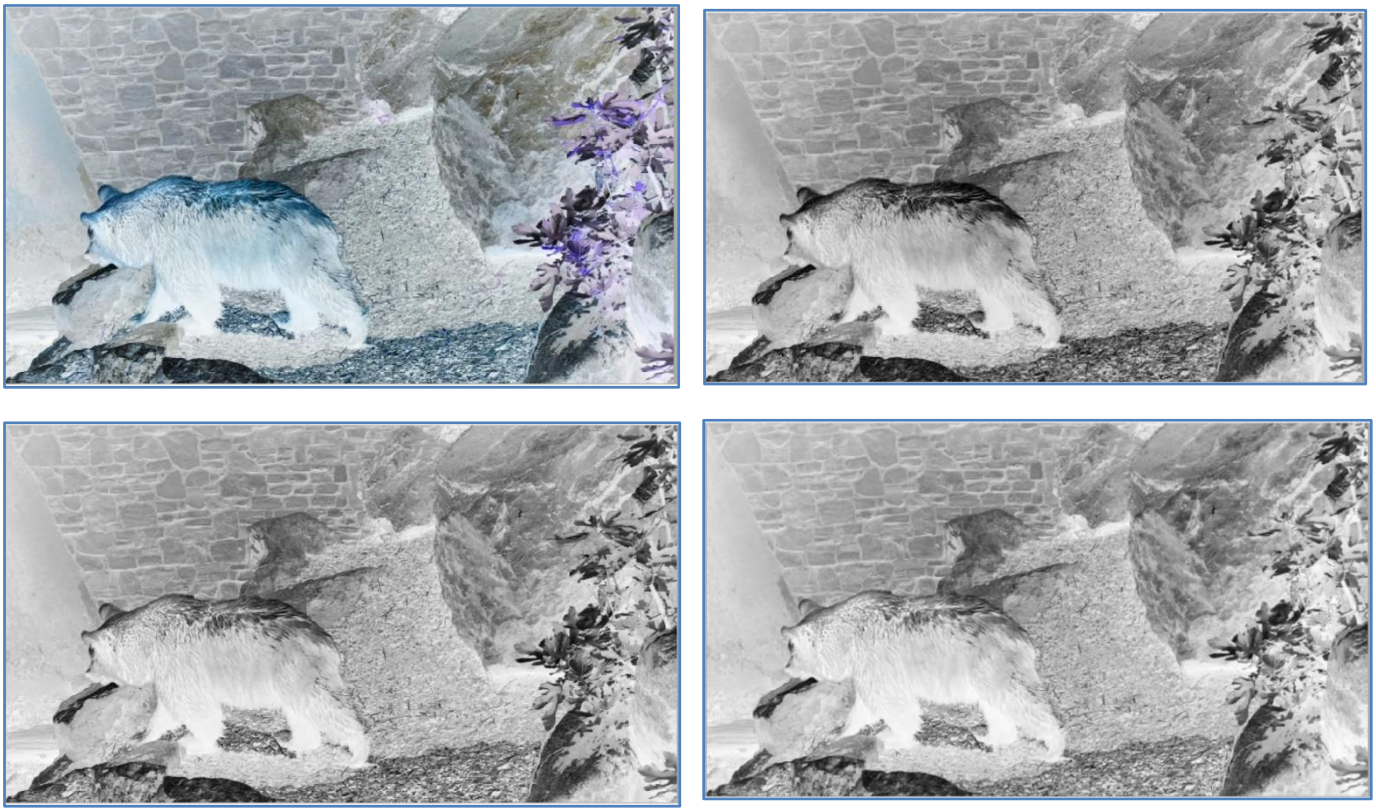


Figure 5: a) RGB to CMY for Bear.raw b) CYAN channel for converted image

Observations

1) CMY image of Bear.raw shows presence of CYAN colour around the back of the bear. It can be seen in the CYAN channel that the intensity is more around bear's back as the region is darker.
2) Yellow channel has the least overall intensity among all the channels. This is seen in the CMY image as well. Since yellow is a combination of Red and Green. We can see in the original image that green colour is not much used in the image.

Figure 6: Original Bear.raw

Dance.raw

Figure 7: a) RGB to CMY for Dance.raw b) CYAN channel for converted image
c) MAGENTA channel for converted image d) YELLOW channel for converted image

Observation

1) CMY image of Dance.raw shows presence of almost all the colours. The overall image appears negative of the original image.
2) CYAN channel has the most overall intensity as the channel image is darkest amongst all the channels.
3) Since the original image contains regions of all the colours.
4) The features of the road are most clearly visible in the Magenta channel.

Figure 8: Original Dance.raw

Bilinear Interpolation

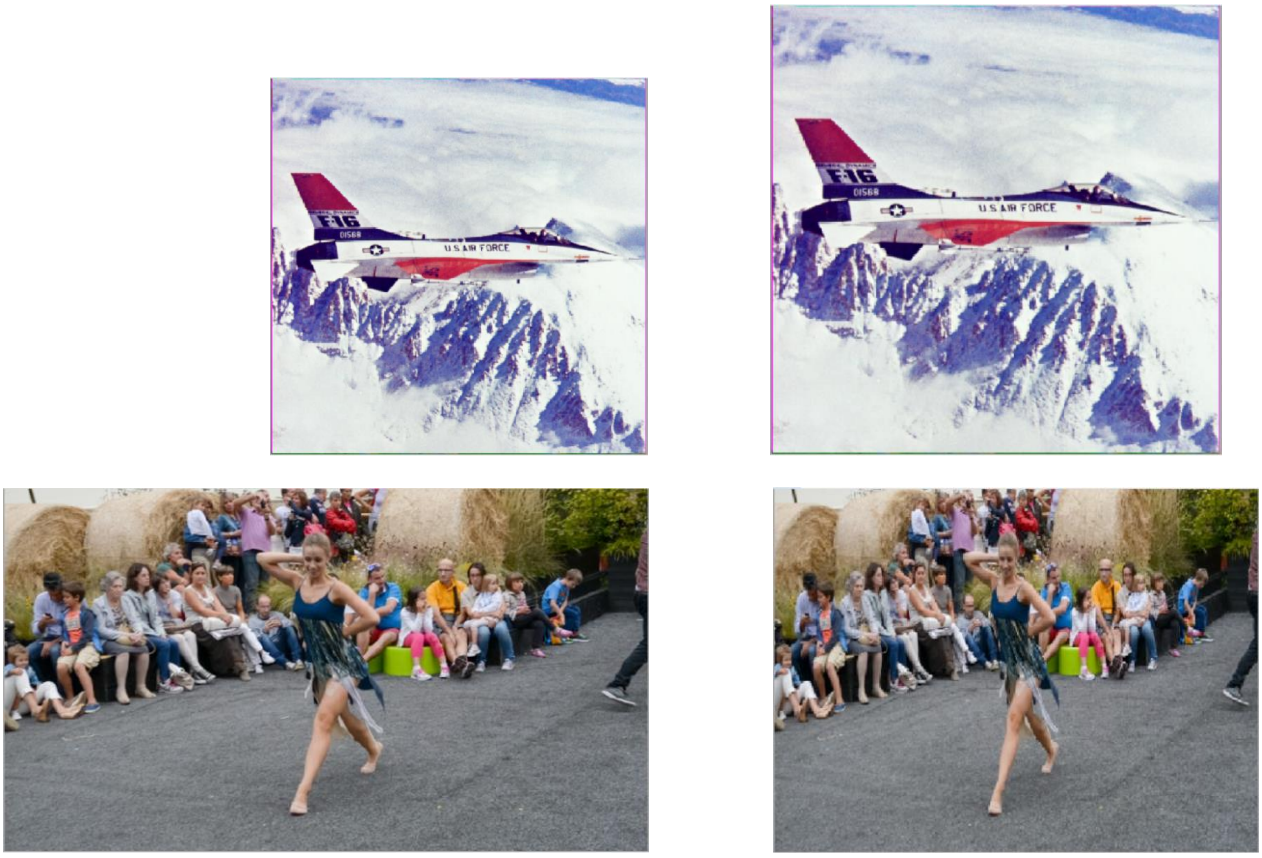


Figure 9: a) Airplane.raw at 512 x 512 and 650 x 650
b) Dance.raw at 854 x 480 and 400 x 300

Observations

1) Both Airplane.raw and Dance.raw have been altered. Airplane.raw has been increased in size while Dance.raw has been reduced.
2) Resizing artefacts can be seen in the altered Dance.raw very clearly as the overall size is reduced. The image appears pixelated overall.
3) The altered Airplane image appears almost perfect with very little blurriness.

Conclusion

The basic tools for image analysis and manipulation have been created. The various grayscale methods help is understanding the image texture. The Grayscale methods also reveal which color contributes the most in the image composition. Bilinear interpolation is a handy technique to scale the image. Using this method, the width and height of any image can be scaled up or down. Up to a certain level, the image can be scaled with no artefacts, but when the size is increased below 50% of the original or more than 200% of the original, pixilation appears.

Problem 2: Histogram Equalization

Part A: Basic Histogram Equalization a) Transfer function b) Bucket Filling

Abstract and Motivation

Histogram Equalization increases the global contrast of images in general. Through this process, the intensities can be better distributed on the histogram. In general, it can happen that the image's histogram is concentrated around a certain value. This makes the image look dull and sometimes even dark, if many pixels are present towards the lower intensity levels. The equalization helps spreading the number of pixels across different intensity values which results in areas of lower local contrast to gain a higher contrast.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. The method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a straightforward technique and an invertible operator. So, in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

There are 2 methods for histogram equalization that have been explored.

1) Equalization using CDF transfer function
2) Bucket-filling algorithm

Approach and Procedures

CDF based method

The following steps were taken:

1. Histograms for each of the channels were created
2. Cumulative-probability Distributed Function (CDF) was created for each histogram. For that, the probability number of pixels were added in a cumulative fashion to the count of the previous pixels.
3. Finding new boundaries: In histogram equalization, the new histogram will have a boundary such that all the values below the histogram are zero. But this does not happen usually. A general equalization formula:

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - 1} \times (L - 1)\right)$$

4. Assigning value to pixel: Final value is multiplied by 255 since values of pixels range from 0-255

Bucket-filling based method

The following steps were taken:

1. Histograms for each of the channels were created
2. For Bucket Filling:
   a) Average pixel intensity = Total number of Pixels/255

b) An array of size 256 x 256 is created. Suppose if a pixel value has an intensity level lower than the average pixel then no pixel of that intensity is changed. But if the intensity level is higher than the average, then it is checked that which bucket needs the pixel first. Once the bucket is filled, the next bucket is decided and the pixel is changed to that bucket's value.

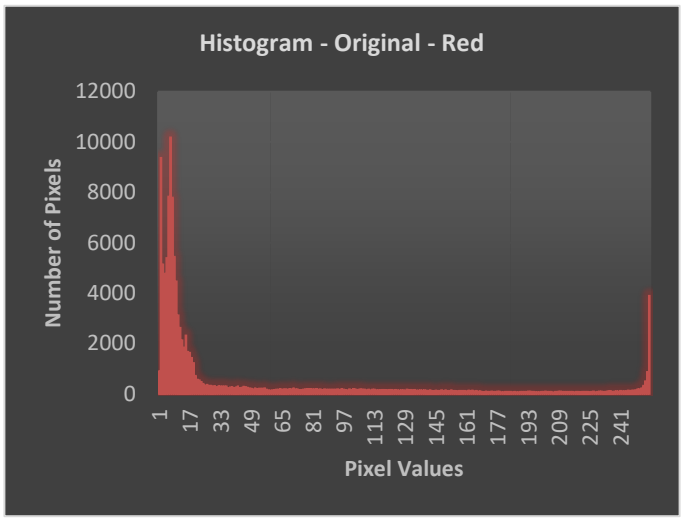
c) For example, size of image is 400x300, then total pixels = 120,000. So, each intensity level should have 120,000/255 = 469 pixels. Suppose number of pixels of intensity level 0 are 7000 and number of pixels in intensity level 1, 2 and 3 are 200, 500 and 100, respectively.

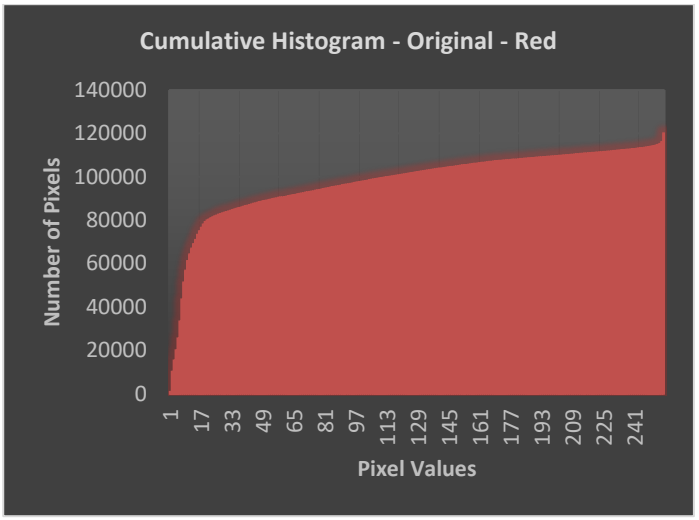
d) Now the image is scanned from the start, if suppose a pixel having value 1 is encountered, then no change is done to it. The program moves forward, now if the pixel encountered is of value 0.

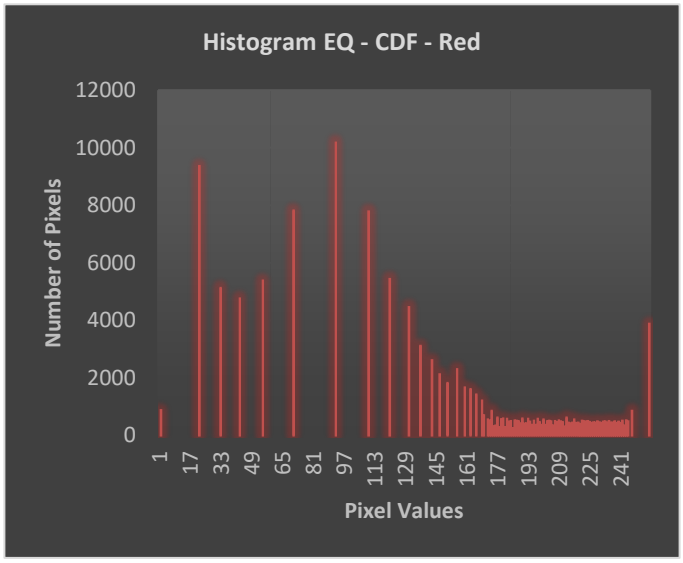
e) Since we know that there are 7,000 pixels with intensity 0, and the first bucket to be filled is of intensity 1 so the current pixel is changed to 1. Now the count of pixels of intensity 0 is 6,999 and count of pixels of intensity 1 are 201. This is done for all the pixels in the image.

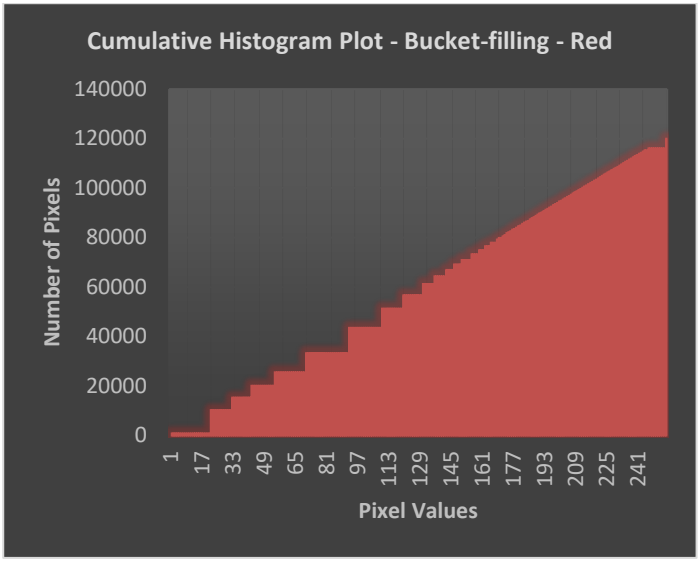
Results and Discussion

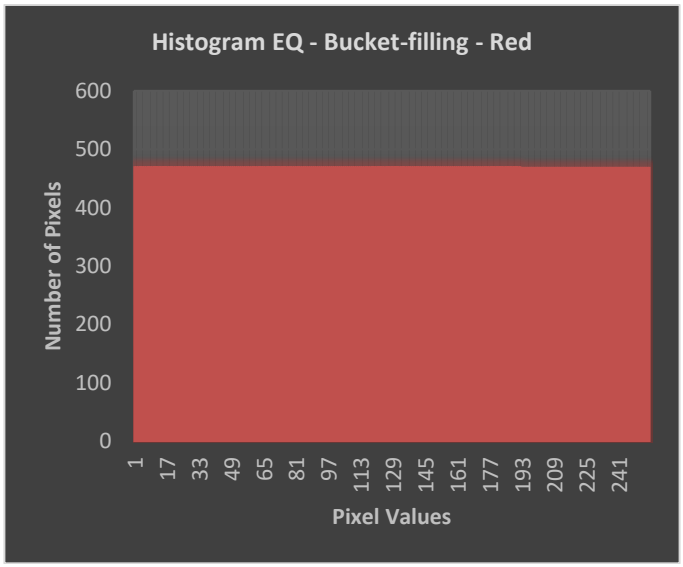
Figure 1: a) Original Desk Image b) CDF (Transfer Function) Desk Image c) Bucket-Filling Desk Image











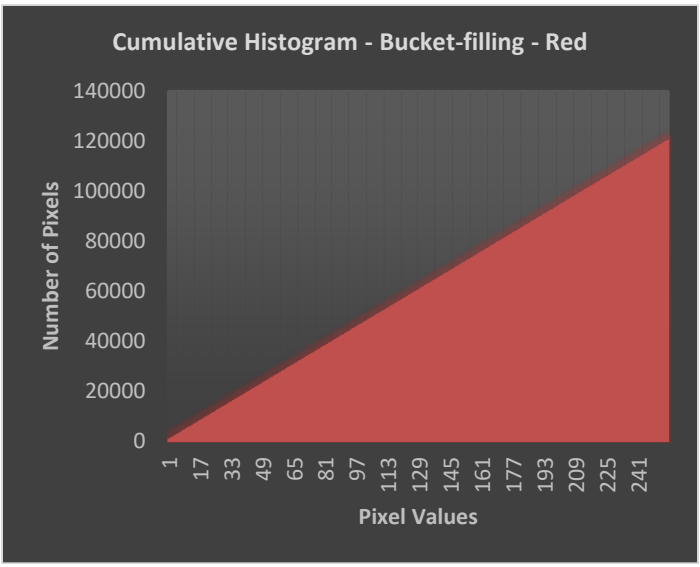


Figure 2: Histogram for RED channel of Desk.raw
a) Original b) CDF c) Bucket-filling d) CDF plot for Bucket Filling

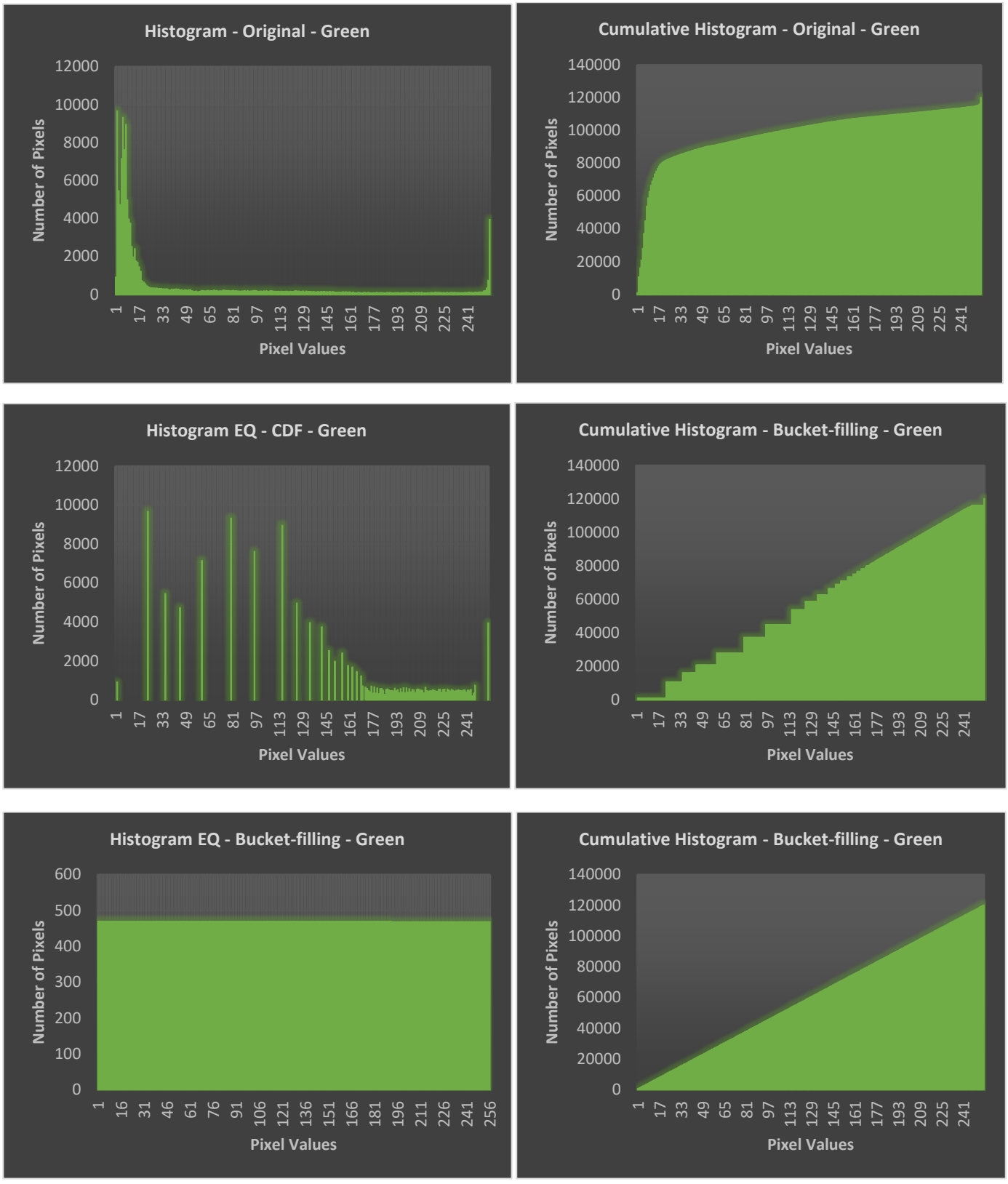
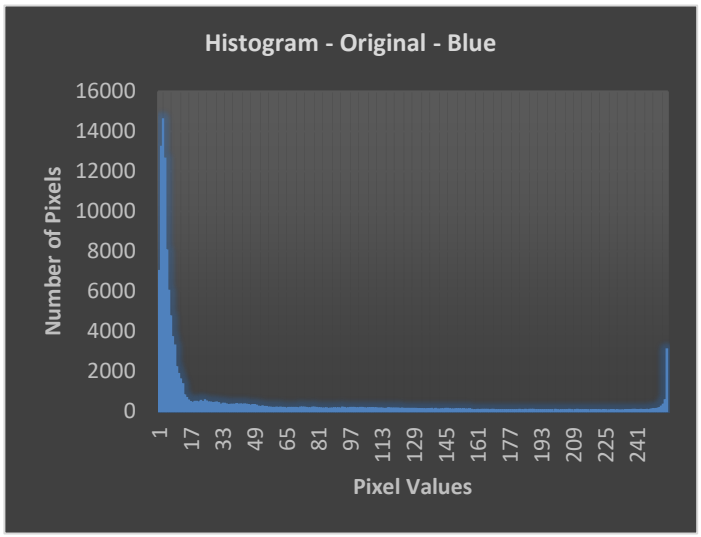
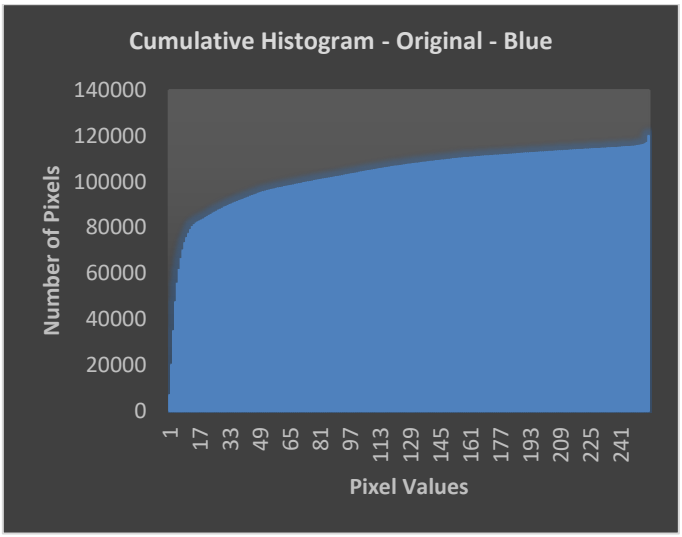
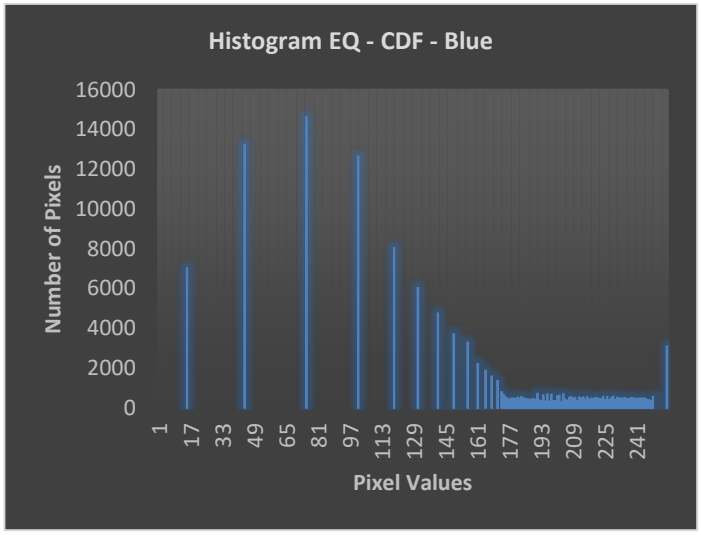
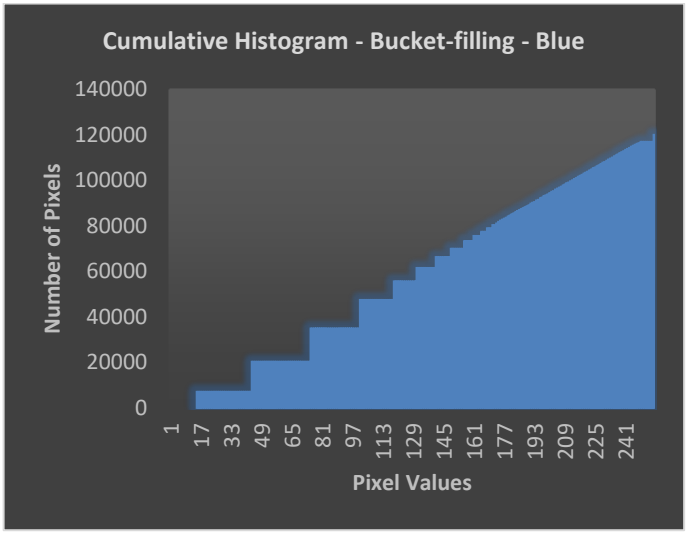


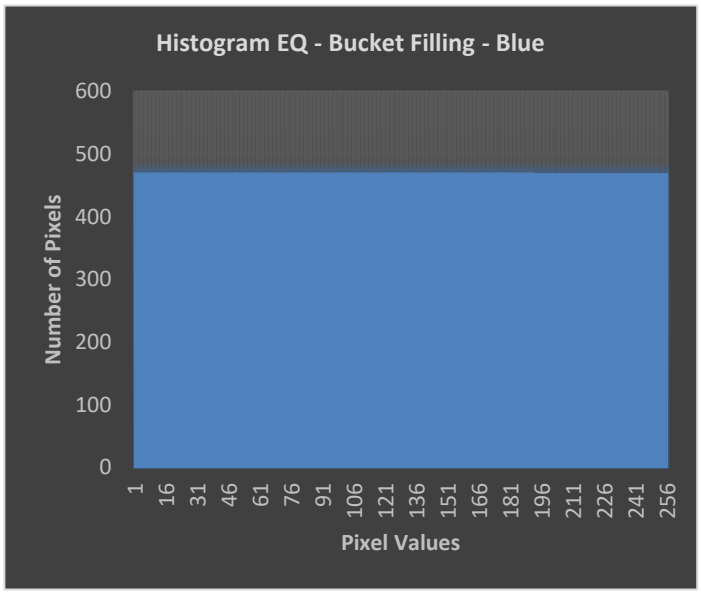
Figure 3: Histogram for GREEN channel of Desk.raw
a) Original b) CDF c) Bucket-filling d) CDF plot for Bucket Filling











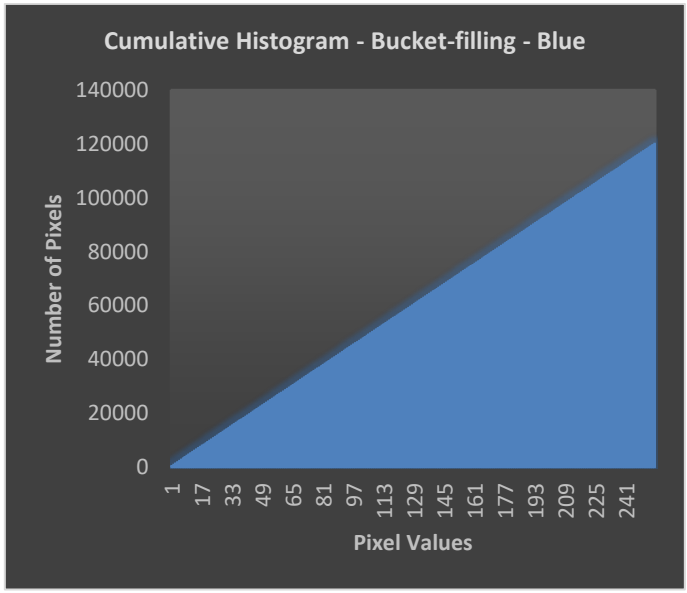


Figure 4: Histogram for BLUE channel of Desk.raw
a) Original b) CDF c) Bucket-filling d) CDF plot for Bucket Filling

Observations

1) **Histogram EQ with CDF:** There is marked improvement in the contrast of the image. The areas that were not visible earlier are visible now. Especially the books and the tiffin-box. In the original image, nearly all the channels have pixel intensities concentrated in the range 0-29. After performing the equalization with CDF, the histogram primarily shifts in the range of 170-250. Cumulative histogram plot is staggering and overall linear.
2) **Histogram EQ with Bucket-filling:** There is improvement in the contrast of the image but some areas of the channels get highly saturated. There is a concentration of blue pixels around lamp. The areas that were not visible earlier are visible now. Especially the books and the tiffin-box. All the intensities have equal pixels 468. Some of the intensities have 469 pixels to match the total number of pixels. Cumulative histogram plot is linear.

Conclusion

Histogram equalization results in an overall improvement in the contrast of the image. This will be particularly helpful in n night-vision applications. Bucket-filling results in a smooth linear cumulative histogram plot.

For improving the results: To avoid over-saturation of some pixels, Adaptive Histogram can be employed. In that, selectively only some regions can be subject to histogram equalization. Interpolation can also be used to distribute the pixels evenly in the image.

Part B: Oil Painting Effect

Abstract and Motivation

Oil-painting effect is well known effect to reduce the color space of an image. Normally a color image has 3 channels and each channel has 8-bits. This leads to a total of 256 x 256 x 256 = 16.78 million colors. Using oil-painting effect, the color palette can be reduced to contain fewer colors.

Figure 5: Original Subject Images for Oil Painting
a) Star_Wars.raw (600 x 338) b) Trojans.raw (1800 x 1200) c) Barn.raw

Approach and Procedures

For achieving Oil Painting effect, there are two steps:

1) **Reduction of Color Space:** The user can enter the number of bits he desires. The original color set will be quantized and will be reduced to the desired space. Example, if the user chooses 6-bits, then the original 240-bit image is quantized to 6-bits. That means, each channel has 6/3 = 2 bits which results in 2^2 = 4 colors. In case of 9-bits, each channel has 9/3 = 3 bits which result in 2^3 = 8 colors. The user must choose a number between 3-24 and the number should be a multiple of 3.
2) **Choosing pixel value using nearest neighbor algorithm**: For deciding what value should a pixel take, an NxN windows is chosen around the pixel. In that window, the most frequently occurring pixel is chosen. This smoothen outs the image avoids mixed noise.

Results and Discussion

64 Colours | **512 Colours**

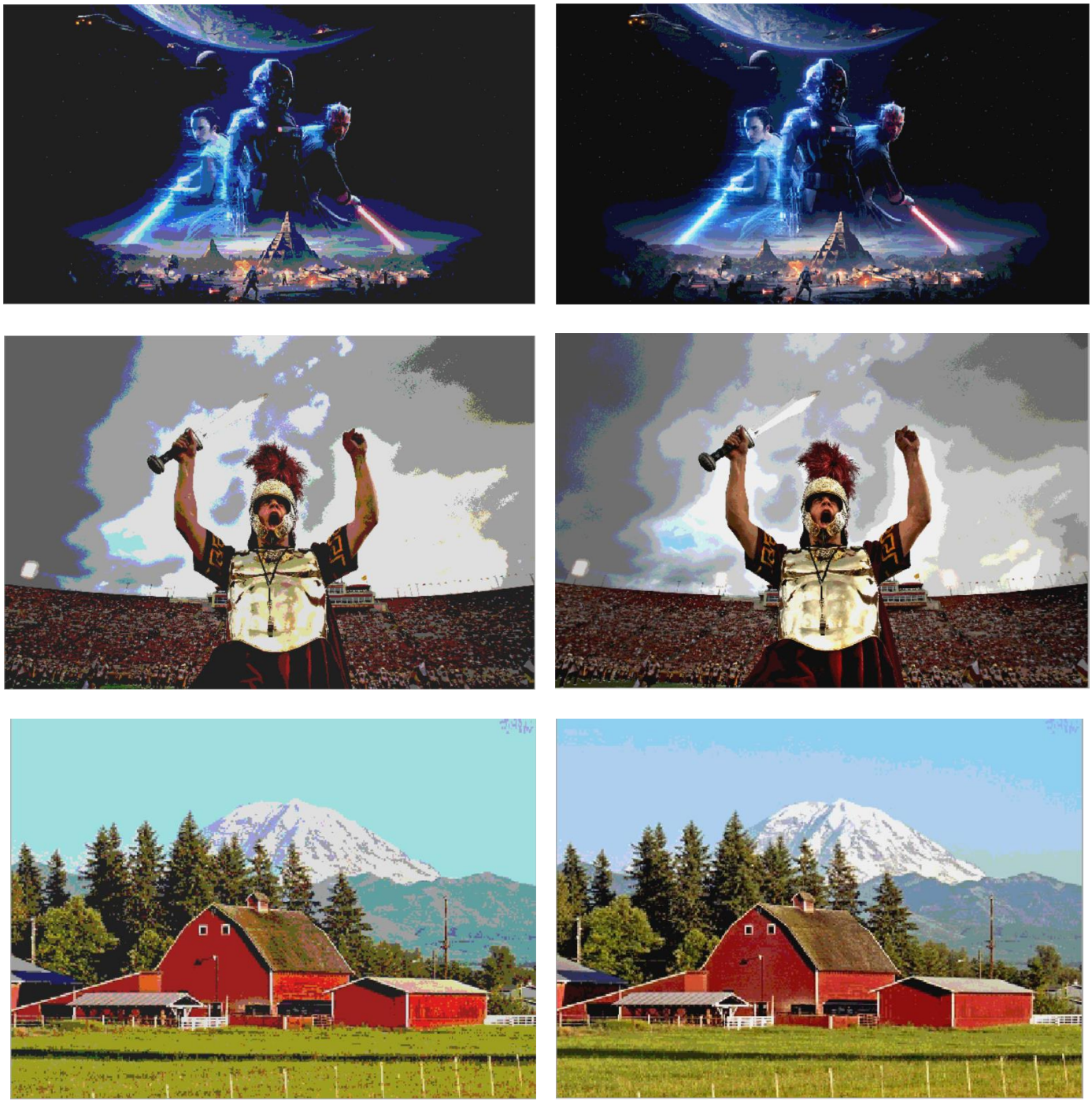


Figure 6: Quantized Images
a) Star_Wars.raw b) Trojans.raw c) Barn.raw

Observations

In the above set of images, it can be observed that images quantized with 512 colors look better and have more color depth. Especially the Trojans and Barn images have marked differences. The sky portion in Barn image has better blue color. The Black background in Star Wars image is deeper in 512 colors than in 64 colors.

64 Colours **512 Colours**

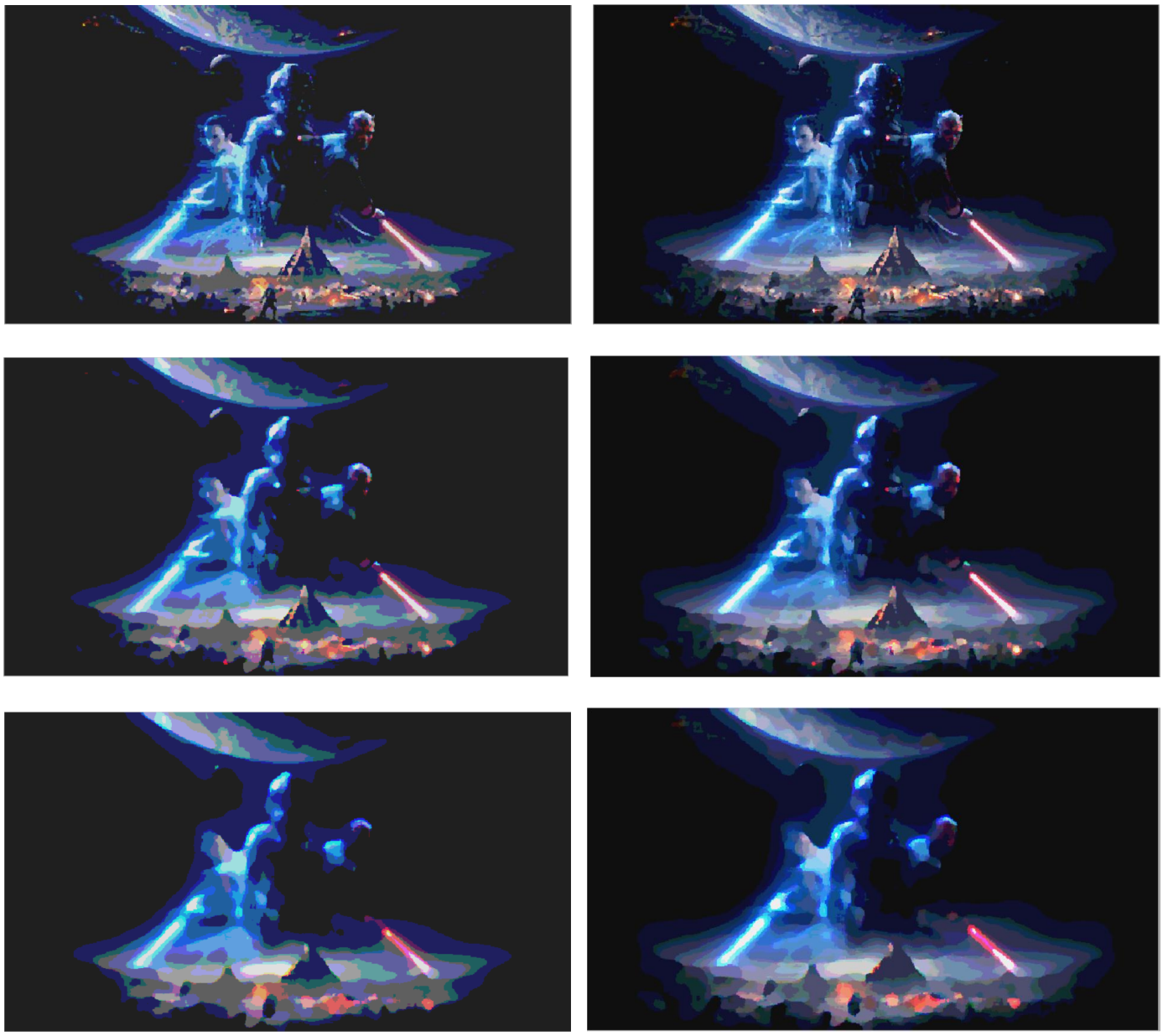


Figure 7: Oil Painting Effect Applied to Star_Wars.raw
a) N = 3 b) N = 7 c) N = 11

Observations

In the above 2 sets of images, Oil-painting effect is more prominent when the number of colors are limited to 64. The black background look much better in images quantized with 512 colors than 64 colors. Also, more is the filter size, better is the smoothening. One exception is last image, 512 colors and N=11, the appearance is quite artistic.

One can notice easily that with increase in the filter size, the “hill” in the image gets more and more blurry. Image with 512 colors has more colors in the sky. Although the difference is very subtle between N=3, 7 and 11, but the sky clearly has more distinguished boundaries with lower N value. The image with N=11 and 64 colors distinctly appears artistic.

64 Colours **512 Colours**

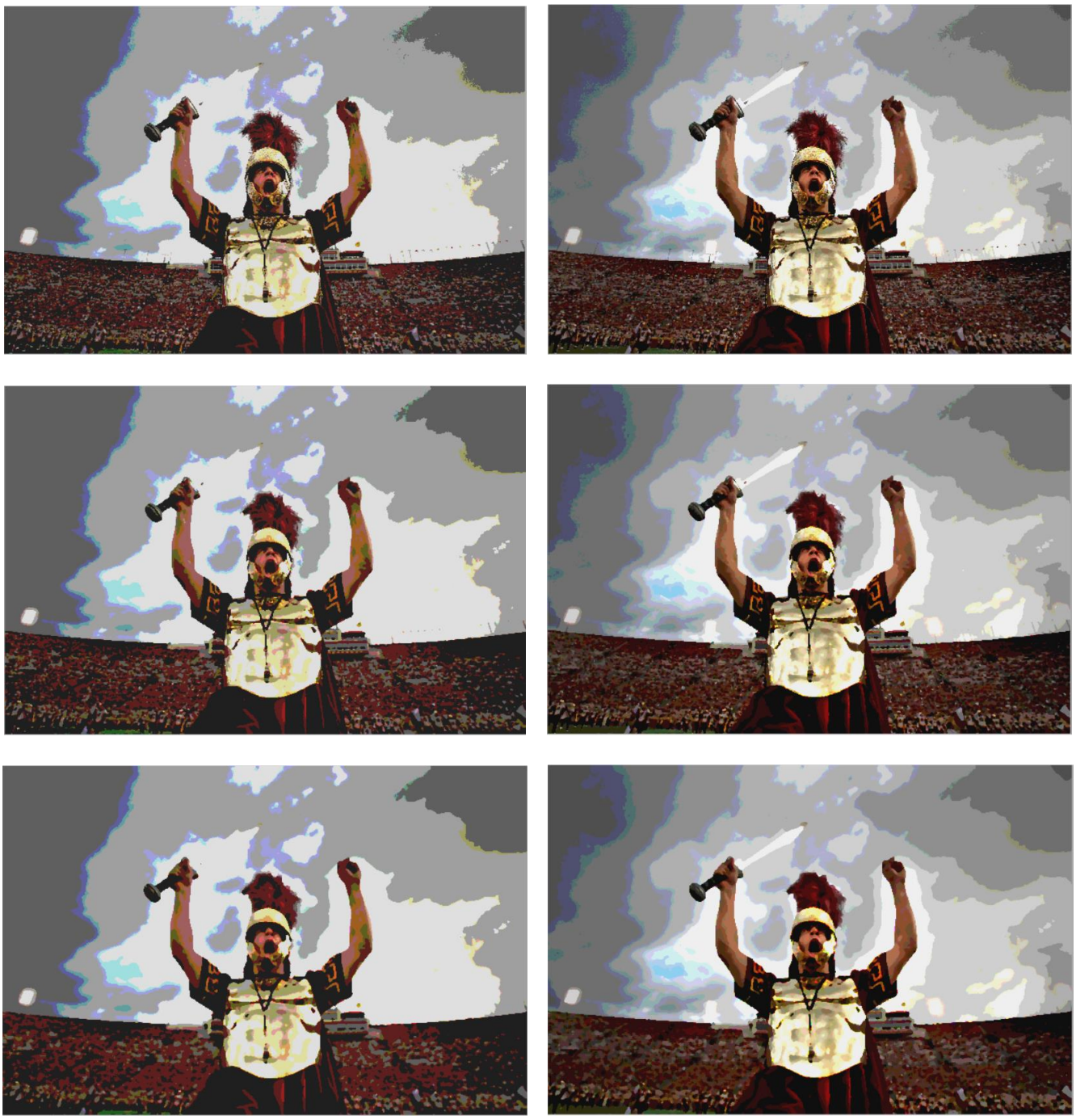


Figure 8: Oil Painting Effect Applied to Trojans.raw
a) N = 3 b) N = 7 c) N = 11

Observations

In the above 2 sets of images, Oil-painting effect is clearly more prominent in 64-color images. In images with 512 colors, there are more color tones available. The effect is most prominent in 64 color-image with N=11. One can notice easily that with increase in the filter size, the crowd gets more and more blurry and the sky gets a two-tone shade in a 64-color image. The image with 64 colors looks overall more smudgy than 512.

Conclusion

Combining the observations of both Star Wars and Trojans image, the Oil-painting effect looks the best when a 6-bit color pallet is selected. The effect gets better with increase in the window size.

Figure 9: Barn.raw a) N=11, number of Colours = 64 b) N=3, number of Colours = 512

Part C: Film Effect

Abstract and Motivation

Film effect is another well-known effect that brings out certain features of the image. It appears as negative of the image with red channel more saturated than usual. The implementation of this effect is quite interested.

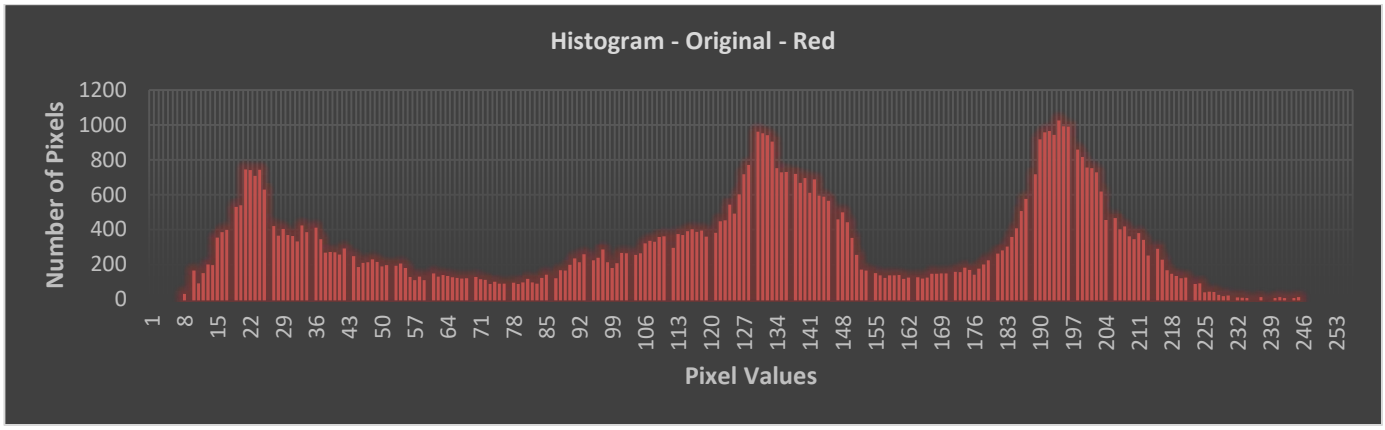
Approach and Procedures

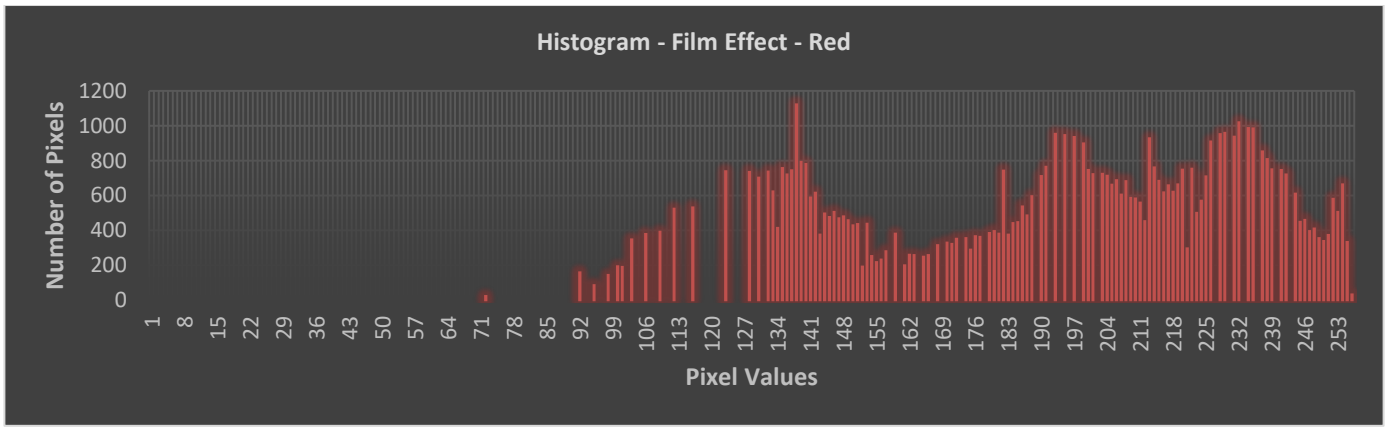
For achieving this effect, the color space of the image must be first changed from RGB to CMY. Next, the image is flipped horizontally. After that, the histogram of each channel must be manipulated to match that of the target image. There 2 methods by which this effect can be achieved:

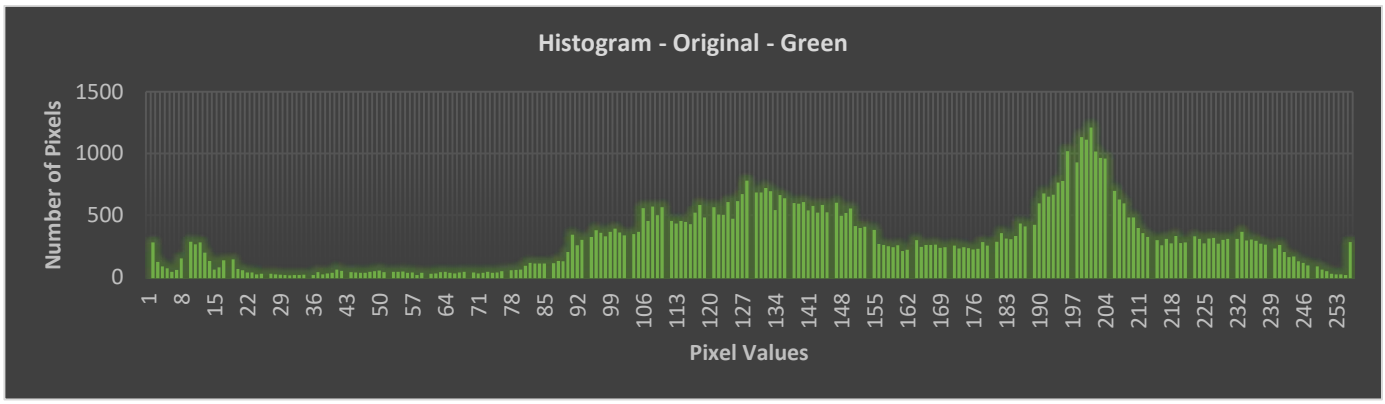
1) **Histogram matching:** The histogram of the original image is matched with a sample image having the similar effect. The Cumulative-probability Distributive Function (CDF) of both original and sample images are calculated and stored. After that, for creating a mapping, the value having the least absolute difference with the original pixels' value in the sample image's CDF is mapped to the original pixel's value.
2) **Creating Transfer-function for Channel-wise histogram manipulation:** The histograms of all channels are analyzed separately for a sample image in an image-editing software like Gimp. The effect is then replicated channel-by-channel.

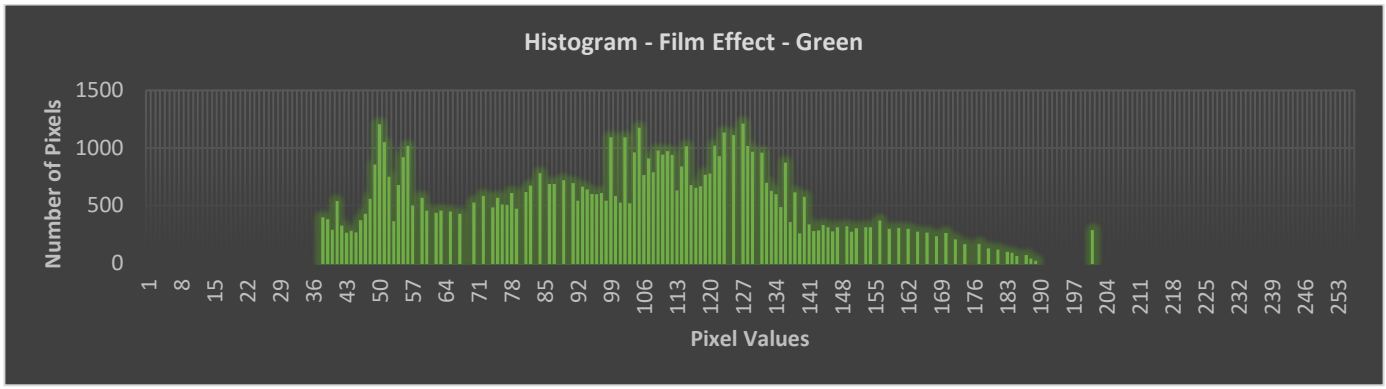
Results and Discussion

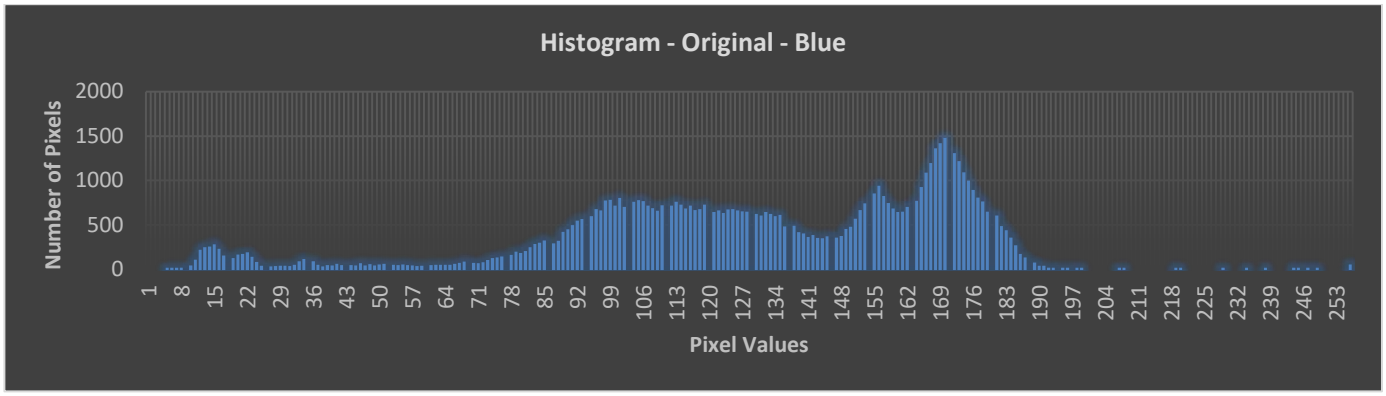
Figure 10: Film Effect applied to Girl.raw


Histogram - Original - Red
Number of Pixels
1200
1000
800
600
400
200
0
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204 211 218 225 232 239 246 253
Pixel Values


Histogram - Film Effect - Red
Number of Pixels
1200
1000
800
600
400
200
0
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204 211 218 225 232 239 246 253
Pixel Values


Histogram - Original - Green
Number of Pixels
1500
1000
500
0
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204 211 218 225 232 239 246 253
Pixel Values


Histogram - Film Effect - Green
Number of Pixels
1500
1000
500
0
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204 211 218 225 232 239 246 253
Pixel Values



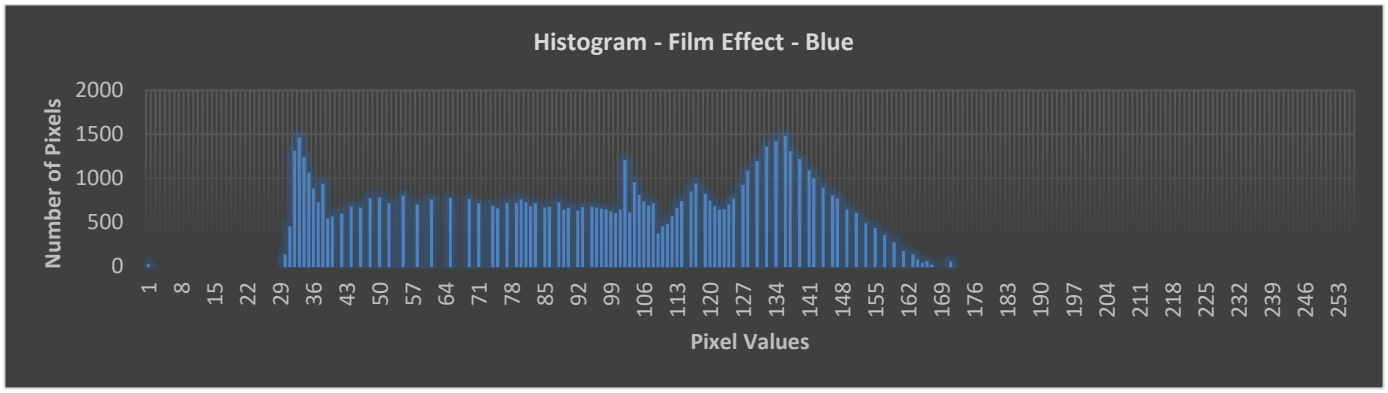


Figure 11: Histogram for R G B channels for Original Image - Girl.raw and Film Effect on Girl.raw

Observations

The histograms of the channels of the result image are inverted in shape. They have also shrunken in terms of total length. The channels have been shrunk as follows:

Red -> 5-250 is now -> 90-255

Green -> 0-255 is now -> 35-190

Blue -> 2-255 is now -> 30-170

Conclusion

It was found that histogram matching performs better than transfer function based histogram manipulation. Not only is the algorithm faster but is also adaptive in nature. The only requirement is that sample image must of the same width and height as the target image. Below is the experiment on Film.raw:

Figure 12: Film effect applied to Original.raw

Problem 3: Noise Removal

Part A: Mix Noise in Color Image

Abstract and Motivation

Noise removal by far is one of the most important part of image processing. The various techniques of noise removal help in enhancing image quality. There are various types of noise that can be present in an image, for example, gaussian noise, salt and pepper noise, mix noise, Poisson noise, speckle noise, etc. Noise can be introduced in an image due to various reasons. Most prominent reason are:

1) Corruption of image during storage or file transmission
2) Noise due to quantization process
3) Application of a noise filter
4) Method noise: Noise due to a certain process applied on an image like scaling, rotation warping, etc.

Sometimes noise is a desirable effect, for example, during dithering process. But a lot of times noise is undesirable and it becomes necessary to remove it or at the least, reduce it to a manageable extent.

There are many process that be used to for noise removal/reduction. Some of the most prominent methods are:

- Low-pass filter: Mean filter, Gaussian filter, Median filter
- Non-Local Means filtering (NLM)
- Principle Component Analysis based filtering (PCA)
- Block Matching 3D filtering (BM3D)
- Cascaded filtering, i.e. using two or more filters in succession

While Low-pass filtering methods are easy to implement when compared to NLM, PCA and BM3D, but there are several disadvantages as well:

- Image tends to get blurry
- Loss of features
- Loss of sharpness
- Overall reduction in contrast and saturation of certain pixel intensities

Approach and Procedures

For removing mix noise in the given subject image **Lena_mixed.raw**, a cascaded low-pass filter was used:

Step 1: Median filter

1. Raw image data was read
2. User enters the desired windows size N, $3 \leq N \leq 11$. In this report, N = 3, 7 and 11 are investigated.
3. Image is masked, i.e. the dimensions were increased, by a factor of (N-1)/2
4. Newly created empty rows, columns and corners are filled with the outermost image edges and corners
5. Convolution: Starting from (0, 0) till (Height-1, Width-1), median value is calculated for each N x N window
6. Each pixel is assigned the median value for that corresponding block

Step 2: Gaussian filter

1. Output from the previous step is used as input
2. An N x N Gaussian kernel is created
3. Image is masked, i.e. the dimensions were increased, by a factor of (N-1)/2
4. Newly created empty rows, columns and corners are filled with the outermost image edges and corners
5. Convolution: Starting from (0, 0) till (Height-1, Width-1), the Gaussian kernel is multiplied element-by-element for each of the corresponding element in the image
6. Each pixel is assigned the value obtained by applying the Gaussian kernel
7.

```
A 9x9 Gaussian kernel with sigma = 1.0:
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
0.000   0.000   0.000   0.001   0.002   0.001   0.000   0.000   0.000
0.000   0.000   0.003   0.013   0.022   0.013   0.003   0.000   0.000
0.000   0.001   0.013   0.059   0.097   0.059   0.013   0.001   0.000
0.000   0.002   0.022   0.097   0.159   0.097   0.022   0.002   0.000
0.000   0.001   0.013   0.059   0.097   0.059   0.013   0.001   0.000
0.000   0.000   0.003   0.013   0.022   0.013   0.003   0.000   0.000
0.000   0.000   0.000   0.001   0.002   0.001   0.000   0.000   0.000
0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
```

Figure 1: A 9 x 9 Gaussian Filter generated for during implementation

Step 3: PSNR Calculation

Since the resulting image after noise removal is going to be different from the input image, we can utilize the Peak Signal to Noise ratio (PSNR) as an indicative measure to judge the quality of the image. For calculating the PSNR, calculation of Mean Square Error (MSE) is required. We know that MSE is direct measure of distortion of an image.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(X'(i) - X(i)\right)^2$$

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

Figure 2: Formula for MSE and PSNR, MAX = 255

Results and Discussion


Figure 3: Input Images a) Lena.raw (Original image) b) Lena_mixed.raw (Noisy image)

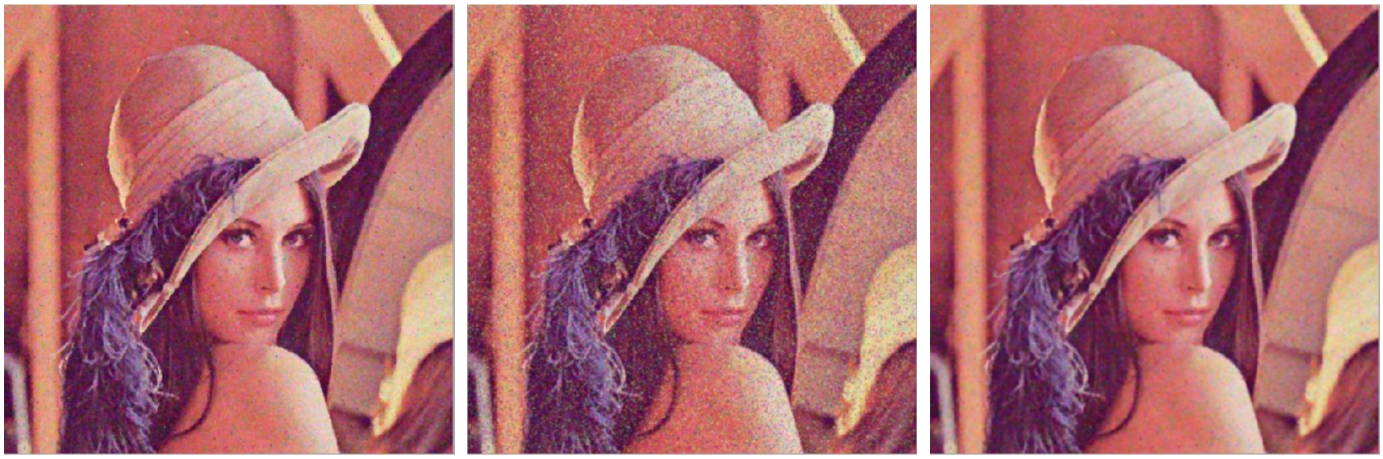


Figure 4: Noise Removal – Filter Size = 3
a) Only Median Filter b) Only Gaussian Filter c) Cascaded Filter (Median and Gaussian)

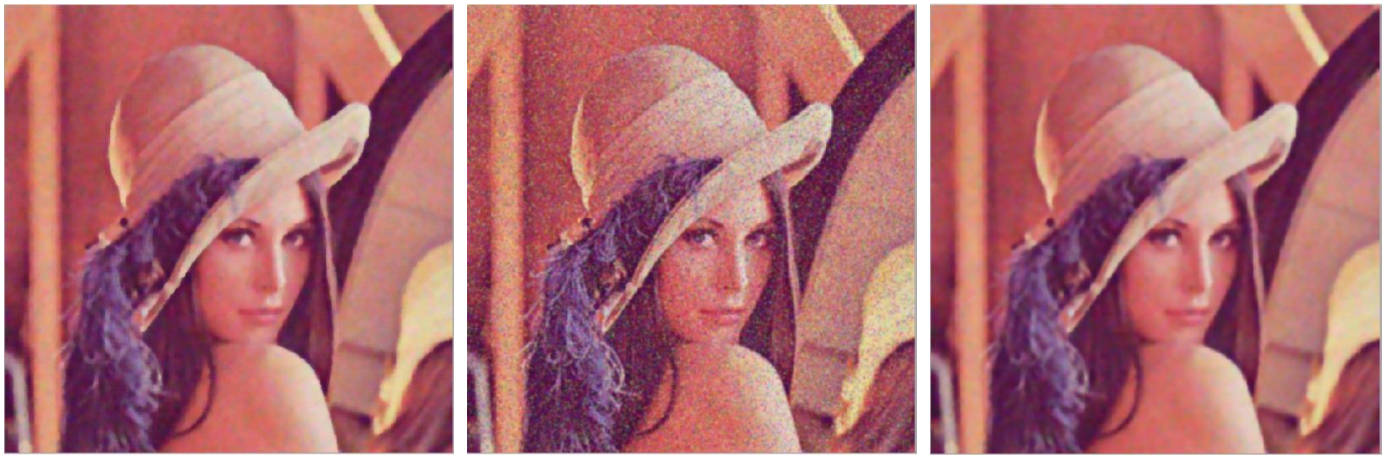


Figure 5: Noise Removal – Filter Size = 7
a) Only Median Filter b) Only Gaussian Filter c) Cascaded Filter (Median and Gaussian)

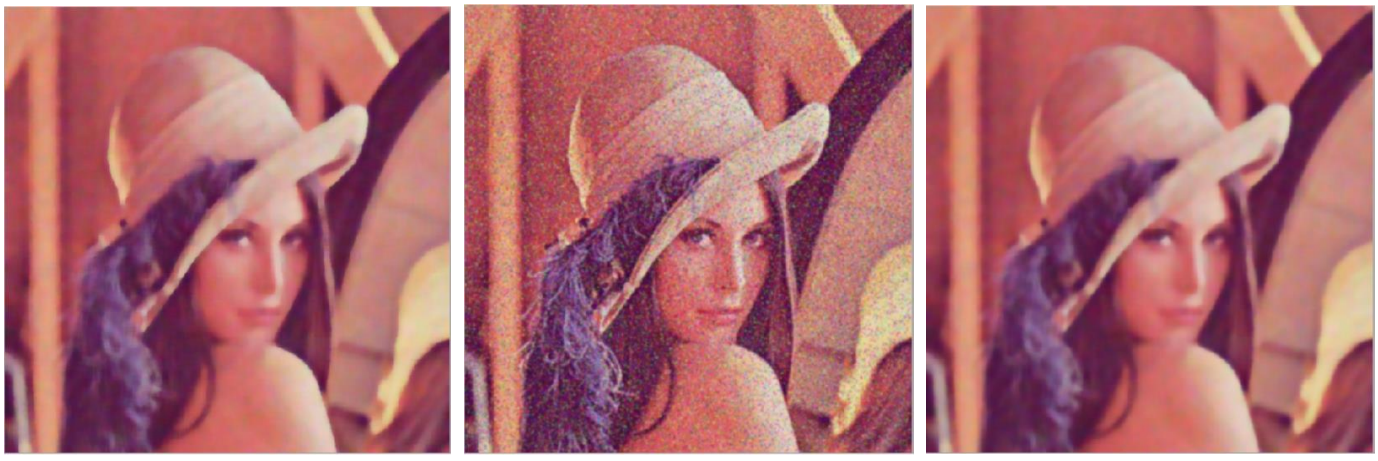


Figure 6: Noise Removal – Filter Size = 11
a) Only Median Filter b) Only Gaussian Filter c) Cascaded Filter (Median and Gaussian)

MSE/PSNR (dB) Data	Block Size (N)	Red		Green		Blue		Overall	
		MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
Noisy Image	-	1272.0	17.09	1440.0	16.55	1293.0	17.01	1335.0	16.88
Only Median	3	262.5	**22.78**	357.4	**22.60**	276.2	**23.72**	221.5	**24.68**
Only Gaussian	3	342.3	**23.94**	140.0	**26.67**	262.0	**23.95**	325.3	**23.01**
Cascaded	3	209.5	**24.92**	116.1	**27.48**	187.0	**25.41**	170.9	**25.80**
Only Median	7	178.1	**25.62**	150.9	**26.34**	160.5	**26.08**	163.2	**26.00**
Only Gaussian	7	292.0	**23.48**	295.9	**23.42**	221.4	**24.68**	269.8	**23.82**
Cascaded	7	185.1	**25.46**	159.7	**26.10**	152.9	**26.29**	165.9	**25.93**
Only Median	11	208.3	**24.94**	210.0	**24.91**	174.0	**25.73**	197.4	**25.18**
Only Gaussian	11	291.9	**23.48**	295.8	**23.42**	221.3	**24.68**	269.7	**23.82**
Cascaded	11	220.6	**24.70**	219.1	**24.72**	173.6	**25.74**	204.4	**25.03**

Table 1: Mean Square Error (MSE) and PSNR Data

Observations

1) Objectively, best results are obtained with N=7, PSNR = 25.93 overall. But subjectively, N=3 looks very good. The image produced using cascaded filter with N=7 looks blurrier than N=3. It can be noticed that N=3 has best PSNR with Green channel. Since humans perceive more than any other color, it can be reason that N=3 provides best result.
2) PSNR increases with increase N but starts reducing after N=7. In most of the cases PSNR reduces considerably for N=11. Example, the PSNR for green channel using cascaded filter is 26.10 dB for N=7, but it only 24.72 dB for N=11. Overall PSNR has also reduced from 25.93 dB to 25.03. Subjectively, with increase in N, the image loses certain feature and details, which makes it look smudged.
3) Subjectively, images with a little noise is visually better than image with no noise and blurriness.
4) Using only Median filter for N=7 and N=3 produces better results than using a cascaded filter.

Conclusion

Low-pass filtering methods remove noise but at the cost of inducing blurriness in the image and cause loss of key features and details. Hence for sophisticated applications, such methods are not preferable. Also, using a Gaussian filter alone cause much more blurriness than using Median filter alone. It is advisable to use just the Median filter alone instead of a cascaded filter for larger N. For the current image, Median filter with N=7 and Cascaded filter with N=3 produced the best result, subjectively. Filter with N=11 performs worst in terms of subjective quality.

Answers to Questions:

1) Both Red and Green have impulse noise. Green has White Gaussian noise as well as impulse noise, while Blue has very prominent Gaussian Noise. Histogram analysis reveals that.
2) We should perform filtering separately for each channel but the algorithm will have diminishing return. Both impulse and Speckle noise can be best reduced with Median. Mean filtering can be used Speckle noise is Median is not available. Gaussian noise is best removed by using a Gaussian filter.
3) Median and Gaussian filters have been used to remove the noise in this report.
4) Gauss should after Median in implementing a Cascade filter otherwise outliers will not be removed and Gaussian filter will spread the effect of the outliers on the image.
5) Yes, window size affects result quality. It was observed that a larger window size removes more noise but at the cost of inducing blurriness. Specially, a Gaussian filter with large window size is not recommended.

Part B: Principle Component Analysis (PCA)

Abstract and Motivation

PCA essentially works on Parseval's theorem or Energy-Compaction theorem in general. This simply means that, all the data can be considered as nothing but energy. Now, in a stream of data, clusters or vectors of components can be formed. For example, in case of sound signals, the various frequencies present in the signal become the components. Number of components determine the dimensionality of the signal. Such that:

Energy of Signal = Energy of frequency 1 + Energy of frequency 2 + …. + Energy of frequency 'n'

In case of a noisy signal, some of the energy of the signal is shared by the noise as well, such that:

Energy of Signal = Energy of Actual Signal + Energy of Noise

Since noise can also be broken down into components we can say that:

Energy of Signal = [Energy of frequency 1 + Energy of frequency 2 + …. + Energy of frequency 'n'] +
[Energy of Noisy frequency 1 + Energy of Noisy frequency 2 + …. + Energy of Noisy frequency 'n']

In real life cases, we generally don't know that which frequency is exactly contributing to the noise. In general, the practical signals have a lot of components, but much of their energy compacted into a very few components:

Energy of Signal = Energy of MAJOR frequency 1 + Energy of MAJOR frequency 2 + …. + Energy of MAJOR 'm'
+ Energy of MINOR frequency (n-2) + Energy of MINOR frequency (n-1) + …. + Energy of MINOR frequency (n)

OR

Energy of Signal = Energy of MOST PROMINENT frequencies + ENERGY of LEAST PROMINENT frequencies

Now, PCA helps in getting rid of the LEAST prominent frequencies, thus reducing dimensionality of the signal. Since most of the energy is concentrated only in the very first few components, the signal can be reconstructed easily.

PCA for Noise Reduction in Images – Global PCA approach

Just like sound, images have components too. The smallest component of an image is pixel. But dealing in pixel alone is very time consuming. That's why, a cluster of pixels or PATCH is used to manipulate the image. Such a patch can be an N X N sized block.

So, for analyzing an image, several such patches are created and they are analyzed for similarity with other. This helps in determining similar patches which in turn helps in identifying patterns in the image. Identifying patterns in an image is good since it may happen that a patch or a pattern is falsely identified as noise.

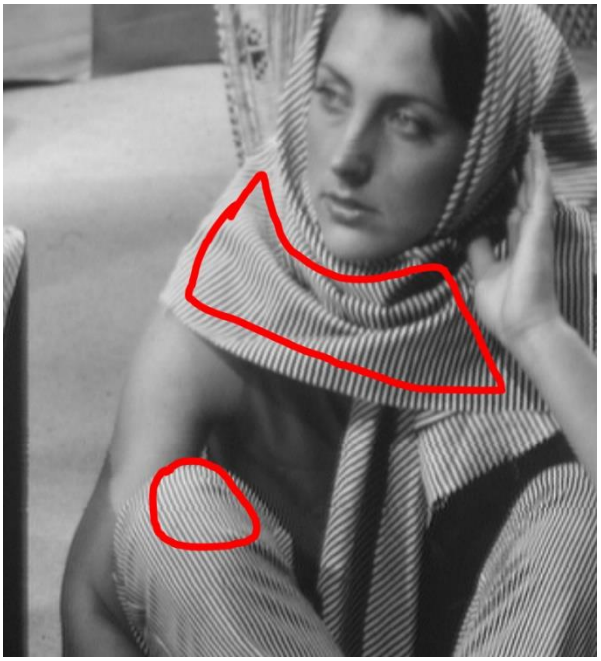


Figure 7: Barbara.png. Crisscross pattern which is falsely recognized as noise by many algorithms

For example, in the accompanying image, Barbara.png, the crisscross pattern on the scarf and the pants is falsely recognized as noise by many algorithms. But PCA identifies this as a separate component of the image which helps in preserving that feature of the image. This is possible because an orthonormal basis of components is created which helps is segregation of components and hence, the segregation of energy contained in them becomes easy.

Patch-based local PCA (PLPCA)

In the patch-based local PCA (PLPCA) approach, patches are collected locally to overcome the limitations of the global PGPCA approach. The local collection of patches means that the patches are collected within a small region of interest in the noisy image. A fixed search window N×N is applied to the whole image. Since the patches are overlapped in PLPCA, there will be multiple estimates for a single pixel. Averaging is used to compute a single pixel's value.

The advantage of this approach is that the orthonormal basis is adapted only to the sub-image and not to the whole image. However, this approach has two limitations:

1) Overfitting
2) Time-consuming.

The overfitting problem is due to the limited number of patches on which to compute the PCA. PLPCA is extremely time-consuming because the PCA needs to be computed repeatedly for each pixel. But the run-length is said to be lesser than NLM while more than BM3D. On the other hand, the result quality is better than NLM and almost comparable to BM3D, this is by far the best advantage of PCA based approach.

Approach and Procedures

For experimentation purposes, the MATLAB code written by C.-A. Deledalle, J. Salmon, A. S. Dalalyan was used.

1) **Threshold (threshold)**: This parameter decides the number of components to be considered. Since this quantity has direct impact on algorithm run time, it is adjusted first.
2) **Patch Overlap (delta)**: This parameter determines the stride at which patches are created and used for PCA calculation. The patches created are used to calculate an average value which is assigned to the pixel later.
3) **Patch-Size (WP):** A larger patch size essentially leads to more number of components being taken into consideration for PCA calculation. If the number of components to be selected is kept fixed, then there is a significant reduction is dimensionality. But if the number of components if kept as percentage of the patch size, then no significant effect is observed.

Results and Discussion

1) **Experiment with different levels of Threshold** (Patch size = 7 and delta = 11) **[Best threshold value = 2.75]**

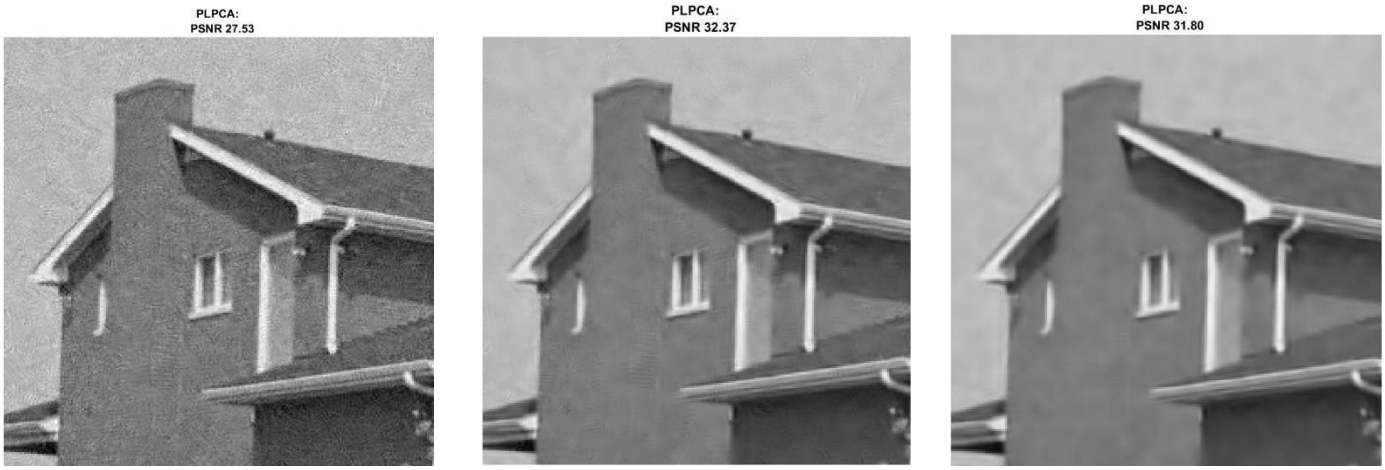


Figure 8: Experiment with threshold value 3, 5 and 7

2) **Experiment with different delta values** (Threshold = 2.75, Patch size = 7) **[Best value of delta = 1]**

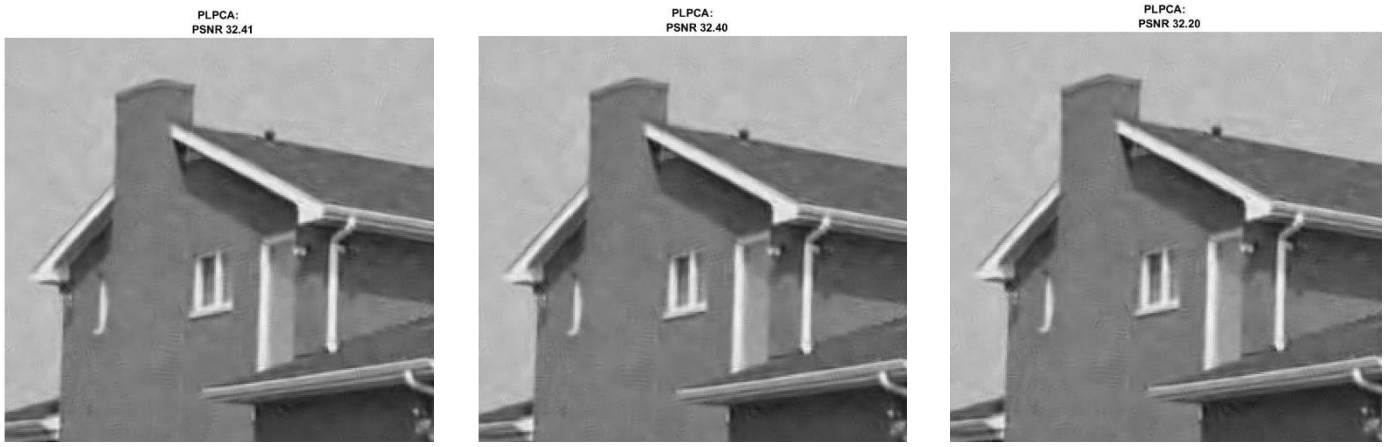


Figure 9: Experiment with delta values 1, 7 and 21

3) **Experiment with different Patch sizes** (Threshold = 2.75, Delta = 7) **[Best Patch size = 9]**

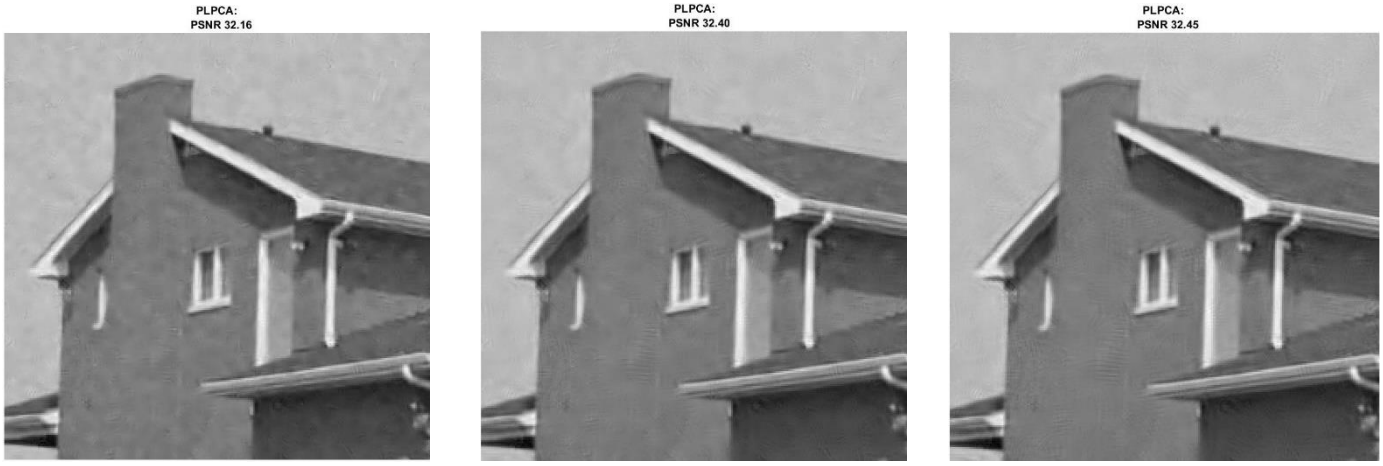


Figure 10: Experiments with Patch-size 5, 7 and 9

Default values of Parameters:	**Best values of Parameters found after experiment:**
Threshold=2.75, Delta=7 and Patch =11, PSNR=32.37 dB	Threshold=2.75, Delta=7 and Patch=9, PSNR=32.45 dB

Observations - Brick pattern on wall and roof is gone in most cases, except for patch-size 9, threshold 5 and delta as 7. This happens because PCA identifies pattern as noise rather than a detail.

Threshold is the number of components chosen for PCA. It is the most the important parameters that affects image quality. It can be observed that a lower threshold value results in strange wiggly artefacts in the image. A higher value blurs the image.

Delta hardly affects the image quality but it does affect the run-time of the algorithm. The best value was found to be 1 but it was very time consuming to proceed with that. Since there was not much difference between the PSNR of image with delta 1 (PSNR = 32.41 dB) and delta 7 (32.40 dB), it was decided to proceed with delta as 7.

Patch-size affects the image quality is a subtle sense. A low value (5) resulted in a PSNR of 32.16 dB while a very higher value (9) resulted PSNR of 32.45 db. Values, especially those 11 and above reduced the PSNR.

Conclusion

PLPCA is an effective denoising algorithm with average running time of 2-3 seconds for 256x256 images. It is affected by choice of parameters and there is no trend for making the choice.

Part C: Block-Match 3D (BM3D)

Abstract and Motivation

Like PLPCA, BM3D also works on Parseval's theorem or Energy-Compaction theorem, essentially. But as a difference, instead of searching for similar sub-patches in a larger patch, BM3D essentially builds a 3D stack of similar 2D blocks. Now, each of the 2D sub-images in the 3D stack are processed for feature reduction. The step of feature reduction is the main thing that helps in noise reduction. After that, an inverse 3D transform is used to get back a 2D patch. It is completely plausible that patches can overlap, which means a single pixel's value may be derived by using values of that pixel from several newly generated 2D patches. Once such aggregation technique is averaging.

This whole process is done 2 times. But in both steps, the method by which similarity of patches is detected is different. In step 1, either of soft or hard thresholding can be used. In step 2, Weiner thresholding is used to identify similarity. Another key difference is both the steps is that, in step 1, the patches are formed from the input image. But in step 2, patches are formed using the output image of step 1. This greatly reduces noise.

BM3D heavily relies on sparse matrix processing and singular value decomposition methods for dimensionality reduction. Algorithms like KLT, PCA or DCT can be used to do that.

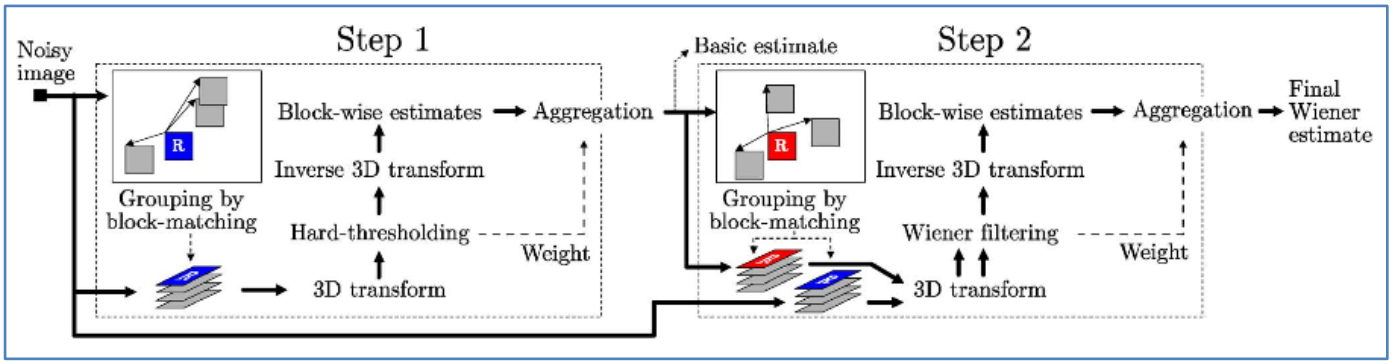


Figure 11: Flow-chart showing the realization of BM3D algorithm

Answers to questions:

2 - The intuition behind BM3D is the same as other non-local algorithms: an image is made by the juxtaposition of many, self-repetitive small image elements called patches. Thus, when looking at small enough pieces, it is possible to find enough similar patches that can be combined and denoised together. Block-matching is a very powerful concept used in video compression. This also introduces homogeneity in the image.

3 – BM3D utilized spatial reduction techniques to produce sparse matrices. This can be done by using any of PCA or DCT. If PCA is used, then BM3D becomes a spatial domain filter. In case of DCT, it will become a frequency based filter. By this logic, BM3D ca ne classified as both spatial and frequency domain filtering method.

4 – BM3D utilized spatial reduction techniques to produce sparse matrices. This can be done by using any of PCA or DCT. If PCA is used, then BM3D becomes a spatial domain filter. In case of DCT, it will become a frequency based filter. By this logic, BM3D can be classified as both spatial and frequency domain filtering method.

Approach and Procedures

For experimentation purposes, the MATLAB code written by Maggioni et al was used.
Following are the default parameters:

```
%%%% Hard-thresholding (HT) parameters:
N1                      = 8; %% N1 x N1 is the block size used for the hard-thresholding (HT) filtering
Nstep                   = 3; %% sliding step to process every next reference block
N2                      = 16; %% maximum number of similar blocks (maximum size of the 3rd dimension of a 3D array)
Ns                      = 39; %% length of the side of the search neighborhood for full-search block-matching (BM), must be odd
tau_match               = 3000; %% threshold for the block-distance (d-distance)
lambda_thr2D            = 0;   %% threshold parameter for the coarse initial denoising used in the d-distance measure
lambda_thr3D            = 2.7; %% threshold parameter for the hard-thresholding in 3D transform domain
beta                    = 2.0; %% parameter of the 2D Kaiser window used in the reconstruction

%%%% Wiener filtering parameters:
N1_wiener               = 8;
Nstep_wiener            = 3;
N2_wiener               = 32;
Ns_wiener               = 39;
tau_match_wiener        = 400;
beta_wiener             = 2.0;
```

Parameters in the code: **Nstep**, **Ns**, **tau_match** were identified as the key factors affecting PSNR of the output image. They correspond to parameters named **delta** (stride), **patch** and **threshold** in the code given for PCA. The following steps were taken:

1) Adjust the threshold keeping delta and patch size constant
2) Adjust patch-size keeping
3) Adjust delta keeping threshold and patch-size content

Results and Discussion

1) **Experiment with Threshold** (Patch = 39 and delta = 16) **[Best threshold: Hard = 3000 and Weiner = 100]**

Figure 12: Experiment with threshold value (HT, WT) = (3000, 400), (1000, 400) and (3000, 100)

2) **Experiment with delta** (Threshold: Hard = 3000, Weiner = 100; Patch size = 39) **[Best value of delta = (1, 3)]**

Figure 13: Experiment with delta values (HT, WT) = (1, 3), (3, 1) and (3, 3)

*HT = Hard Threshold, WT = Weiner Threshold

3) **Experiment with Patch sizes** (Threshold: Hard = 3000, Weiner = 100; Delta = (1, 3)) **[Best Patch size = 60]**

Figure 14: Experiments with Patch-size 10, 39 and 60

Following are the best parameters that were determined after experimentation:

```
N1                  = 8;
Nstep               = 3;
N2                  = 16;
Ns                  = 60;
tau_match           = 3000;
lambda_thr2D        = 0;
lambda_thr3D        = 2.7;
beta                = 2.0;

%%%% Wiener filtering parameters:
N1_wiener           = 8;
Nstep_wiener        = 1;
N2_wiener           = 32;
Ns_wiener           = 60;
tau_match_wiener    = 400;
beta_wiener         = 2.0;
```

Best PSNR value achieved is 32.95 dB. Compared to PLPCA, which has a PSNR of 32.45 dB, BM3D performed better.

Observations - Brick pattern on wall and roof is gone in most cases.

Threshold is the number of components chosen for dimensionality reduction. It is the most the important parameters that affects image quality. It can be observed that a lower threshold value results in vertical artefacts. A higher value blurs the image.

Delta hardly affects the image quality but it does affect the run-time of the algorithm. The best value was found to be (3, 1) but it was very time consuming to proceed with that. This results in 0.1 dB increase as compared to default parameters of (HT, WT) = (3, 3).

Patch-size affects the image quality is a subtle sense. A low value (10) resulted in a PSNR of 32.21 dB while a very higher value (60) resulted PSNR of 32.95 db. But higher value takes more running time as well.

Conclusion

BM3D is an effective denoising algorithm with average running time of 5-6 seconds for 256x256 images. It is affected by choice of parameters and there is no trend for making the choice. It performs slightly better than PLPCA in terms of objective quality (PSNR). BM3D algorithm introduces hallucinations in the image, i.e. sometime the algorithm introduces features in the result image that are not there. For example, in the sample Barbara image a crisscross pattern on the pant in introduced which was not there in the image. In the original image, only vertical lines are present. The sum of all the patches for a certain location is 1, we average he pixel found in those different patches. After a stack is created, we weigh the contribution of a group according the noise in that group. Noise groups are weighed lower.

In signal processing, the Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process. So, using this in place of Hard or Soft thresholding provides a more sophisticated and better match for similarity of patches.

Citations

[1]: http://fourier.eng.hmc.edu/e161/lectures/contrast_transform/node3.html

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

[3]: BM3D Understanding: https://www.youtube.com/watch?v=BlDl6M0go-c

[4]: http://imagine.enpc.fr/publications/papers/BMVC11.pdf

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

[6] http://www4.comp.polyu.edu.hk/~cslzhang/paper/PR_10_x_3.pdf

[7] https://www.math.u-bordeaux.fr/~cdeledal/files/slidesBMVC2011_PatchPCA.pdf

[8] https://jivp-eurasipjournals.springeropen.com/articles/10.1186/s13640-017-0203-4

[9] http://josephsalmon.eu/code/index_codes.php?page=GPPCA

[10] https://www.sciencedirect.com/science/article/pii/S0031320309003677

[11] https://gist.github.com/microo8/4065693

[12] http://www4.comp.polyu.edu.hk/~cslzhang/LPG-PCA-denoising.htm

[13] http://lib.stat.cmu.edu/multi/pca.c

[14] http://www4.comp.polyu.edu.hk/~cslzhang/paper/PR_10_x_3.pdf