EE 569: Digital Image Processing
Homework #1

Faiyadh Shahid
Student ID: 4054-4699-70
fshahid@usc.edu

September 19, 2015

Problem 1

Pixel Manipulation is an important part of solving image processing problems. The concept of manipulating pixels rises from the need to resize or apply a color filter to the image. In this problem, two examples of pixel manipulation problems will be discussed: one of which includes image resizing and the other is related to converting a grayscale image to color image.

(a) Image Resizing via Bilinear Interpolation

Motivation

Image Resizing is used in manipulating the size of the image from a particular aspect ratio to a similar or different aspect ration. There are various techniques that have been developed by the community. One of them includes bilinear interpolation. The importance of such idea can be found built-in within the popularly used software called Adobe Photoshop. An example is attached in the following which shows that bilinear interpolation is a part of Photoshop image resizing.

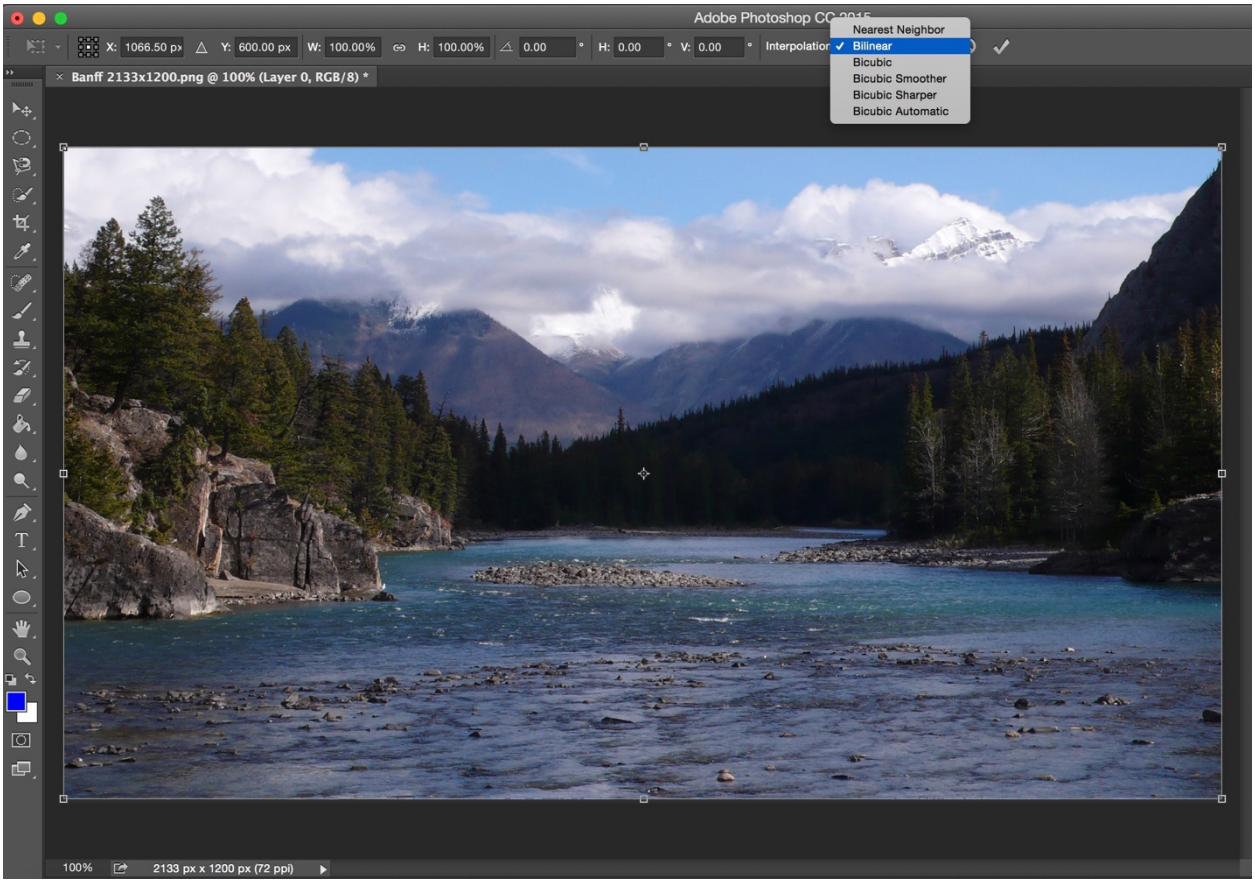


Figure 1: An example of Image Processing Idea in action in Adobe Photoshop

Method

The approach used in Bilinear Interpolation comes from the concept of Interpolating a point on a given straight line. Since we are dealing with two dimensions (2-D), the idea extends to finding a point in a two dimensional real space. In this particular problem, I would like to acknowledge the assistance of Professor Jay Kou in providing us with the intuition and a lecture note [1] by Dr. Richard Alan Peters II for providing the mathematical concept that can be employed for the idea. The approach employed can be briefly explained using Figure 2.

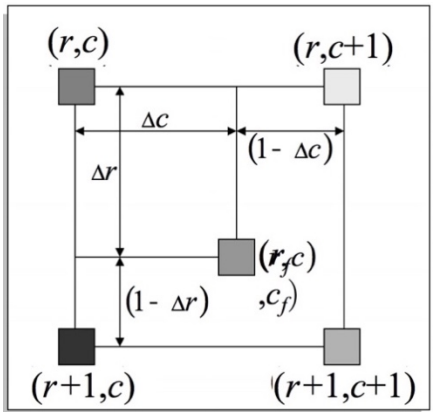


Figure 2: Bilinear Interpolation [1]

Mathematically, if we want to interpolate two points r_f and c_f to the new image **J**, we take four points in the original image **I** and find our required Δ*r* and Δ*c,* which are the differences between our interpolated point and first point in the original image. Finding those points, we solve for new image **J** using Equation 1. To be noted, *r* and *c* relates to row and column index respectively.

$$\boldsymbol{J(r_f, c_f) = I(r,c) * (1 - \Delta r) * (1 - \Delta c) + I(r+1, c) * \Delta r * (1 - \Delta c)} \\ \boldsymbol{+ I(r, c+1) * \Delta c * (1 - \Delta r) + I(r+1, c+1) * (\Delta r) * (\Delta c)}$$

To be noted, *r* and *c* relates to row and column index respectively.

Result & Discussion

For our problem, an image of size 512x512 was asked to be transformed into an image of size 650x650, which relates to converting an image with same aspect ratio (650:650= 1:1). The obtained result is displayed in Figure 3.

Figure 3: Image Resizing : Left: Image Size (512x512) Right: Image Size (650x650)

To be mentioned, while pasting the image on the Word Document, there was a resizing employed which I believed used a similar technique to the one in the discussed in the problem. However, the size extension of the image can be verified from the code provided separately.

It can be said from the extended image that interpolation is purely resolution-dependent since we are dealing bitmaps relating to pixels. Hence, it is not possible to get a perfect pixel extension through this method. An example of such case can be observed by zooming into the moon of the starry night image [Figure 4].

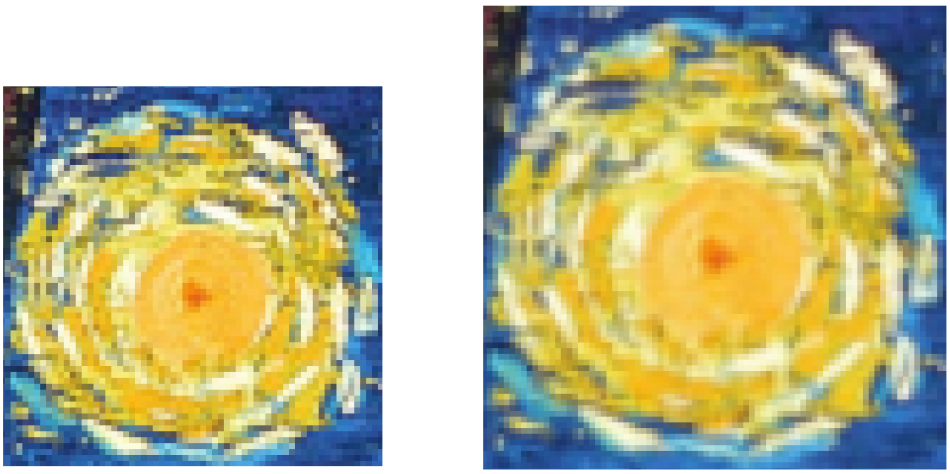


Figure 4: Zooming into both images (Left: 512x512, Right: 650x650)

However, the overall look of the resized image is not affected much by this process. Hence, bilinear interpolation has proved to be good choice for visualization.

(b) Demosaicing of Bayer-patterned Color Image

Motivation

Color Filter Array (CFA) is used in digital camera sensors to get a colored image from captured grayscale values. The importance of such idea relates to the cost factor since we would need to have sensors for three channels. Apart from that, it also helps in saving computational energy. There are various algorithms that have been developed to convert a grayscale value to a colored value. In this section, we will deal with two of them: i) Bilinear Demosaicing (the basic approach), ii) Malvar-He-Cutler (MHC) Linear Image Demosaicing (an improved approach)

Method 1: Bilinear Demosaicing

To understand the method, we need to know how the Color Filter Array looks. An example of Color Filter Array, also called Bayer Pattern is attached in Figure 5.

R 11	G 12	R 13	G 14	R 15	G 16	R 17
G 21	B 22	G 23	B 24	G 25	B 26	G 27
R 31	G 32	R 33	G 34	R 35	G 36	R 37
G 41	B 42	G 43	B 44	G 45	B 46	G 47
R 51	G 52	R 53	G 54	R 55	G 56	R 57
G 61	B 62	G 63	B 64	G 65	B 66	G 67
R 71	G 72	R 73	G 74	R 75	G 76	R 77

Figure 5: Color Filter Array

As observed the color filter array has a pattern of continuous **R-G** pattern in the odd row and **G-B** in the even row. It is evident from the array that it has 50% values assigned to green channel, the rest 50% equally divided among red and blue channel. For bilinear demosaicing, we find the missing pixels for each channel by calculating by averaging the neighboring pixel values. For example: if we want to find the value of R at (2,2) position [assigned to Blue in CFA), we can take the average of

neighboring red pixel values i.e. (1,1), (1,3), (3,1), (3,3). In this way we can produce a total number of eight filters for all the channels and solve for them.

In terms of programming, we separate three channels and find the missing pixels in each channel by using **if-condition** on the idea of even/odd rows and columns. For examples: If I want to find the missing pixels for red channel at a blue position, I observe that all the blue values are placed even-indexed row and even-indexed column. Hence, I call for even-indexed row and column and average it using the neighboring pixels. A snippet of the code for red channel is provided in Figure 6.

```
% Missing Red values at Green and Blue position
for r= 3:row+2
    for c= 3:col+2
        if I_R(r,c) == 0
                if mod(r,2) == 0 && mod(c,2) == 0
                    I_R(r,c) = 1/4*(I_R(r-1,c-1)+I_R(r-1,c+1)+I_R(r+1,c-1)...
                    +I_R(r+1,c+1));
                elseif mod(r,2) ==0 && mod(c,2) ~= 2
                        I_R(r,c) = 1/2*(I_R(r-1,c)+I_R(r+1,c));
                    elseif mod(r,2) ~= 0 && mod(c,2) == 0
                            I_R(r,c) = 1/2*(I_R(r,c-1)+I_R(r,c+1));
                end
        end
    end
end
```

Figure 6: A code snippet of Bilinear Demosaicing

To be added, the original image was extended in top and bottom by 2 rows and in left and right by 2 columns symmetrically maintaining CFA pattern. This was done to avoid any computational complexity in the corners and boundaries.

The final result of the procedure is provided in Figure 7 along with the given grayscale image. It can be observed that the overall look of the image visually appears to be a Parrot colored appropriately. However, if we zoom-in to the image and attempt to look at pixel details, we will find out some artifacts which are produced as a result of this method.

Figure 7: Bilinear Demosaicing. Left: Provided Grayscale Image; Right: Colored Parrot Image

A common artifact that can be observed is the artificially jagged pattern called the zipper effect [Figure 8-Left]. The reasons of such occurrence is due to the interpolation of pixel values, which causes the values in the region of lighter and heavier values to get different color values in neighboring spots [2]. Along with that, we can observe erroneously estimated colors, also known as false colors [Figure 8-Right]. Although these errors are minor in this image, we can achieve better results through our next procedure. Figure 8 shows some of these artifacts.

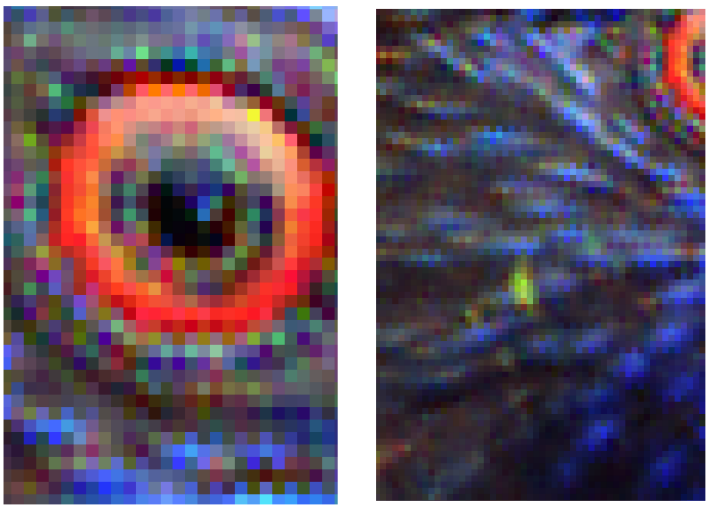


Figure 8: Artifacts observed: Left: Artificially jagged pattern ("Zipper effect") Right: False colors (Major error is the green color)

Method 2: Malver-He-Cutler (MHC) Linear Image Demosaicing

In this method, we improve Bilinear Demosaicing by adding a correction value to it. The correction value is multiplied by a scaling factor. So, the basic formula behind the procedure is:

$$Value\ of\ Missing\ pixel = Value_{bl} + (scaling\ factor) * Correction\ term$$

$Value_{bl}$ corresponds to value obtained using bi-linear transformation. The correction value can be obtained from the filters that are provided in the proposed algorithm paper [3]. There are total number of eight filter based on CFA array and three shapes of filters with values provided in the paper [Figure 9].

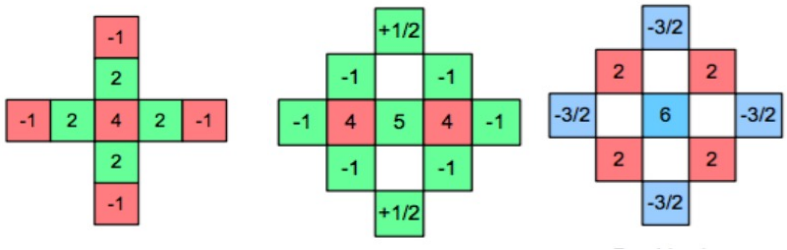


Figure 9: Three filter shapes for MHC Algorithm[3]

Based on this filter shape, we choose the scaling factors. There are three scaling factors specified in the paper, namely α,β,γ and the values that are found to be applicable are ½ , $\frac{5}{8}, \frac{3}{4}$ [3]. For the first filter shape, we multiply by α. For the second filter shape, we multiply by β and the last one by γ.

For the correction term, we select based on the filter type we have. For example, if we are trying to solve for missing red at green locations, we take the middle red pixel to be 1 and divide the other red pixels by 4. So our correction term becomes:

$$C(R\ at\ B) = R(r,c) - \frac{1}{4} * (R(r-2,c) + R(r+2,c) + R(r,c-2) + R(r,c+2))$$

Here r and c are the row and column indices.

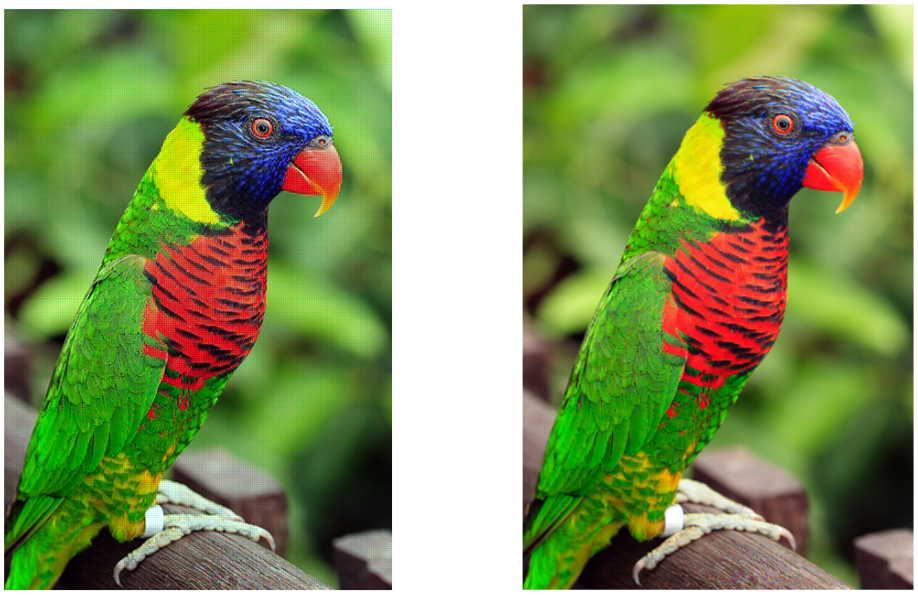


Figure 10: Left: MHC Algorithm Demosaicing Right: Bilinear Demosaicing

The result obtained from MHC Algorithm is shown in Figure 10-Left along with Bilinear Demosaicing in the right. We can observe that the changes in both images are subtle. However, it seems MHC by adding a gradient correction value has obtained more details on the image compared to the Bilinear Demosaicing (Figure 10-Right). It is because in case of Bilinear Demosaicing, we only considered the values at neighboring pixels. However, in case of MHC Algorithm we also considered for values outside the range of neighboring pixels for missing pixel and added a correction value. The correction value is supposed to avail from the fact of details while creating the image.

Problem 2

Image Enhancement is one of the mostly used ideas from image processing. It is the key reason through which we are able to create different looking image effects through the concepts discovered in image processing. In this problem, we are going to look at two methods by which we can manipulate the histogram to create more visually perceptive images. We will also develop an oil-filtering image effect, which will involve the idea of quantization in a basic manner.

(a) Histogram Equalization

Motivation

Histogram Equalization is used to enhance an image by manipulating the grayscale values in an image. We can access the information about the distribution of grayscale values in an image through histogram. The importance of such problem can be found in many places now-a-days such as, military applications, space stations (NASA) and so on. We can do histogram equalization through various methods. In this problem, we will tackle two methods: i) transfer-function based histogram equalization, ii) cumulative probability based histogram equalization. It is to be mentioned that the equalization was applied on each channel separately for both methods and then the final image was created. Both of this method uses cumulative histogram to map values into the output equalized channel.

Methods

The steps followed for the first method are: i) We find the histogram for each channel separately; ii) We create a cumulative histogram, which basically adds up the total number of pixels in full range of intensities; iii) We create a normalized Cumulative Distribution Function (CDF) such that the values range from 0 to 1; iv) We create a temporary output which multiplies each values of CDF with the max intensity (in this case 255); v) Finally, we map the values of input channel using the temporary output in a final equalized matrix (channel). The process is repeated for all the channels and the final equalized output is thus obtained. The transfer function in this case is:

$$T(x) = \frac{Max\ Intensity}{Total\ \#\ of\ pixels} * \sum_{i=0}^{x} H(i)$$

Here, *x* represents the gray level value, max intensity is 255, *H(i)* is the histogram value at *i* and total number of pixels is the product of the size of the image.

The steps followed for the second method are: i) We find the histogram, cumulative histogram as done for method one; ii) We find the total number of pixels each gray value should consist of i.e. N = Total number of pixels / 256; iii) We create an array with 256 bins such that the maximum number of pixels that can be hold in each bin is equal to N; iv) We find the bin where each pixel should fall into from the cumulative histogram. In case the bin is full, we move to the next greater value; v)

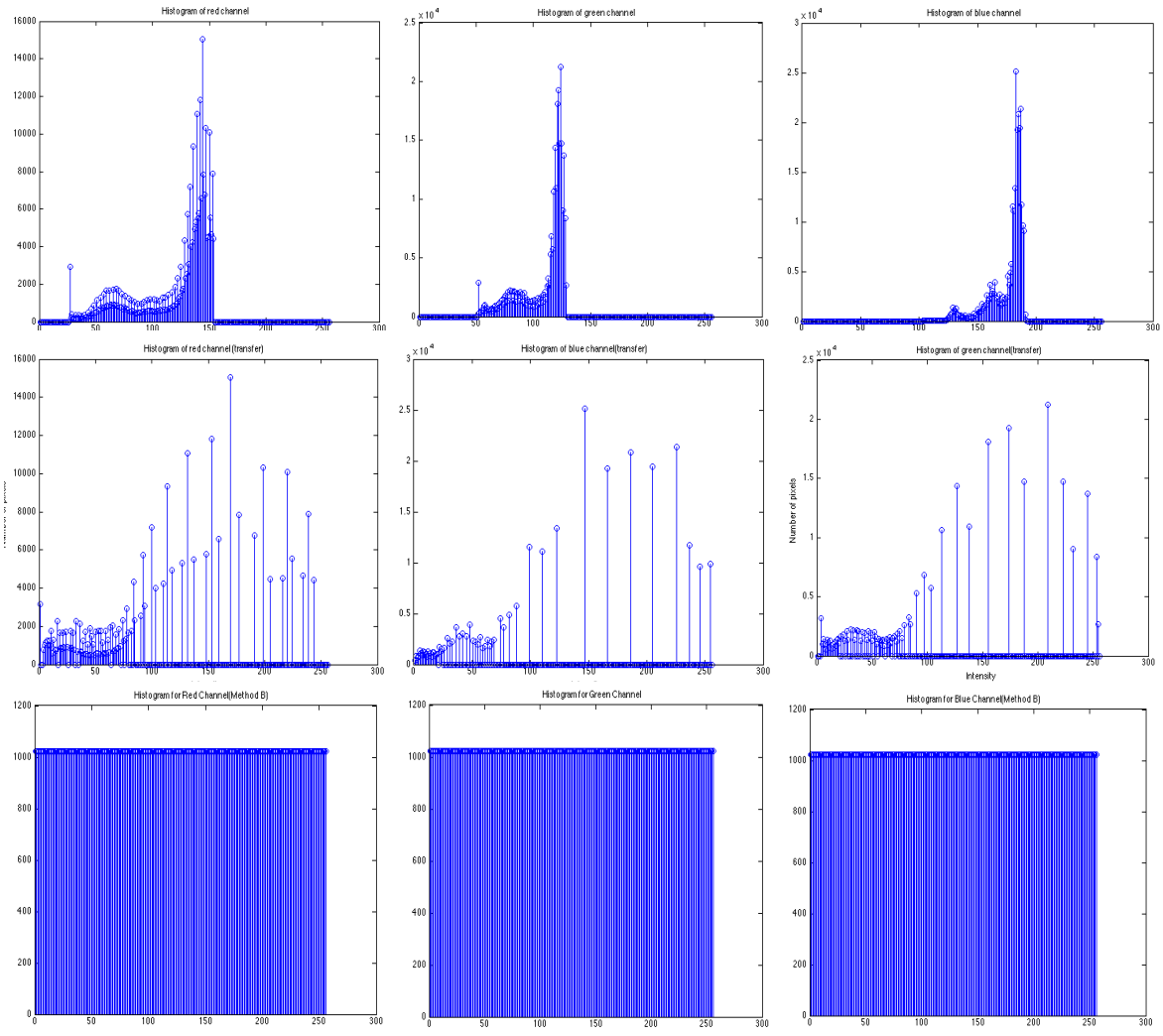
Finally, we place all the pixels in its destined bins. Hence, we should get a histogram which has equal number of pixels for all gray values and the cumulative function should look like a ramp function.

Results & Discussion

The results obtained from both methods looks almost similar [Figure 11]. It is observed that the pink hue spread on the image has decreased and the contrast among pixel values have increased, which is known as contrast stretching. We can get more idea by looking at the histogram of the images for each channel after the methods are applied [Figure 12]. For the first method, we can observe that the histogram can gained a more spread distribution. In case of second method, the histogram has been equalized such that each gray value has equal number of pixels.

Figure 11: Left: Original Image; Middle: Equalized image using Method A; Right: Equalized image using Method B.

A thing to comment on the overall look of the image obtained from Method A and Method B is that image obtained from Method B looks brighter than the image obtained from Method A. The reason behind such case is the original image has lesser value along the white intensities. Method B make it more uniform by pushing the values towards the brighter side. Since they both do mapping to values, the transfer function obtained from Method A and cumulative distribution obtained from Method B are the same [Figure 13] except for normalization.



(a) **(b)** **(c)**

Figure 12: (a) Histogram for red channel; (b) Histogram for green channel; (c) Histogram for blue channel: Top to bottom: Unequalized image, Transfer function based equalization, Cumulative probability equalization.

In case of Method A, we have done one-to-one mapping which leads to spreading the values. On the other hand, for Method B, we did one-to-many mapping and assigned equal number of pixels to all gray values. Hence we find an equalized histogram for each channel.

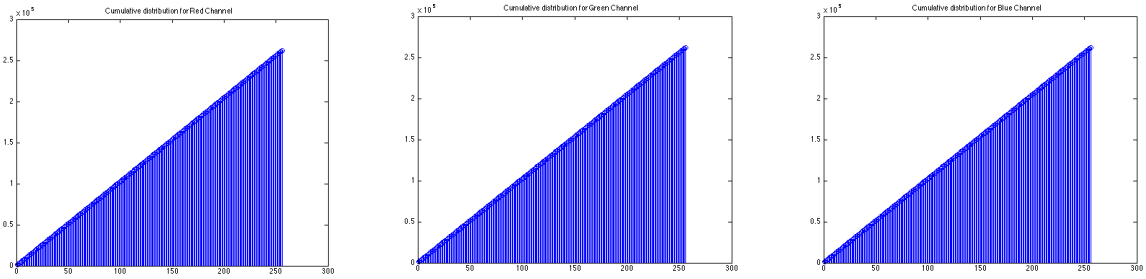


Figure 13: Transfer Function for Method A & Cumulative Distribution for Method B

An immediate drawback of Method B is that the uniform distribution may not be a good choice if the values in an image are skewed towards one gray value at lot. An image with a lot of values on max intensity and min intensity will be distributed such that it creates an image whose effect is not desirable. There are various methods which can be used to do equalize histogram. One of them is creating a power distribution or a Gaussian distribution. We can also build an adaptive control on the histogram to control discretely part of the image. However, it seems that both the methods have been able to provide a decent result in showing the texture of jet plane.

(b) Oil Painting Effect

Motivation

Filters have become popular these days which involve usage of special effects. Oil painting effect is one of them that can be employed as an image filter for special effects. Doing it on Adobe Photoshop also needs a lot of work to go through. However, we can implement a painting effect which can assist us in getting an idea of how to get started on applying the filter effect.

Method

Creating the filter effect takes two steps to be completed:

Step 1: We quantize the colors such that each channel has lesser value of colors than 256 values with which they are provided. To accomplish this step, we first divide the total number of pixels (N) in each channel by the number of gray values (n) we want (=N/n). Then, we find threshold values from the cumulative histogram of the channel to which N/n values occur. Thus we should get n bins created. After that, we take the average of n bins starting from 0 to 255 and assign the mean values in the bin ranges. We repeat the process for all the channels and concatenate them. This

is our quantized image. This procedure can be done through K-mean clustering as well.

Step 2: After we quantize the image for each channel, we do boundary extension so that we can apply filter of different sizes to our image. Then, we apply the filter kernel to each rows and columns. Next on each filter kernel, we find the most frequent value. Thus the image will get values filled most neighboring pixels. Finally, we combine all three channels and the output produced will give almost an oil painting look. However, there is a dependence on the size of filter which we will find in the discussion section.

Result & Discussion

The results obtained from the quantization effect are provided in Figure 14 for both 64 color (4 gray values in each channel) and 512 colors (8 gray values in each channel).

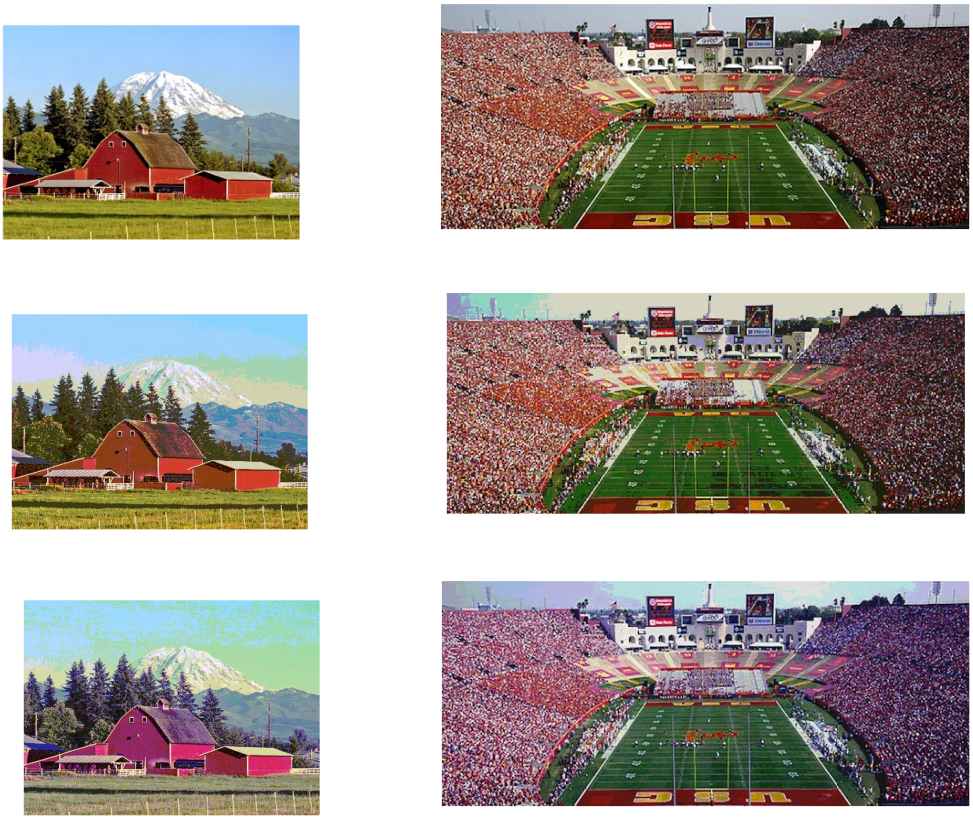
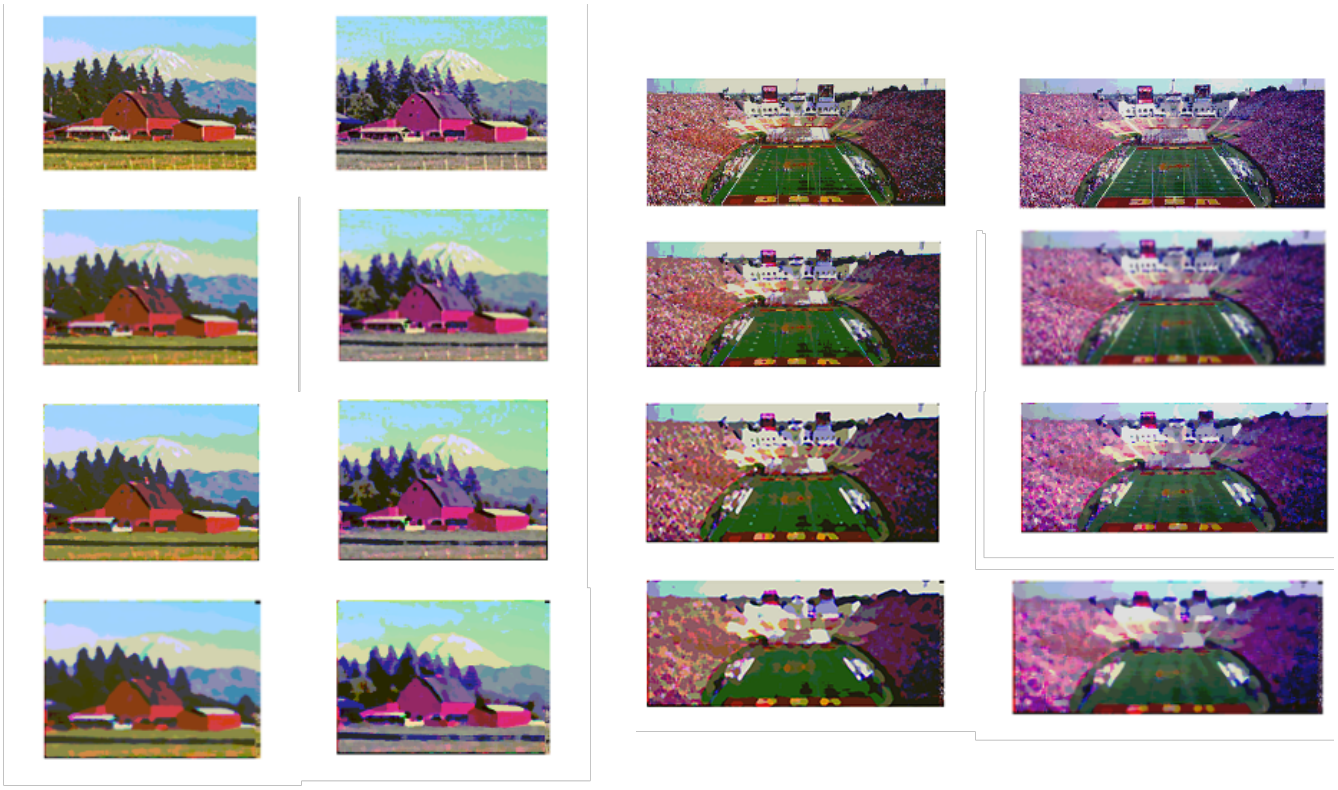


Figure 14: Top to Bottom: Original Image – 64 color quantized image – 512 color quantized image. Left to Right: Barn.raw- Coliseum.raw

We can observe from the quantized image that the color content has already decreased and it is starting to give an oil effect feel for 64 color image. However, for 512 color quantized image, the contents are getting deeper since we have colors. As a whole, the image of 64 color has more resemblance to the original.

After we have quantized the image, we apply the filter kernel described for oil painting effect and try to check for which filter order provide us with better results. The filter size that were applied on the image are 3x3, 5x5, 7x7 and 11x11. The filter effect was applied for both 64 colored quantized image and 512 colored quantized image. Figure 15 includes all the images that were produced in the process.



(a) **(b)**

Figure 15: (a) Oil filter on Barn image: Top to bottom: 3x3,5x5,7x7,11x11 filter; Left to Right: 64 color image,512 color image (b) Oil filter on Coliseum image: Top to bottom: 3x3,5x5,7x7,11x11 filter; Left to Right: 64 color image,512 color image

As observed from the images, the increasing of filter size makes the image looks more like an oil painting effect. In case of Barn images, a pretty good oil painting effect is achieved at 11x11 filter for both 64 color quantized image and 512 color

one. We can also take 7x7 as a good consideration. In case of Coliseum image, we get a pretty good result at 11x11 for both 64 color quantized image and 512 color one. An explanation to that could be as we increase the filter size, the number of neighborhood colors gets to the same value, which is desirable to get a oil painting effect. It is to be added that the oil painting effect produced in these images can not be supposed to mimic the oil painting effect produced by an artist.

There is a relationship between the filter size and computation time. As we increase the filter size, the computation time proportionally increases. The time count for an 11x11 filter on 64 color quantized image was found to be within the range of 40-45 seconds when applied on a Macbook Pro Retina with 2.5 GHz Intel Core i7 & 16 GB RAM. The function **tic-toc** from MATLAB was used to calculate the computation time. The computation time for similar filter size on 512 color quantized image was found to be around 44-49 seconds. Although this technique is good way to learn about oil painting effect, it is not practically useful as a filter.

Problem 3

Noise removal is an important element in digital photography. It is not possible for sensor to pick the right intensity all the time. As a result, noise gets introduced to the image. Filters have been discovered to remove the noise by identifying their types. In this problem, we will go through some basic filtering techniques along with a non-linear filter called guided image filter.

(a) Mixed noise in color image

Motivation

Color image deals with noise in three separate channels. Hence to solve the noise issue, we need to apply filters which can remove the noise. Before removing the noise, we need to check in our histogram what sort of noise are in the colored image.

Method

To find the type of noise available in the channels, we need to look at the histogram of three channels [Figure 16]:

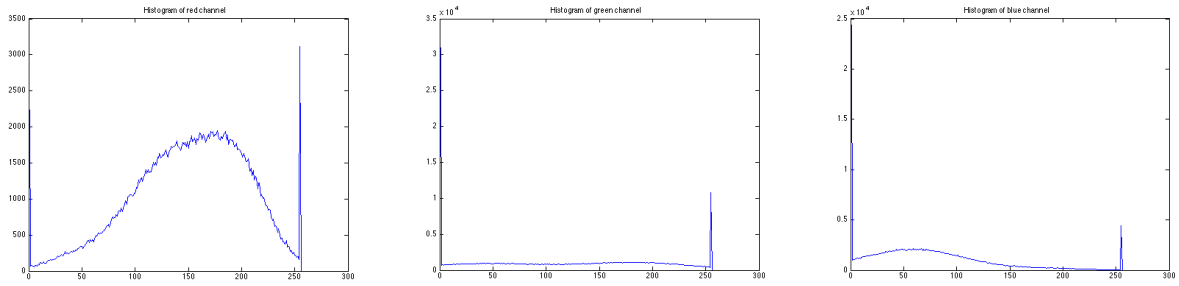


Figure 16: Left to Right: Red channel of noisy pepper image, Green channel of noisy pepper image, Blue channel of noisy pepper image,

Here we plotted the noise in continuous domain to make it easier for us to understand the pattern. We can observe that there are two impulses for all the channels at pure black and white intensity. So, we have "**salt and pepper**" noise in the image. We also have some additive random noise in the channels. Red channel has more additive noise following a Gaussian-like pattern.

First to apply our filter, we need to extend our boundaries of the image depending on the window size of the filter. The procedure of repeating is accomplished in the previous oil painting problem as well. Now we will apply linear image filters on the image and observe the results. In this case, we will take median filter and averaging filter (also known as low pass filter). The order of applying two of these filters will make a difference since filtering an image is not a linear process. At the same time, the window size of filters will also have an effect. We start with window size of 3x3 of each filter and apply them in two orders (first median + next low-pass and first low-pass + next media). The results obtained are shown in Figure 17.

Figure 17: Left to Right: Original Noisy Image; Median Filter-Next Low pass filter; Low pass Filter-Next Median filter

Result & Discussion

It is visually observed that applying median filter first provides better result than the other. Intuitively, it is more logical to have median filter at first than the averaging

filter. Applying average filter first will average the impulse noises ("salt and pepper") at both sides. Peak Signal-to-Noise Ration **(PSNR)** of higher value for the first suggest that it is good choice. PSNR is a quantity usually used to check for noise-to-signal ratio in a logarithmic scale. However, PSNR can not always be used since it does not count the human's vision capability in its in its intuition. So, it is necessary to use visual sense to apprehend the quality of image.

Since we are set with the first filter cascading order, we change the size of window to investigate what happens. Intuitively, the averaging of the values will make the image blurred as observed in Figure 18.

Figure 18: Left to Right: Applying filter for window of size N=5,7,9,11

To improve the given performance, we can introduce spatial function through Gaussian filters which will decrease the blurring effect with an increased window size. We can also use different shapes in filter windows which can make difference to a certain extent. Apart from that we can enter into transformation domain and preserve edges by applying Bilateral filtering and Non-Local Means (NLM) filtering. The next filter uses an innovative procedure of using a guided image to denoise.

(b) Guided Image Filtering [4]

Method

Guided filter is an example of a non-linear filter that we can use to remove edge degradation. The filter uses a guided image to compare with the input image and creates an output image ultimately. Let's suppose that ***p*** be our input image, ***I*** be our guided image and ***q*** is the output image. Our final solution for the output image is:

$$q = a_{mean} * I + b_{mean}$$

where,

a_{mean} and b_{mean} are the weighted average in the window kernel with the following calculations:

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}$$
$$b_k = \bar{p}_k - a_k \mu_k.$$

where μ_k and σ_k^2 are the mean and variance of guided Image *I* in the window of size k. ε is a regularization parameter preventing a_k from being too large [4]

The implementation of the filter was done first by extending the boundaries based on the window size. Then, a function was created which calculates the weighted average and pass through every necessary parameter needed to be calculated as shown above. In this case the guided image was chosen to be the same as the original noisy pepper image. So, *I* = *p*.

Result & Discussion

The result obtained after applying the filter are shown in Figure 19:

Figure 19: Left to Right: Original noisy image, Filtered image.

The above result is shown for r=3 and ε= .01. The guided image filter was also applied with different r and ε values. The results obtained are shown in Figure 20.

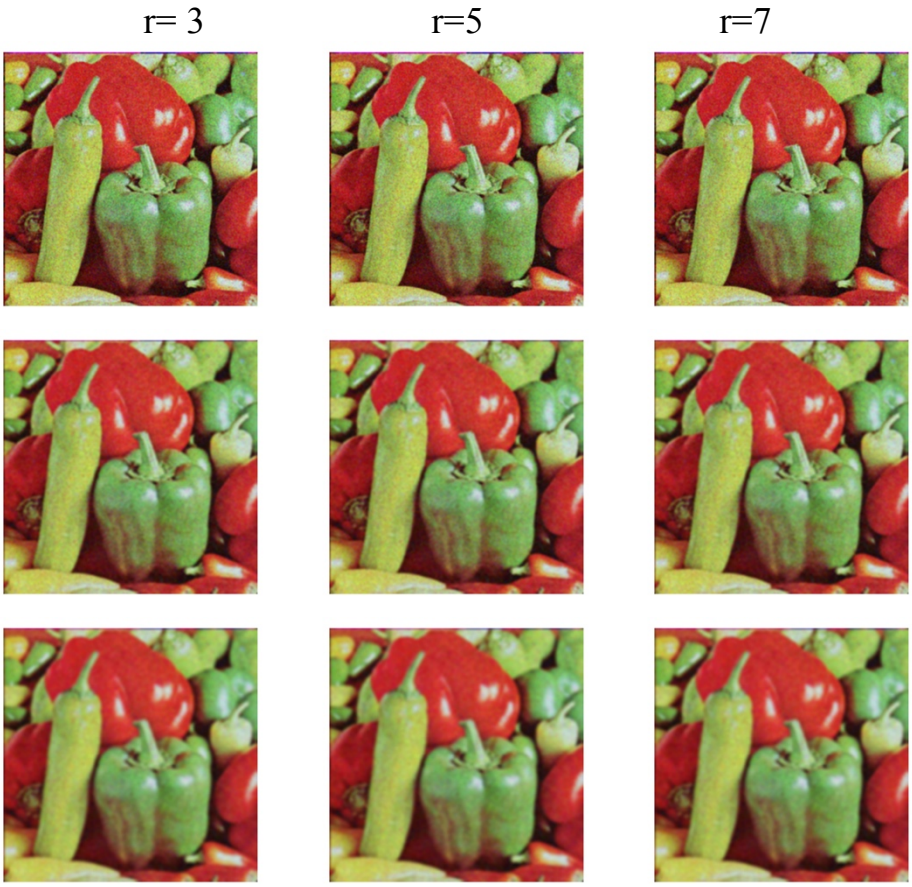


Figure 20: Top to bottom: ε = (.1)^2, ε = .2^2, ε = .8^2

As we observe with an increased ε, the picture tends to get blurry and the regularization parameter's control on the linearity of image degrades. We investigate whether the given filter has better performance than linear filter (say Low pass filter). We can observe it from the results provided in Figure 21 which shows that guided image filter has better performance than low pass (averaging) filter.

Figure 21: Left to Right: Low pass filter – Guide Image filter

Guided image filtering is better than linear filter is that it is not spatially invariant like linear filters and has the independence of adding more details through its concept than the linear filter. Applying the guided image filter multiple times creates a blurrier effect. We can observe in Figure 22 that as we apply the filter 5 times on the same image, we get a blurrier effect. In this case, the original pepper image is considered.

Figure 22: Applying guided image filter 5 times on the original pepper image

(c) Block Matching and 3-D (BM3D) Transform Filter

The principle reason behind developing filters to denoise is image is that it is difficult to achieve the best configuration for all types of noises. Hence, we develop more filters which can assist us in solving and achieving different desired results. BM3D uses an expressive method using the idea of 3D transformation. In image processing, we deal with matrices known as blocks. BM3D converts them to three dimensional groups and apply a filtration process to attenuate the noise [5].

I believe from my understanding of the paper that BM3D is on most of the part a denoising tool involved in frequency domain since it uses 3-D transformation and thus work in transformation domain. However, it performs an inverse 3-D transformation to put it back into blocks, thus involving some portion of the work in spatial domain as well.

Note: Some of the images in this report may not be good enough. The image files can be made available or can be created from the MATLAB solutions in the source code. If any help is required in this regard, please contact me via e-mail. I will be grateful to help in the process.

References

[1] R. Peters ,2011. [Online]. Available: https://ia802707.us.archive.org/23/items/Lectures_on_Image_Processing/EECE253_12_Resampling.pdf

[2] Losson, Macarie, Yang. "Comparison of color Demosaicing methods." Advanced in Imaging and Electron Physics, Elsevier, 2010, 162, pp. 173-265.

[3] Malvar, Henrique S., Li-wei He, and Ross Cutler. "High-quality linear interpolation for demosaicing of Bayer- patterned color images." Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP'04). IEEE International Conference on. Vol. 3. IEEE, 2004

[4] He, Kaiming, Jian Sun, and Xiaoou Tang. "Guided image filtering," IEEE Transactions on Pattern Analysis and Machine Intelligence, 35.6 (2013): 1397-1409.

[5] Dabov, Foi, Katkovnik and Egiazarian. "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering," IEEE Transactions on image processing, vol. 16, no. 8, August, 2007.