EE569 Digital Image Processing

Spring 2018

Assignment 1

Name: Thiyagarajan Ramanathan

USC ID: 4973341255

Issue Date: 01/08/2017 Due Date: 02/04/2018

Problem 1: Basic Image Manipulation (30%)

In this problem, you will conduct a series of simple manipulations on grayscale and color images to get familiar with image data access, processing and output.

(a) Color Space Transformation (Basic: 18%)

Different color spaces are adopted for different applications. In this problem, you will examine a couple of color spaces that are employed in image processing.

1. **Color-to-Grayscale Conversion**
   Each color pixel of an image is described by a triple (R, G, B) for red, green, and blue color intensities. There are many ways to convert a color image to its grayscale image. The **lightness** method averages the most prominent and least prominent colors: (max(R, G, B) + min(R, G, B)) / 2. The **average** method simply averages the values: (R + G + B) / 3. The **luminosity** method is a more sophisticated version of the average method. It also averages the values, but it forms a weighted average to account for human perception. The formula for luminosity is 0.21 R + 0.72 G + 0.07 B. Convert the color *Tiffany* image in Figure 1 to its gray scale images using the above three methods. Show the three gray scale images and discuss which method works best overall.

Figure 1: Color Tiffany image

2. **CMY(K) Color Space** - The Cyan-Magenta-Yellow-(Black) (CMY(K)) color space is frequently used in image printing. It is defined by:

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

What is the CMY representation for test images *Bear* and *Dance* in Figure 2? For each input color image, produce 3 output grayscale images, corresponding to the cyan, magenta, yellow three channels and show these three grayscale images in your report.

What is the CMY representation for test images *Bear* and *Dance* in Figure 2? For each input color image, produce 3 output grayscale images, corresponding to the cyan, magenta, yellow three channels and show these three grayscale images in your report.

Figure 2: Bear and Dance images

(b) Image Resizing via Bilinear Interpolation (Advanced: 12%)
Use the bilinear interpolation to re-size the input color image *airplane* of size 512x512 in Figure 3 to an output image of size 650x650. An image re-sizing example for the *House* image is given in Figure 4.

Figure 3: Airplane

(a) house image of size 512x512

(b) resized house image of size 650x650

Figure 4: Original and resized House images.

(1.a.1) **Color Space Transformation**

Abstract and Motivation:

The first part of the problem requires us to convert a color image to a gray scale image. Any color image has 3 components namely Red, Green and Blue. The number of colors per channel can be any value such as 2-bit (4 colors per channel), 4-bit (16 colors per channel) and so on. Generally, most of the color images have 8-bits (256 colors) per channel. This means that there are 24-bits per pixel. When compared to this, a gray-scale image only has 8 bits per pixel, which makes it very easy to process by decreasing the complexity of the model built for processing that particular image. The main aim of converting a color image to a gray scale image is that, it is easier to work with gray scale images when compared to the color images.

Approach and procedures:

There are three main methods for converting a color image to gray scale image. They are lightness method, luminosity method and averaging method. Although all the three methods give us a gray scale image, different methods are preferred depending on the application. All the three methods are explained below.

i) Lightness method:

This method allocates the mean of the most prominent and the least prominent component of each pixel.

$$\text{New grayscale color value}=0.5 * (\max(R,G,B) + \min(R,G,B))$$

ii) Average method:

This method averages all the three components R, G, B of each pixel and allocates it to the new pixel.

$$\text{New grayscale color value}=(R+G+B)/3$$

iii) Luminosity method:

This method uses the formula 0.21R + 0.72G + 0.07B to find out the gray scale color value of the new pixel. This method gives more weightage to the green color, to match the human brightness perception.

Experimental Results:

Lightness method

Fig 1

Average Method

Fig 2

Luminosity Method

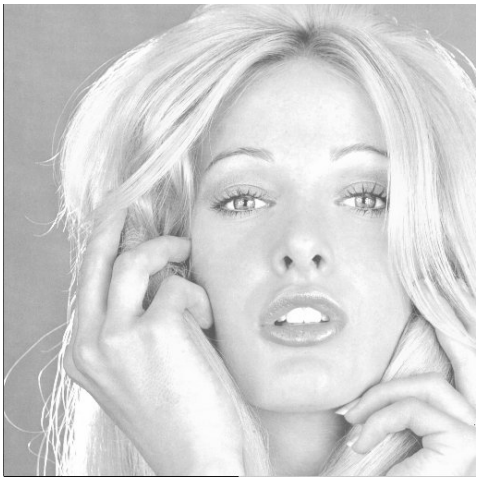


Fig 3

Discussion:

Comparing the three images, we can easily visualize that the image obtained from the lightness method (fig 1) is very dull when compared to the images obtained from the other methods. If we consider a pixel which has a higher value, i.e., towards the brighter side, we take the mean of that value and the minimum value of the three components. Hence the brightness is significantly reduced.

Though the brightness variation in the image obtained by the averaging method is slightly better than the lightness method, this method works well only if the color values of each channel are all close to each other. If one channel value is very high and the other is very low it significantly affects the image brightness quality.

The image obtained by the luminosity method seems the best because of the fact that it gives more weightage to the green color. Since the human eye is most receptive to the green color, even slight variations in this color is easily perceived. Hence the brightness variations in this method is closer to the one in the original image.

From the images given, we can observe that the image obtained from the luminosity method is the best as the brightness factor is more similar to the original image but in gray scale. This is mainly due to the reason that this method gives more weightage to the green color. Hence this is the commonly used method of color to gray scale conversion and it is used in most of the image processing software.

(1.a.2) **Conversion from RGB to CMY**

Abstract and Motivation:

The second part of the problem requires us to convert the image from RGB to CMY. Since the colors Cyan, Magenta and Yellow are complement to the colors Red, Blue and Green respectively. When light falls on the former colors, the latter colors are visualized by the eye. One of the reasons why image printers use the CMY colors instead of the RGB colors is that the color Yellow is not that easy to obtain with the RGB colors. Another reason is that if the colors Red, Green and Blue are printed, the colors visualized are Cyan, Magenta and Yellow. Instead, if the images are printed in Cyan, Magenta and Yellow the colors visualized will be Red, Green and Blue which are the primary colors.

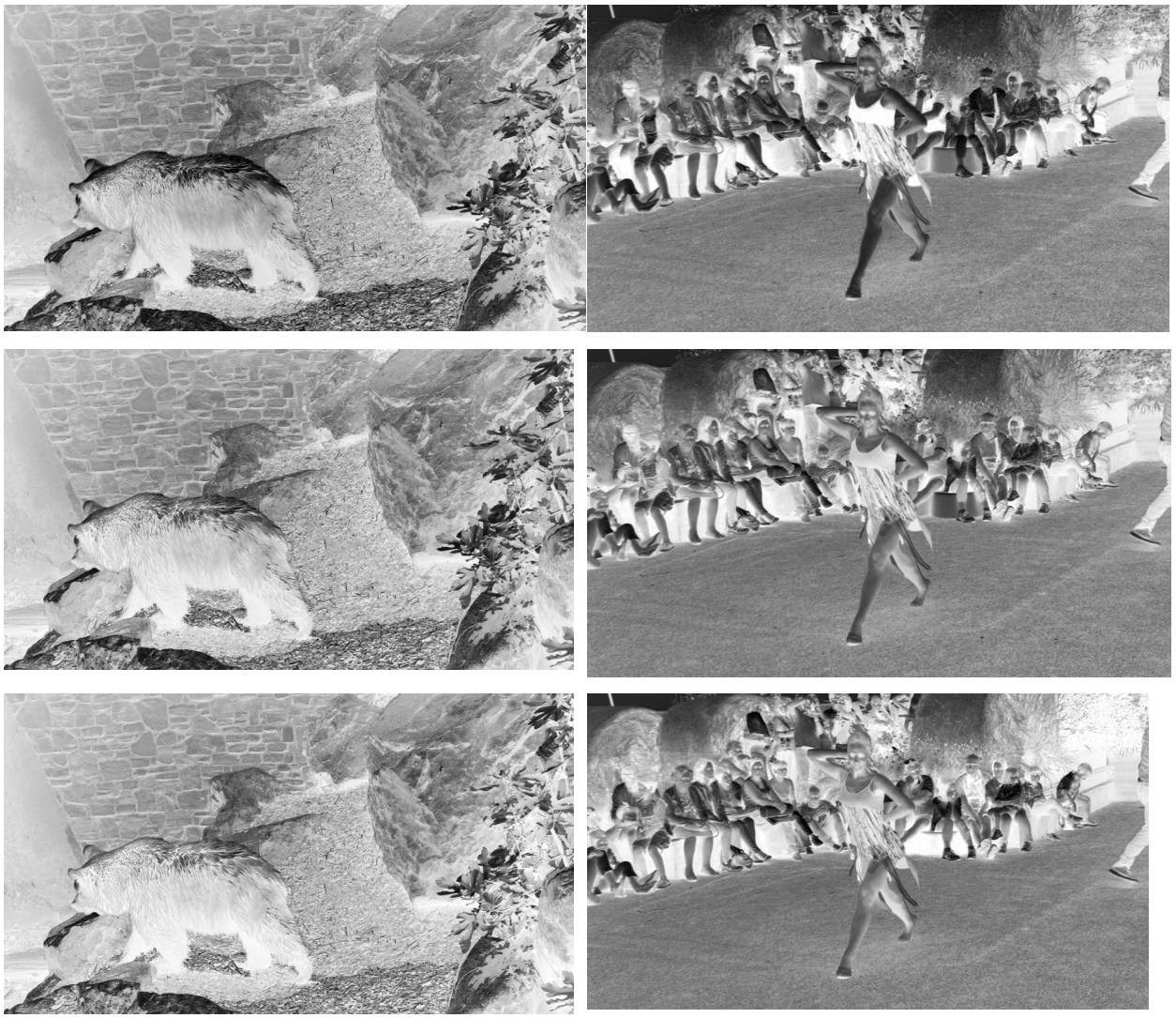
Approach and Procedure:

For the second part, simple transformations are required to transform RGB to CMY. Cyan is the complement of Red, Magenta is the complement of Green and Yellow is the complement of Blue. An image can be easily converted from RGB to CMY by subtracting each color value of each pixel from the total number of colors of each

channel. This can also be done by normalizing the values of R, G and B and subtract them from 1 to get the values of the C, M and Y component.

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

Experimental Results:



Figures (4-9) (starting from top left to right) Bear_cyan, Dance_cyan, Bear_magenta, Dance_magenta, Bear_yellow, Dance_yellow

Discussion:

Depending on the application, the images can be represented using the RGB channels or the CMY(K) channels. RGB colors are used to display images in any electronic device such as a phone, television etc. The primary reason why most of the images are represented in the CMY(K) channels these days in printing is because of the reason that RGB colors absorb the colors Red, Green and Blue and what we actually see are the colors Cyan, Magenta and Green. It makes more sense to represent the images in Cyan, Magenta and Yellow, so that the colors we see are Red, Green and Blue.

(1.b) Image Resizing via Bilinear Interpolation

Abstract and Motivation:

Image resizing is a very important application in image manipulation. For example, when you want to properly fit an image in a power-point or a word document, you need to resize the image. Image resizing is nothing but, change the dimensions of an image from its original values to the desired values. Usually, increasing the dimension of the image reduces the quality of the image.

Approach and Procedure:

Some of the common algorithms for image resizing are nearest neighbor interpolation, bilinear interpolation, etc. The algorithm for bilinear interpolation is very simple. The main idea is to calculate the ratio of stretching or shrinking of the image. The objective in the given question is to change a 512 x 512 image to a 650 x 650 image via bilinear interpolation. In this case, the ratio of stretching is given by 511/649. This ratio implies that, 512 pixels are being stretched to form an image having 650 pixels. The first and last pixels of each row and column of the new image are same as the original image. The second pixel of the resized image will be bilinear interpolated with the first and the second as well as the pixels above and below required pixel position. Since the number of pixels in each row/column of the original image is lesser than that of the resized image, we need to find the association of the new intermediate pixel with four of the neighboring

pixels. This is why we interpolate the 4 neighboring pixels to get the pixel in the new image. The formula for bilinear interpolation is given by,

$$F(p', q') = (1 - \Delta x)(1 - \Delta y)F(p, q) + (\Delta x)(1 - \Delta y)F(p, q + 1) + (1 - \Delta x)(\Delta y)F(p + 1, q) + (\Delta x)(\Delta y)F(p + 1, q + 1)$$

Here, $F(p', q')$ is the pixel of the resized image which has to be found. Δx and Δy are the correlation factors which change for each and every pixel based on their closeness to all the four neighboring pixels.

Experimental Results:

Fig 10 Airplane_scaled (650x650)

Discussion:

As we can see from the image, bilinear interpolation works well for image resizing. Comparing the original image and the resized image we can observe that the resized image

is slightly lower in clarity. One of the main problems with bilinear interpolation is that, the sharpness of the resized image is very much reduced when compared to the sharpness of the original image. To overcome this problem, other methods for resizing such as bicubic, etc., have to be used. The main advantage of bilinear interpolation is that it is computationally less intensive when compared to the other methods of Image resizing.

Problem 2: Histogram Equalization (40 %)

(a) Histogram Equalization (Basic: 12%)

Implement two histogram equalization techniques:

- Method A: the transfer-function-based histogram equalization method,
- Method B: the cumulative-probability-based histogram equalization method

to enhance the contrast of the *Desk* image in Figure 5 below.

(1) Plot the histograms of the red, green and blue channels of the original image. The figure should have the intensity value as the x-axis and the number of pixels as the y-axis.

(2) Apply Method A to the original image and show the enhanced image. Plot the transfer function for each channel.

(3) Apply Method B the original image and show the enhanced image. Plot the cumulative histogram for each channel.

(4) Discuss your observations on these two enhancement results. Do you have any idea to improve the current result?

Figure 5: Desk image

(b) Image Filtering – Creating Oil Painting Effect (Advanced: 16%)

An exemplary oil-painting effect for the *Barn* image is shown in Figure 7. This effect can be implemented as a filter with the following two steps.

(a) Barn **(b) Barn with a reduced color set**

Figure 6: Barn images with a full and a reduced color sets

Step 1: Quantize all colors of the input color image, denoted by I_o, into an image containing only 64 colors, denoted by I_r. In other words, each channel should have only 4 values. The original and truncated *Barn* images are shown in Figs. 6 (a) and (b), respectively.

Step 2: For each pixel of image I_r, select the most frequent color in its *N*x*N* neighborhood (*N* is an odd number, usually ranging from 3 to 11), as the representative color for this pixel in another output image denoted by I_s. The result of N = 5 is shown in Figure 7.

Figure 7: Barn with the Oil-Painting Effect

Implement and apply the oil-painting filter to the *Star_Wars* and *Trojans* images as shown in Figs. 8 and 9.

(1) Show the 64-color version of both images by following Step 1. Discuss how you choose the threshold.
(2) Implement the oil painting process as described in Step 2 with several different values of *N*. Which *N* gives a better result? Discuss your observations.
(3) What happens to this filter when the input image has 512 colors instead? Show the corresponding results for both images and explain your observation.

Figure 8: Star_Wars

Figure 9: Trojans

(c) Image Filtering – Creating Film Special Effect (Advanced: 12%)

Design an algorithm to achieve the film special effect. An example is shown in Figure 10. Describe your algorithm step by step and apply your algorithm to test image *Girl* given in Figure 11. Show the film special effect result in your report.

Hint: investigate the color channel relationship between the input and output images in Figure 10, which will give your ideas about the film special effect.

Figure 10: An exemplary film special effect: the input (left) and the output (right).

Figure 11: Girl

(2.a) **Histogram Equalization**

Abstract and Motivation:

Histogram equalization is mainly used for enhancing the image. In some images, the color concentration may be very close to the higher color ranges, i.e., greater than 200 or may be very closer to the lower ranges say, lesser than 50. In this case the it is difficult to visualize all the regions of the image as most of them will be completely dark or completely light. This issue is called as low contrast. Contrast of an image is defined as the difference between the maximum pixel intensity and the minimum pixel intensity. Images with low contrast are images which cannot be visualized completely. Histogram equalization is used to rectify this problem and increase the contrast. In typical low contrast images, the color range is small and mostly spread on the higher color regions.

Histogram of an image is a graphical visualization of the colors in an image. The x-axis contains the colors from 0-255 of each channel and the y-axis represents the intensity of each color, i.e., how many times each color is repeated in that image.

Approach and Procedure:

There are two main methods for histogram equalization.
Method A: Transfer function based histogram equalization
Method B: Cumulative probability based histogram equalization

In both the methods, the first step is to find the histogram of the image. After finding the histogram, we need to find the cumulative histogram which has the range from 0-1. In transfer function based histogram equalization, we need to find the transfer function for each channel of the image. To find the transfer function, the cumulative histogram has to be multiplied by 255 first. Then to apply this to the original image, each color in the original image must be replaced by the color in the transfer function. For example, if the value of the color 0 in the transfer function of that channel is 10, all the pixels having the value 0 for that particular channel in the original image have to be replaced by the color 10. This process has to be repeated for all the colors in each channel. The main idea is that, the color in the original image has to be replaced by the color shown in the transfer function.

In Cumulative probability based histogram, all colors are equally divided. For example, if the resolution of the image is 100x100 and there are 255 colors in total for each channel, then each color must be repeated for (10,000/255) times. So, if this procedure is repeated for all channels, the image will have equal pixels for all colors in the image. This aim can be achieved by implementing the bucket filling algorithm. In the bucket filling algorithm, the total intensity that each pixel must have is calculated. After that, it is made sure that each pixel has that intensity. Suppose the color 0, has 100 pixels less than the intensity that it is needed to have, 100 pixels having the color value 1 will be taken and replaced by the color 0. This process is repeated for all the pixels. If a pixel has a higher or a lesser intensity than required, the intensity of the neighboring colors is shared and in this way, all the pixels will have the equal intensity.

Experimental Results:

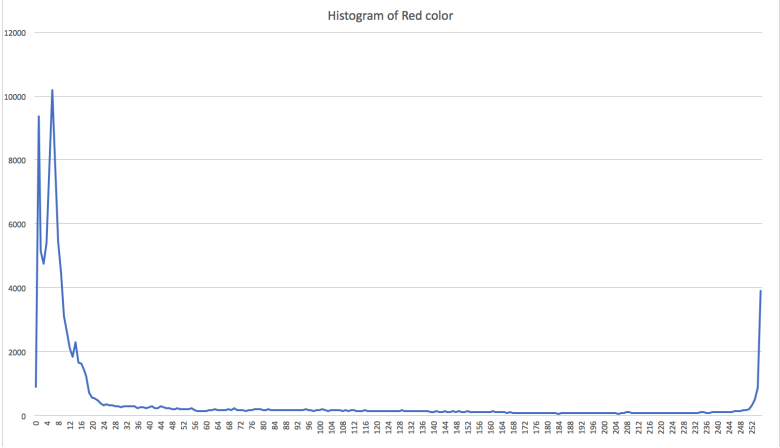


Fig 11 (Histogram of Original Image)

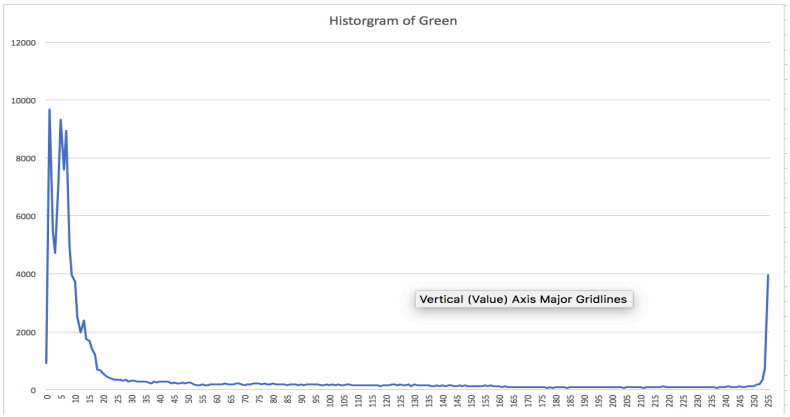


Fig 12 (Histogram of Original Image)

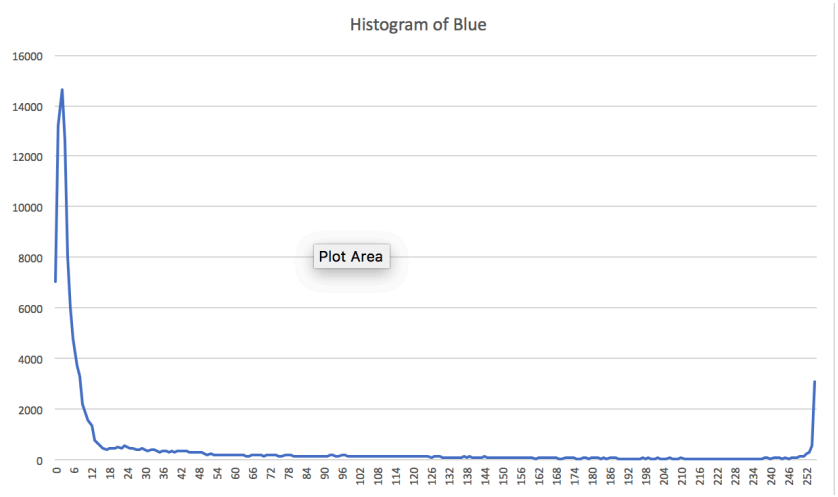


Fig 13 (Histogram of Original Image)

Fig 14 (Transfer function based Histogram equalization)

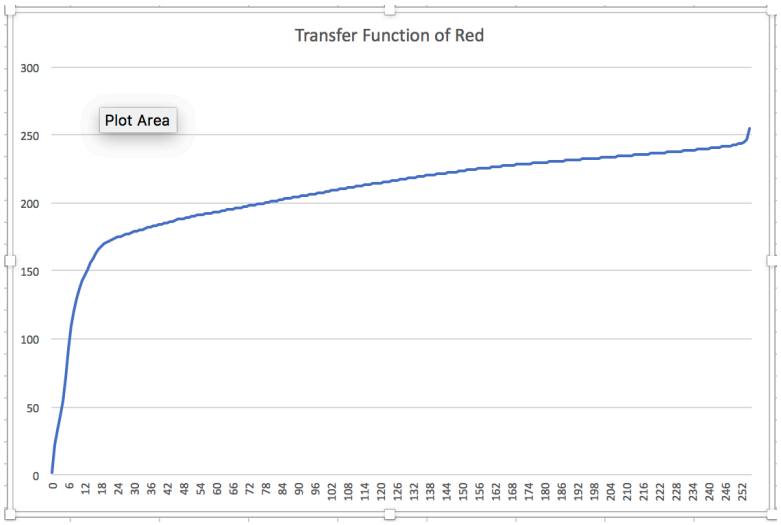


Fig 15 (Transfer function of Red Channel)

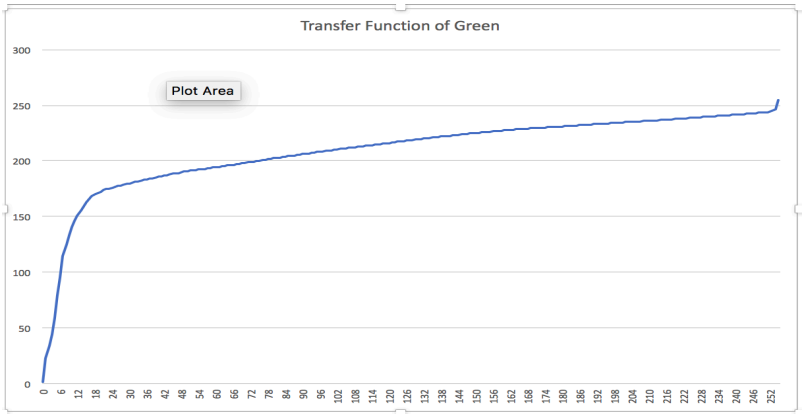


Fig 16 (Transfer function of Green Channel)

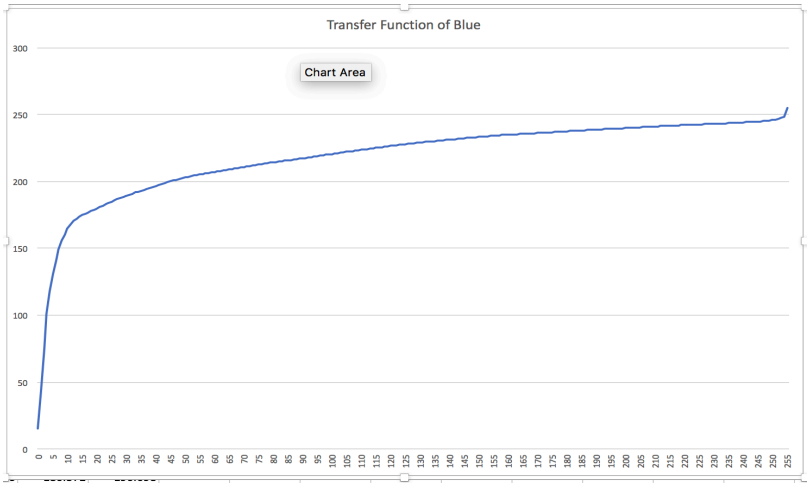


Fig 17 (Transfer function of Blue Channel)

Fig 18 (Cumulative probability based histogram equalization)

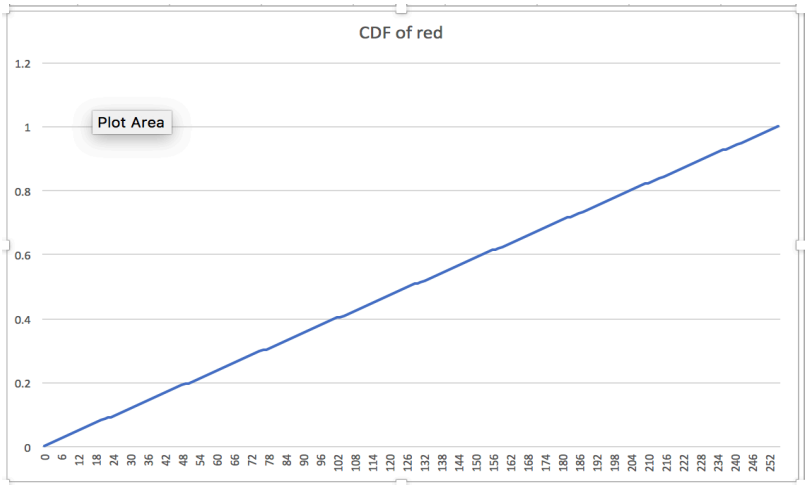


Fig 19 (CDF of Red Channel)

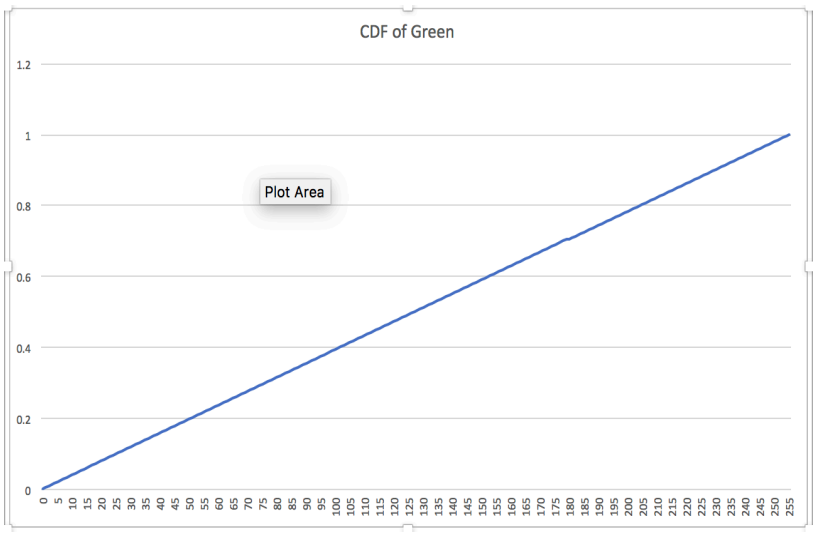


Fig 20 (CDF of Green Channel)

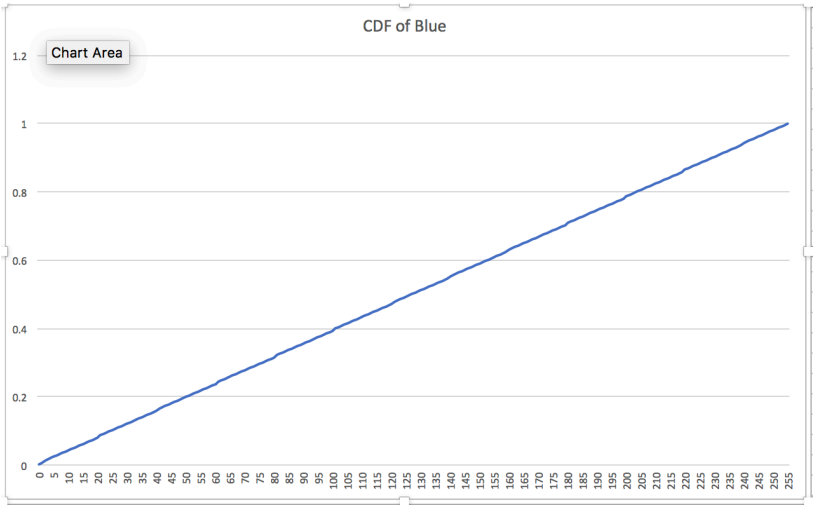


Fig 21 (CDF of Blue Channel)

Discussion:

Figures 14 and 18 are the enhanced images obtained from the Transfer function based method and the Cumulative probability based histogram equalization methods. We can observe the fact that when compared to the original image, these images are very much different and we can see the regions of the image which were completely dark in the original image.

From the histograms of each channel of the original image, we can see that the colors are mostly concentrated towards the lower intervals. All the three channels are concentrated between the colors 0 and 25. This is the reason why the image has a poor-

quality due to the low contrast. Only if the contrast is high, we can completely visualize all the regions of the image. By saying that contrast is high, we are saying that the color range in the image is wide, i.e., between 0 and 255 instead of a very small range in the original image.

The transfer functions of each channel for histogram equalization is given in figures 15,16 and 17. The colors in the original image have to be replaced by the colors given in the transfer function. For instance, if the transfer function is considered as g(f), then the color value 'f' in the original image will be replaced by the color value 'g(f)' specified by the transfer function. Another important thing to be observed from the transfer functions of the three channels is that all the three channels have a similar transfer function which implies that for every channel, the colors are replaced by almost similar transfer function values.

The image obtained by both the methods are almost similar. From the cumulative distribution function of the output image, we can observe that the distribution is uniform, which means that all the pixel values have an equal intensity. Though the transfer function based method and the cumulative probability based method are different approaches to enhance the image, the images obtained are very similar because of the fast that both of these images increase the contrast significantly.

We can see that the enhanced images have green patches on the top left corner. The histogram equalization techniques we have followed are global in nature, i.e., we have considered the histogram of the entire image and then modified the histogram. There are advanced histogram equalization techniques such as the adaptive histogram equalization which has the same algorithm, but it performs the histogram equalization for a local section of the image. In this, we compute the histogram for each distinct section of the image, and then implement the transfer function based method or the cumulative probability based method. If we use this method, there is a possibility that the green patches may disappear because, the regions which are dark in the original image will still be left dark because there is actually nothing in those regions of the image. Thus, adaptive histogram equalization can give us better results when compared to the global histogram equalization.

(2.b) **Image Filtering: Creating Oil Painting effect**

Abstract and Motivation:

The purpose of this problem is to create an oil paint effect to the original image. Though that is the final effect, reducing the color set and implementing the filter is the underlying main objective. Reducing the color set is known as quantizing the image. Previously, low cost displays did not have the ability to display the entire color set. The devices which were able to display all the colors were very costly. Hence the solution for this problem was to decrease the color range so that the image could be displayed using the low-cost devices. Image filtering is one of the most important steps in Image processing as it is the major pre-processing step. Oil painting filter effect also has the basic idea of filtering. The same steps can be followed for denoising an image except for choosing the replacement color. Another important reason why Image filtering is pretty famous because it is used to create different effects for the images, hence making it more attractive and changing the routine from a monotonous normal image to a new and creative image.

Approach and Procedure:

There are two main steps needed to create the oil painting effect.

Step 1:

Quantizing the image

Quantizing the image means reducing the color set of the image. For example, an image may have 256 colors per channel. By this step 1, we will be reducing the number of colors from 255 to any desired value such as 4, 8, etc., If it is a 4 color per channel image, then the total number of colors will be 4x4x4 which is 64 colors.

Step 2:

Applying the filter

After reducing the color set of the image, the next step is to choose a window. The oil filtering effect follows the 'mode' filter effect. If we choose the window to be NxN, for every pixel N^2 pixels will be considered for applying the filtering effect. Consider the window to be 3x3. What this step basically does is, for every pixel coming in, it takes the

neighboring 9 pixels and calculates the most frequent color for all the three channels. Then, it replaces the color for that pixel with the color that is most repeated. When this step is repeated for all the pixels, the oil painting effect is obtained. If the window size is 5, then the 25 neighboring pixels will be considered to find out the most repeating color.

The oil painting effect is obtained by the successful completion of these two steps. Another important thing is to note is that, when it comes to the boundary pixels implementing the window will be difficult. To overcome this issue, the very first step would be to extend the boundary to satisfy the required window size. The boundary can be extended by mirroring the boundary rows and columns based on the window of filtering needed.

Experimental Results:

Fig 22 (Reduced Color set image of Star Wars)

Fig 23 (Reduced color set image of Trojans)

Fig 24 (Star Wars Oil Painting Window: 3x3)

Fig 25 (Star Wars Oil Painting Window: 5x5)

Fig 26 (Star Wars Oil Painting Window: 7x7)

Fig 27 (Star Wars Oil Painting Window: 9x9)

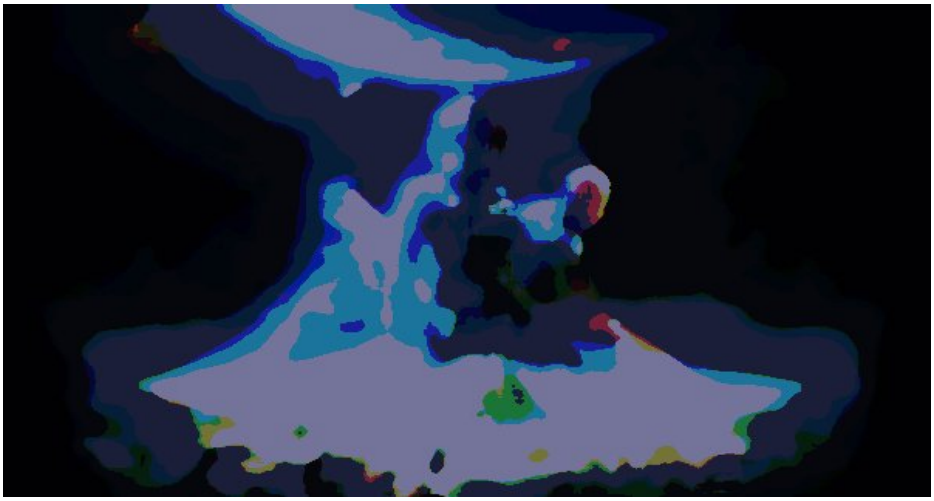


Fig 28 (Star Wars Oil Painting Window: 11x11)

Fig 29 (Trojans Oil Painting Window: 3x3)

Fig 30 (Trojans Oil Painting Window: 5x5)

Fig 31 (Trojans Oil Painting Window: 7x7)

Fig 31 (Trojans Oil Painting Window: 9x9)

Fig 32 (Trojans Oil Painting Window: 11x11)

Fig 33 (Star Wars reduced color set: 512)

Fig 34 (Trojans reduced color set: 512)

Fig 35 (Star Wars Oil Painting with 512 colors Window: 3x3)

Fig 35 (Trojans Oil Painting with 512 colors Window: 3x3)

Fig 36 (Star Wars Oil Painting with 512 colors Window: 11x11)

Fig 36 (Trojans Oil Painting with 512 colors Window: 11x11)

Discussion:

Figures 21 and 22 show the reduced color set images of Star Wars and Trojans respectively. The original image has 256 colors per channel and the reduced color set image has only 4 colors per channel. The quantization colors are chosen with the help of the histogram of the image. If the desired number of colors for the output image is 4, then the histogram of original image is divided into 4 equal bins having equal pixel intensities among them. For instance, if the dimension of the image is 400x300, the bins have to be chosen in such a way that each bin has 30,000 pixels in it. The first step is to choose the color at which the 30,000 mark is reached. After that, the weighted average of that particular bin is taken to estimate the first quantization color. This process is repeated for the next three bins to find the other three quantization colors. After finding the quantization color, all the colors in each bin are replaced by the quantization color of that corresponding bin. The same procedure is repeated for all the 3 channels and the reduced color set image is obtained.

The oil painting filter effected obtained with different window sizes are shown in the figures 24-32 for the Star-wars and the Trojans image. The oil painting effect reduces the sharpness of the image drastically. If we observe the Star-wars image, the small details

in the black background have vanished. As the window size increases, the sharpness of the image reduces further. We can clearly visualize the oil painting effect if the window size is really high. This is what the 'mode' filter does. It just replaces each pixel with the most repeated color of the neighboring pixels. If we consider more number of pixels for calculating the mode, then the neighboring pixels have the same color and this causes the minute differences to disappear basically reducing the sharpness of the image.

The images created with smaller window sizes give a better effect. When the smaller window images are compared with the original image, the details in the original image are replicated to an extent even after the application of the oil painting effect. In the case of larger window images, most of the details are missing when compared to the original image. Hence, the images created with a smaller window size are better.

The images having 512 colors are shown in figures 33 and 34. The image looks more colorful when compared to the image having 64 colors due to the obvious reason that there are more number of colors in this image. When this image is subjected to the oil painting filter with window 11x11, it is better than the same window applied to the 256 colors image. This also due to the increased number of colors. Thus, it is better to apply oil painting filter for an image with the higher number of reduced colors. This depends totally on the image. There is another factor which also plays a very important role in the oil painting filter effect, the range of colors in the original image. If the image is very colorful, i.e., less regions with dark colors, then the oil painting filter looks much better. This can be observed from the oil painting effects applied to the Star Wars image and the Trojans image. The Trojans image looks much better with the oil painting effect since there are no dark regions unlike the Star Wars image which has a lot of dark regions. Hence, for the oil painting image, the number of colors must be reduced and it will be better if there are no large dark regions in the image.

(2.c) **Image Filtering: Creating Film Special Effect**

Abstract and Motivation:

The main aim of this problem to understand the film special effect illustrated in the question and apply it to the Girl image given. This can be used to create special effects to

images. This can be very helpful in night vision where changing the color may help to understand the image more better. Different colors can reveal different details in that image. For example, if we have a very dark image, inverting it might help us get some information about the picture than it previously was. This can be used to convert a negative image to the colors that we can distinguish.

Approach and Procedure:

The first step in this was to understand the relation between the original and the film image given. We can easily understand that the image has been flipped from right to left. This can be implemented easily by swapping the columns of the image about the central column. So, the last column goes to the first and the first column goes to end. By repeating this for all the columns, the image can be easily flipped.

From the given Film image, we can easily observe that the colors have been inverted. So, the next step is inverting the colors. The colors can be inverted easily by subtracting each color from the total number of colors present. These 2 steps can be implemented very easily as well.

The last and the final step would be to understand the relation of the histograms of the film and the original images. From the histograms, we can observe that the color range in the film has been shrunk to a different interval from 0-255 for all the three channels. We need to identify the range of the colors from the histogram of the film. After finding the new range of the color, then the next step would be shrink the color range of the inverted original image. Consider the new shrunk image interval to be (a, b). So, we must convert the (0, 255) interval to (a, b). This can be done with the help of the formula:

$$New\ color\ value = a + Old\ color\ value * (b - a)/255$$

By finding the color ranges for all the three channels, and implementing this formula we can get the film effect.

Experimental Results:

Fig 36 Girl with the special film effect

Discussion:

As mentioned above, the first step is to find out the value of the shrunk color ranges for each channel. We can find this with the help of the histogram of the given film image. From the histograms we observe that, the color red range has shrunk from (0-255) to (95-255), the color green's range has shrunk from (0-255) to (31-198) and the color blue's range has shrunk from (0-255) to (15-181). The main difficulty was to identify the new color ranges. We can also use a computer program to find out the shrunk image color range values. After finding the color ranges, we can use the above-mentioned formula to change the color range of the original inverted Girl image to the new color range value. The Girl film image obtained by using this method is given in Fig 36. The algorithm is given below.

Step 1:

Flip the image from right to left. This can be implemented by flipping the columns of the image.

Step 2:

Inverting the colors of the image. This can be done by subtracting each color by the total number of colors.

Step 3:

Shrink the color range of each channel, by first identifying the shrunk color range and then shifting the original inverted color range to the new color range obtained from the film image. The following formula can be applied to change the color range to the desired range.

$$New\ color\ value = a + Old\ color\ value * (b - a)/255$$

(a, b) is the desired color range and the input color range is $(0,255)$

This is how the film effect is applied on the original girl image to get the girl film image.

Problem 3: Noise Removal (30 % + 10 %)

In this problem, you will implement a set of denoising algorithms to improve image quality. You can us the PSNR (peak-signal-to-noise-ratio) quality metric to assess the performance of your denoisin algorithm. The PSNR value for R, G, B channels can be, respectively, calculated as follows:

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

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

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

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

Max: Maximum possible pixel intensity = 255

(a) Mix noise in color image (Basic: 15%)

Perform noise removal on a color image corrupted by a "mix" type of noise. The original and noisy *Len* images are shown in Figure 12.

Figure 12: The original and noisy Lena images.

1. Identify noise types in the noisy *Lena* image, and answer the following questions.
(1) Do all channels have the same noise type?
(2) Should you perform filtering on individual channels separately for both noise types?
(3) What filters would you like use to remove mixed noise?
(4) Can you cascade these filters in any order? Justify your answer.
(5) Discuss the effect of different filter window sizes.

2. Get the best results in removing mixed noise. Include the following in your report:
(1) Describe your method and show its results.
(2) Discuss its shortcomings.
(3) Give some suggestions to improve its performance.

(b) Principal component analysis (PCA) (Advanced: 15%)

Principal component analysis (PCA) is a dimensionality reduction algorithm, yet it can also be used as a tool for noise filtering. In this part, please read carefully the reference paper [1].

1. Explain why PCA can be used as a filtering approach for noisy data. What do the components mean and how to choose the number of components in image denoising.
2. Describe and implement the patch-based local PCA (PLPCA) in [1]. **DO NOT** use any code from the Internet or other sources, as it would be considered as plagiarism.
3. Apply the PLPCA to *House_noisy.raw* (Gaussian noise, $\sigma = 25$) in Figure 13. Optimize different parameters, e.g. patch size, the number of components, etc., and discuss their effect on the denoising process. State the best PSNR and your optimized parameters in your report.
4. Apply the approach in part (a) to *House_noisy.raw* and compare the performance of the PCA approach with those of filters used in part (a). Please explain the advantages of the PCA approach and why. **DO NOT** quote statements directly from [1] or any other online source. Try and explain in your own words. Reports and source codes are subject to verification for any plagiarism.

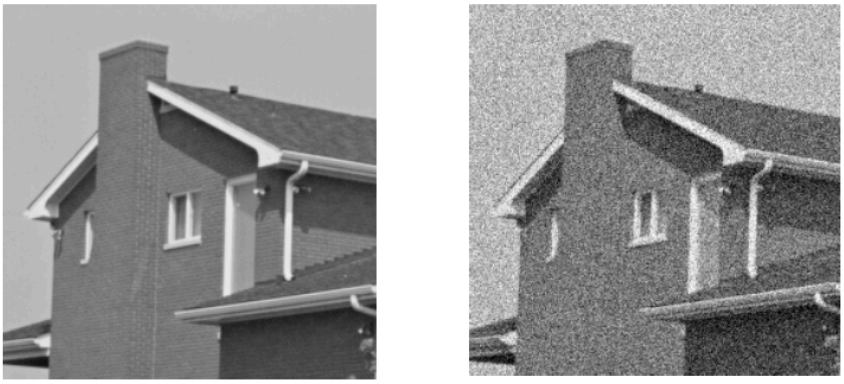


Figure 13: The grey-level original and noisy House images

(c) Block matching and 3-D (BM3D) transform filter (Bonus: 10%)

In this part, you will get familiar with another state-of-the-art denoising algorithm proposed in [2].

1. Please explain the BM3D algorithm in your own words, and implement the BM3D filter (Write your own code or use any available online source code but include the source in your reference) to denoise the noisy image: *House* (Figure 13). Discuss the effects of several tunable parameters on the denoising result.

Note: It is recommended that you use the code provided by the authors on their website [3]. Their code is written in MATLAB; so that it is okay to use MATLAB for this part (You still qualify for 5% bonus points if you have used C/C++ everywhere else).

2. Why does the author utilize block matching instead of other clustering methods like k-means? What is the motivation of step 2 (block matching and Wiener filtering) and compare the denoising performance with and without step 2.
3. How would you classify BM3D - spatial domain filter, frequency domain filter, or both? Justify your answer.
4. Conduct qualitative performance comparison between the algorithms developed for Problem 3(b) and BM3D.

(3.a) **Mix Noise in Color Image:**

Abstract and Motivation:

The only enemy for any signal is noise. There are innumerable ways in which an image can be corrupted by noise. The noise can be acquired during transmission. In any image processing technique, the first step is to de-noise the image. This is very much needed in satellite imaging where the images collected by the geo-satellites are very noisy. Understanding the data itself, depends on how the image is being de-noised. Image de-noising is a vast field, having use in almost all the image processing devices. All the image processors in televisions, mobile phones, etc., have their inbuilt de-noising algorithms. In this question, we will be denoising the Lena image to make it look better and to decrease the PSNR between the original image and the de-noised image.

Approach and Procedure:

There are many methods for de-noising an image. Some of them are averaging filter, Median filter, Non-local means filter, patch based PCA filter, Block Matching 3D etc. The first step in any denoising algorithm is that, predicting the type of noises present in that image to some extent. Different filtering algorithms are used for removing different types of noise. The Lena mixed image mainly has salt and pepper noise and Gaussian noise. The salt and pepper noise is otherwise called as outliers as they have value 0 or 255 due to camera sensor faults. The Gaussian noise is Gaussian distributed noise, can be removed easily by applying the averaging filter to the image.

What this averaging filter does is, it calculates the mean of the neighboring pixels based on the filtering window. If the filtering window is 3x3, then the neighboring 8 pixels along with current pixel are taken and the mean is calculated. The mean color value is allocated to the current pixel. When this process is repeated for all the pixels, Gaussian noise can be removed to a significant level.

The median filter is used to remove the salt and pepper noise. These have the value 0 or 255 in each channel. The idea of the median filter is similar to that of the averaging

filter, except that instead of the mean color being allocated to the current pixel, the median color among the neighboring pixels is allocated to the current pixel.

Apart from just applying the filters, the order of cascading the filters also matters. For example, to remove an image having both salt and pepper noise and Gaussian noise, the median filter has to be applied first and then the averaging filter has to be applied. It is also better to identify the noise in each channel separately before applying it to all the channels. By trying different filters to different images, particularly different channels can help denoising the image to the maximum PSNR possible.

The PSNR can be calculated by the

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

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

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

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

Max: Maximum possible pixel intensity = 255

Experimental Results:

Fig 37

PSNR: 26.015 dB

Fig 38

PSNR: 26.0299 dB

Fig 39

PSNR:26.0271

Fig 40

PSNR: 26.2914

Discussion:

The different images obtained by implementing the averaging and the median filter with different window specifications for different images is given in figures 37-40. The order of the cascading done is, median followed by the averaging filter in order to remove the isolated noise first and then remove the Gaussian noise. The following table gives the PSNR values for different types of windows applied for the median and the averaging filter applied to different channels along with their corresponding PSNR values.

Median Filter Channels	Averaging Filter channels	Median Filter window	Averaging filter windows	PSNR (in dB)
Red, Green	Blue	3x3	3x3	24.854
Red, Green	Blue	3x3	5x5	25.3356
Red, Green	Blue	5x5	5x5	25.9826
Red, Green	Red, Green, Blue	3x3	5x5	25.4718
Red, Green	Red, Green, Blue	3x3	5x5	25.7417
Red, Green	Red, Green, Blue	5x5	3x3	25.5268
Red, Green	Red, Green, Blue	5x5	5x5	25.6991

Red, Green, Blue	Red, Green, Blue	3x3	3x3	26.015
Red, Green, Blue	Red, Green, Blue	3x3	5x5	26.0271
Red, Green, Blue	Red, Green, Blue	5x5	3x3	26.2914
Red, Green, Blue	Red, Green, Blue	5x5	5x5	26.0299

The image shown in Fig 37 is the image obtained by applying the median and the averaging filter for all the three channels with windows 3x3 for both the filters. This image has a PSNR value of 26.015dB. The image shown in Fig 38 is the image obtained by applying the median and the averaging filter for all the three channels with windows 5x5 for both the filters. This image has a PSNR value of 26.0299dB. The image shown in Fig 39 is the image obtained by applying the median and the averaging filter for all the three channels with windows 3x3 for the median filter and 5x5 for the mean filter. This image has a PSNR value of 26.0271dB. The image shown in Fig 40 is the image obtained by applying the median and the averaging filter for all the three channels with windows 5x5 for the median filter and 3x3 for the mean filter for. This image has a PSNR value of 26.2914dB.

Figure 37 and 38 have almost the same PSNR value but we can differentiate the sharpness values of these images. Figure 37 is sharper when compared to figure 38. Considering the figures 39 and 40, figure 39 is less sharp than the figure 40. This is mainly due to the application of different window sizes for different filters.

(1-1) All the channels do not have the same noise. The channels Red and Green have isolated noise and the channel Blue has Gaussian noise. This can be understood by applying the median filter only to the channels Red and Green and not to the channel Blue.

(1-2) By implementing different filters to different channels separately, the PSNR is reduced by a small value. Considering the PSNR value for the images in which the median filter is applied to only the channels Red and Green and the averaging filter to only channel Blue, and the image in which the median filter is applied to all the channels and the mean filter is also applied to all the channels the PSNR difference is about 1.14dB. Considering the former image with the image in which the median filter is applied to the channels

Red and Green, and the averaging filter to all the three channels, the PSNR difference is 0.6728dB. Thus, by implementing the filters to separate channels, the PSNR value can be slightly different. In this case, applying the filters to all channels gives a better result than by implementing the filter for the separate channels.

(1-3) As mentioned above, we can use the median and the averaging filters to remove the noises since the main types of noise are salt and pepper noise and Gaussian noise. By applying these filters to different channels and with different window sizes, different PSNR values can be obtained.

(1-4) No, we cannot cascade these filters in different orders. The first filter would be to remove the salt and pepper noise and the second filter would be to remove the Gaussian noise. If we apply the averaging filter followed by the median filter, then the averaging filter will not be able to filter properly, as the average color value is significantly affected by the presence of 0 or 255 color values. The PSNR value is also drastically affected if the order of the cascading is changed.

(1-5) When different window sizes are applied, the sharpness of the image reduces and the PSNR values increase slightly. This is because, when the window size increases, more number of neighboring pixels have to be considered for the averaging filter and when this is repeated for all the pixels, the sharpness of the image reduces.

The best method is applying the median filter with window 5x5 for all the channels and the averaging filter with window 3x3 for all the channels. This gives a PSNR value of 26.2914dB. When compared to the median and averaging filter being applied to the specific channels, this image has a higher PSNR value. When compared to the other images with different windows, this image is sharper and appears with less grains.

The algorithm followed to obtain the image in figure 38:

- Apply the median filter first to all the channels with a window size 5x5
- Apply the averaging filter to all the channels with a window size 3x3

Other filtering methods such as the Block matching 3D and Patch based PCA can be applied to improve the image quality by reducing the noise. Though the median filters and the averaging filters are the basic set of filters, they help reducing the noise level to a great extent and these filters are computationally very efficient. The main problem with these set of filters is, the sharpness of the image is reduced as it considers the neighboring pixels of the entire image. The details of the images are reduced on the application of the median and the averaging filters. Other techniques such as the non-local means can be used to reduce the PSNR value further, still retaining the sharpness of the image. If we use the other sophisticated and state of the art de-noising techniques, the image quality will still improve.

(3.b) **Principle Component Analysis (PCA)**

Abstract and Motivation:

Principal Component Analysis is one of the main Dimensionality Reduction algorithm which has many applications. Some of its main applications are present in Machine Learning, Data Mining, Noise Reduction etc., Principle component analysis is based on the concept of singular values and Eigen vectors. There are three main methods of implementing PCA for de-noising images. The first one is patch based global PCA, patch based hierarchical PCA and patch based local PCA. The difference in the algorithm of the three methods is very small, but the PSNR values and the computational times are very different. The Global PCA creates a single orthogonal axis by performing PCA on all the patches collected from the image whereas the local PCA creates multiple orthogonal axes by performing PCA on specific sections/quadrants of the image. Patch based local PCA is computationally intensive when compared to the global PCA but has better efficiency. Global PCA has only one axes for the entire image whereas local PCA has number of axes equal to the number of sub-sections in the image. We will see how the concept of dimensionality reduction can be used for noise removal in images using patch based Local PCA.

Approach and Procedure:

The first step in the Patch based Local PCA is to divide the image into a number of patches. The difference between patch based local and global PCA is that, the patches are taken all at a time in patch based PCA whereas in local PCA the image is divided into sub sections. After the first step, we need to find the covariance matrix. Covariance matrix is nothing but finding the co-relation of each patch with all the other patches in that particular section of the image in case of local PCA. Finding this covariance matrix is the most important step in PCA.

For this covariance matrix, we need to find the Eigen values and the Eigen vectors. So, any patch can be represented by the linear combination of these Eigen vectors and the Eigen values. Considering there exists 'n' Eigen vectors and assuming the noise is to be uniformly spread over the entire image, the patches are projected over k orthogonal axes ($k<n$). By doing this projection of higher dimension onto a lower dimensional subspace, the low singular values (mainly noise) are eliminated. In order to make this better, it is better to project the images on the axes with the highest variances, i.e., the axes with the highest variance values. This is how the number of components are chosen. Local PCA is a better algorithm than the global PCA because of the fact that the global PCA will have only one axes which is formed based on the entire image, whereas the local PCA will have multiple axes based on the number of the sections that the image has been divided into. Local PCA definitely has more computational intensity when compared to the global PCA, but has better results. There are 3 main parameters for the patch based local PCA algorithm namely size of the patch, size of the search window and the threshold value. By modifying these values, we can obtain different PSNR values.

Experimental Results:

Figure 41

PSNR: 31.15

Fig 42

PSNR: 31.38

Fig 43

PSNR: 14.6675dB

Discussion:

The image in Figure 41 has a sigma value of 25, searching zone of 22, size of the patch of 14 and the threshold factor of 2.75. The image in figure 42 has a searching zone of 40, size of the patch as 20 and a threshold of 2.75.

PCA is an algorithm which reduces the dimensionality of the system. This can be used for noise removals in images since most of the images have a uniform noise reducing the dimensionality of image can help eliminating the noisy components of the image. The dimensionality can be reduced by reducing the dimension of the Eigen vectors. Components of an image are called as the orthogonal axes needed to represent the image. By reducing the components, the dimensionality of the images can be reduced. These components can be found out by the dimension of the Eigen vectors. Those Eigen vectors having very low singular values can be removed to reduce the number of components. Those lower singular values are caused by the noise which is spread uniform throughout the image. Hence by reducing the number of components, we can essentially remove the components corresponding to the noise.

The house image is processed using the patch based local PCA code given in [1]. The default window size was 22 and the patch size was 14 and this image had a PSNR value of 31.15dB. Window size is based on the number of local patches we need to create for the local PCA algorithm. The patch size corresponds to the pixels in each patch, basically deciding the number of patches for the entire image. By modifying these values, the PSNR value can be improved. When the patch size was increased to 20 and the window size was increased to 40 the PSNR was found out to be 31.38dB. Although the PSNR value increased, the computation time increased as the search window size increased.

Optimized values for highest PSNR:

Searching Window size: 40

Patch Size: 20

The house image de-noised by the use of mean and the median filter is shown in figure 43. We can see that the PSNR value is 14.6675dB which is very low when compared to the PSNR value obtained with the help of local PCA which is 31.38dB. The mean and the median filtering methods are just linear filters whereas the PCA filters based on the spatial domain as well, i.e. it considers the correlation of each patch not just with the neighboring patches but also with the patches in that local region/or all the patches in the image as well. This is the main reason why PCA works better than the linear filtering methods. Thus, the main advantage of using PCA is that, it filters on the spatial dimensions as well. Another important advantage of PCA is that, no prior information is required to figure out the orthogonal axes. The principal components (lesser in number than the original number of components) can be calculated with just the noisy image. Thus, PCA does not require any prior knowledge of the image as well as it gives almost 100% higher PSNR when compared to the linear filtering methods.

(3.c) **Block Matching and 3D (BM3D)**

Abstract and Motivation:

The Block matching 3D is a state of the art technique in image processing to remove the noise. This is the best filtering method available for removing Additive white Gaussian noise. This algorithm serves very well until the noise levels are really high. As the name

implies, the image is divided into numerous blocks and then taken for processing. There are two main steps in BM3D. The first step follows the hard thresholding method for the transformation from 2D to 3D and the second steps uses the Weiner Filtering for transformation from 2D to 3D array. The main difference between other filters and BM3D is that, the image has to be de-noised using step 1 of the filter before the actual BM3D (step 2) can be implemented. The aim of this question is to, understand the BM3D algorithm and then implement it for the noisy image given in the question.

Approach and Procedure:

The image is divided into a number of blocks using which the reference blocks are created and then used in the process of converting the 2D array of patches into 3D containing one of the values as similarity between each block with the reference blocks and hence the name, Block Matching 3D. There are 2 main steps in BM3D.

In step 1, the image is divided into a number of blocks. Each current block is then compared with the reference blocks. This is why this algorithm is called as Block Matching. After all the blocks are compared, a 3D array is created by putting the matched blocks together. After this, hard thresholding is applied to convert the coefficients into the transform domain. The lower coefficients are all eliminated to discard the noisy components.

After the conversion into transform domain, to estimate the new values of each pixel, the weighted average of all the stacks in which this particular block is present is taken. This is repeated for all the blocks and the new slightly de-noised image is formed. This image is now taken for step 2, the important step of Block Matching 3D.

In step 2, the image is again divided into blocks and the same logic as in step 1 is followed to create the groups of each block. The only difference between step 1 and step 2 is that, instead of hard thresholding for converting the blocks into the transform domain, Weiner filtering is used to convert the blocks into the transform domain. After this, the weighted average is taken to estimate the pixel values by finding out the relation of each block with the corresponding reference blocks in which that particular block is present. This procedure is repeated for all the blocks and the de-noised image is obtained. This is the explanation for BM3D algorithm. The code is taken from the source [2].

Experimental Results:

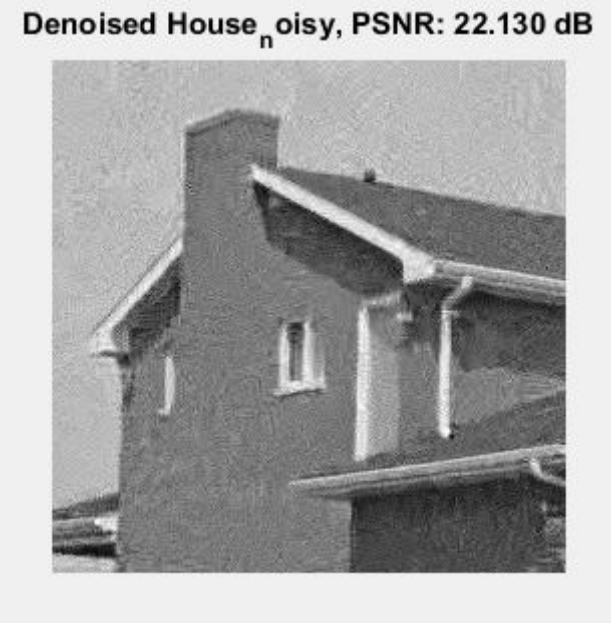


Fig 44 BM3D image

Discussion:

The image obtained after filtering by BM3D has a very good PSNR value. The image itself looks noiseless. There are a lot of parameters for the BM3D filtering such as the threshold values for the hard thresholding step and Weiner filtering step. Low values of thresholding (distance between two of the similar patches) help us keep a limited number of patches for block matching. However, if this number is too small, then the filter may produce erratic results as there would be less number of groups of each block. Another important parameter is choosing the size of the patches. The size of the patches must be relatively small for low noise images and must be very high for very noisy images otherwise most of the information would be lost in the noise itself. These are some of the parameters of BM3D and setting them properly can yield better results.

The main difference between block matching and other clustering methods is that, in block matching a block may be present in multiple number of blocks. Whereas in other clustering methods, if a particular block is assigned to one group, it cannot be present in any other group. In this case the weighted average of the pixel cannot be obtained which

reduces the quality of the other clustering based noise filter methods. This is the main difference between BM3D and other clustering based noise filtering methods.

Step 2 in the BM3D algorithm is the most important step of the algorithm. Weiner filter is used to convert the blocks into the transform domain using the energy spectrum of the image. The denoising image without the step 2 just gives us the image with just the transformation just based on the hard thresholding. Without the Weiner filtering step, there will be no transformation based on the energy spectrum and hence it will be similar to the non-local means or the k means clustering methods. Step 2 Weiner filtering is what makes the BM3D unique from all the other algorithms. Also, the idea of converting the 3D array to a sparse array in the transform will be lost.

BM3D can be classified as both a spatial domain filter and a frequency domain filter. Other algorithms such as Non-local means would just simply average the similar blocks. But in BM3D, this step also includes the spatial dimensions. Hence, one of the steps of BM3D also include the spatial domain filtering. The 3D stacks are converted into the transform domain, which basically has the frequency components being present. Hence, this step involves the frequency domain filtering. Thus, BM3D is both a spatial domain and a frequency domain filtering.

When compared to the image obtained from the Patch based local PCA, the quality of the image obtained from BM3D is less. This is mainly due to the fact that the BM3D is not that computationally intensive as compared to the PCA. The patch based local PCA took about 1.8 seconds whereas the BM3D just took 1.2 seconds. This is a very important thing to note as in fast applications, the computational intensity also plays a role. Hence for applications in which the PSNR is not that much important but the speed of the program is important, BM3D can be used. If the filtering is more important than the speed of execution then the PCA can be used. BM3D can also be optimized more by changing the other parameters given.

REFERENCES:

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

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