EE 569: Homework #1

Issued: 8/28/2015 Due: 11:59PM, 9/20/2015

General Instructions:

1. Read Homework Guidelines and MATLAB Function Guidelines for the information about homework programming, write-up and submission. If you make any assumptions about a problem, please clearly state them in your report.
2. Do not copy sentences directly from any listed reference or online source. Written reports and source codes are subject to verification for any plagiarism. You need to understand the USC policy on academic integrity and penalties for cheating and plagiarism. These rules will be strictly enforced.

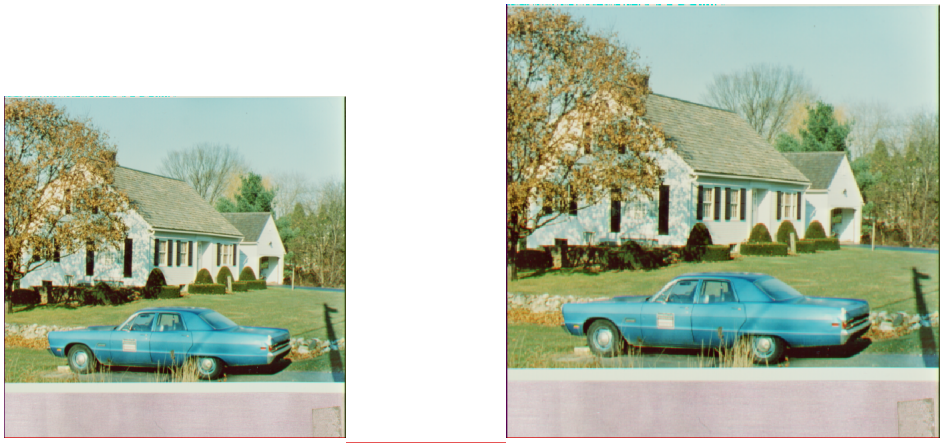
Problem 1: Image Manipulation and Interpolation (30%)

In this problem, you are asked to do a series of simple manipulations on the pixels of grayscale and color images so that you will be familiar with image data access, processing and output.

(a) Image Resizing via Bilinear Interpolation (Basic: 10%)

Use the bilinear interpolation to re-size the input color image *the-starry-night* of size 512x512 in Figure 1 to an output image of size 650x650. An image re-sizing example for the *House* image is given in Figure 2.

Figure 1: The-starry-night



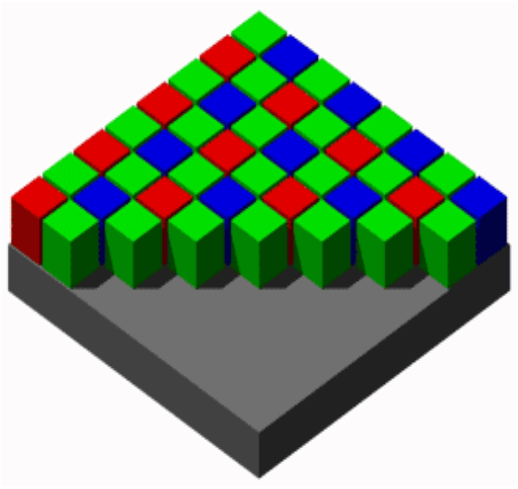
(a) house image of size 512x512 (b) resized house image of size 650x650

Figure 2: **Original and resized House images.**

(b) Demosaicing of Bayer-patterned Color Image

To capture color images, digital camera sensors are usually arranged in form of a color filter array (CFA), called the Bayer array, as shown in Figure 3. Since each sensor at a pixel location only captures one of the three primary colors (R, G, B), the other two colors have to be re-constructed based on their neighbor pixel values to obtain the full color. Demosaicing is the process of translating this Bayer array of primary colors into a color image that contains the R, G, B values at each pixel.

You are required to implement two demosaicing algorithms in this part. Exemplary demosaicing results are given in Figure 4.



(a)

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

(b)

Figure 3: **Bayer Pattern**

1) Bilinear Demosaicing (Basic: 10%)

Bilinear demosaicing is the simplest demosaicing method. The *Fruit_Shop* image in Figure 4(b) shows the results of bilinear demosaicing. Generally speaking, the reconstructed color value is computed as the average of its two or four adjacent pixels of the same color. To give an example, the green value at a blue or red sensor location is computed as the average of four adjacent green pixels:

$$\hat{G}^{bl}(i,j)=\frac{1}{4}(G(i-1,j)+G(i+1,j)+G(i,j-1)+G(i,j+1)). \qquad (1)$$

Implement bilinear demosaicing and apply it to the *Parrot* image in Figure 5. Show your results and discuss the strength and shortcomings of this algorithm. (Hint: Do you see any artifacts? If yes, explain the cause of the artifacts using sampling theory).


(a)fruit_shop CFA image


(b) bilinear demosaicing result


(c) MHC demosaicing result

Figure 4: **Demosacing results of Fruit_Shop image**

Figure 5: Parrot CFA image

2) Malvar-He-Cutler (MHC) Linear Image Demosaicing (Advanced: 10%)

Malvar et al. proposed an improved linear interpolation demosaicing algorithm [1]. It yields a higher quality demosaicing result by adding a 2nd-order cross-channel correction term to the basic bilinear demosaicing result. The MHC linear demosaicing result of the *Fruit_Shop* image is given in Figure 4(c). The MHC algorithm is stated below.

To estimate a green component at a red pixel location, we have

$$\hat{G}(i,j) = \hat{G}^{bl}(i,j) + \alpha\Delta_R(i,j) \tag{2}$$

where the 1st term at the right-hand-side (RHS) is the bilinear interpolation result given in (1) and the 2nd term is a correction term. For the 2nd term, alpha is a weight factor, and Δ_R is the discrete 5-point Laplacian of the red channel:

$$\Delta_R(i,j) = R(i,j) - \frac{1}{4}\big(R(i-2,j) + R(i+2,j) + R(i,j-2) + R(i,j+2)\big) \tag{3}$$

To estimate a red component at a green pixel location, we have

$$\hat{R}(i,j) = \hat{R}^{bl}(i,j) + \beta\Delta_G(i,j) \tag{4}$$

where Δ_G is a discrete 9-point Laplacian of the green channel.
To estimate a red component at a blue pixel location,

$$\hat{R}(i,j) = \hat{R}^{bl}(i,j) + \gamma\Delta_B(i,j) \tag{5}$$

where Δ_B is a discrete 9-point Laplacian of the blue channel. The weights α, β, γ control how much correction is applied, and their default values are:

$$\alpha = \frac{1}{2}, \beta = \frac{5}{8}, \gamma = \frac{3}{4} \tag{6}$$

The above formulas can be generalized to missing color components at each sensor location. Actually, the MHC demosaicing can be implemented by convolution with a set of linear filters. There are eight different filters for interpolating the different color components at different locations, illustrated in Figure 6 [1].

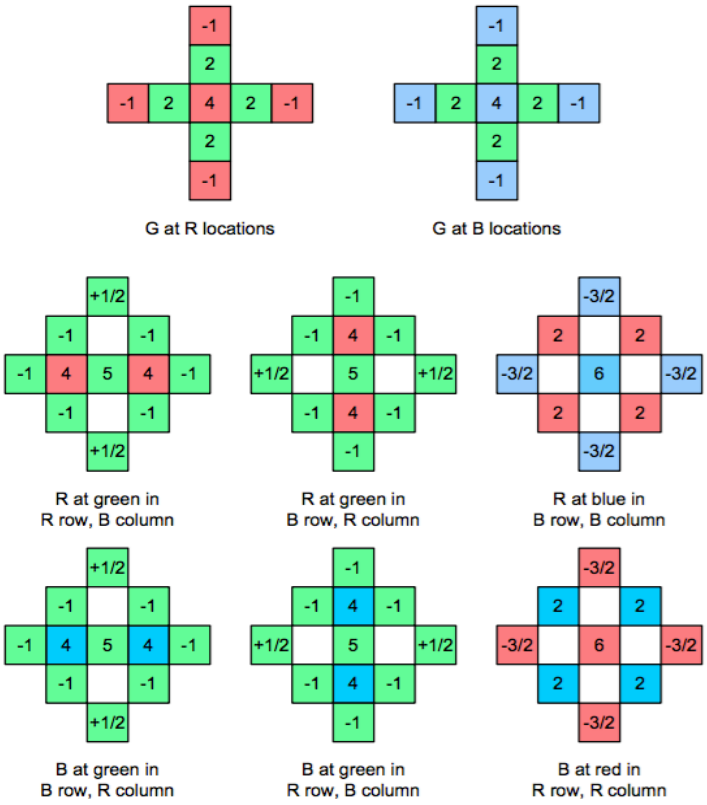


Figure 6: Filter coefficients

Implement the MHC linear demosaicing algorithm and apply it to the *Parrot* image in Figure 5. Show your results. Compare your result with the result using bilinear demosaicing in the previous problem, and explain the performance differences between these two algorithms in your own words.

Problem 2: Histogram Equalization and Image Filtering (40 %)

(a) Histogram Equalization (Basic: 20%)

Implement the following 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 *Jet* image in Figure 7 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?

Note that MATLAB users CANNOT use functions from the Image Processing Toolbox except displaying function like imshow().

Figure 7: **Jet**

(b) Image Filtering – Creating Oil Painting Effect (Advanced: 20%)

An exemplary oil-painting effect for the *Yosemite* image is shown in Figure 8. This effect can be implemented as a filter with the following two steps.

(a) Yosemite

(b) Yosemite with a reduced color set

Figure 8: Yosemite with a full and a reduced color sets

Step 1: Quantize all colors of the input color image, denoted by I_0, into an image containing only 64 colors, denoted by I_1. In other words, each channel should have only 4 values. The original and truncated *Yosemite* images are shown in Figs. 5 (a) and (b), respectively.
Step 2: For each pixel of image I_1, 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_2. The result is shown in Figure 9.

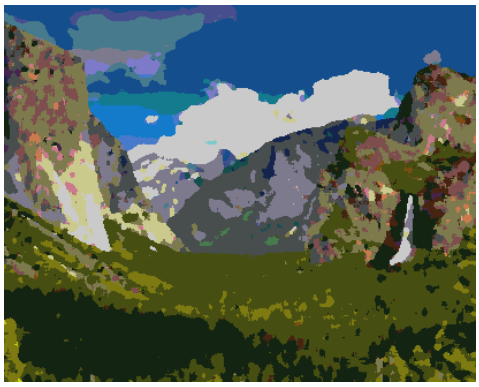


Figure 9: Yosemite with the Oil-Painting Effect

Implement and apply the oil-painting filter to the *Barn* and *Coliseum* images as shown in Figs. 7 and 8.
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 10: Barn

Figure 11: **Coliseum**

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

Noise is a common degradation factor for digital photography. In this problem, you will implement a set of denoising algorithms to improve image quality. You can use the PSNR (peak-signal-to-noise-ratio) quality metric to assess the performance of your denoising 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 $N \times M$ (7)

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

Max: Maximum possible pixel intensity = 255

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

In this part, you are required to perform noise removal on a color image corrupted by a mixed type of noise. The original and noisy Peppers images are shown in Figure 12.

Figure 12: The original and noisy peppers images.

1) Identify noise types in the noisy peppers image, and answer the following questions.
   a) Do all channels have the same noise type?
   b) Should you perform filtering on individual channels separately for both noise types?
   c) What filters would you like use to remove mixed noise?
   d) Can you cascade these filters in any order? Justify your answer.
   e) Discuss the effect of different filter window sizes.
2) Get the best results in removing mixed noise. Include the following in your report:
   a) Describe your method and show its results.
   b) Discuss its shortcomings.
   c) Give some suggestions to improve its performance.

(b) Guided Image Filtering (Advanced: 15%)

We often see edge degradation by applying a low-pass linear denoising filter. Some non-linear filters have been designed to preserve the edge sharpness. One example is the guided filter [2], which exploits a second image called the guidance image. The guidance image can be the image itself, a different version of the image, or a completely different image.

1) Explain the idea of the guided filter and implement it based on [2]. Apply it separately to each channel of the noisy *Peppers* image in Figure 12 with the noisy image itself as the guidance image. State and explain the complexity of your implementation.
2) Find out the best configurations of parameters r and ε, and discuss the change in filter's performance with respect to the values of r and ε.
3) Does this filter perform better than linear filters? Justify your answer in words.
4) The guided filter has an interesting property. That is, applying it to an image multiple times will result in a cartoon matting effect. Please verify this claim by applying it to the original *Peppers* image.

(c) Block Matching and 3-D (BM3D) Transform Filter (Bonus: 10%)

In this part, you will get familiar with a state-of-the-art denoising algorithm proposed in [3].

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 *Peppers* image (Figure 12). 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 [4]. Their code is written in MATLAB; so it is okay to use MATLAB for this part, even if you have been coding on C/C++ platform (You would still qualify for 5% bonus points if you have used C/C++ everywhere else).
2) Both the Gaussian and guided filters are spatial domain filters. How would you classify BM3D - spatial domain, frequency domain, or both? Justify your answer.
3) Conduct qualitative performance comparison between the algorithms developed for Problem 3(a) and Problem 3(b) and BM3D.

Note: DO NOT quote statements directly from any listed references or online sources. You need to write your report in your own words. Written reports and source codes are subject to verification for any plagiarism.

Appendix:

Problem 1: Simple Image Manipulation			
the-starry-night.raw	512x512	24-bit	Color (RGB)
house.raw	512x512	24-bit	Color (RGB)
house_scaled.raw	650x650	24-bit	Color (RGB)
fruit_shop_CFA.raw	574x800	8-bit	Grayscale
fruit_shop-bl.raw	574x800	24-bit	Color (RGB)
fruit_shop-MHC.raw	574x800	24-bit	Color (RGB)
parrot_CFA.raw	424x636	8-bit	Grayscale
Problem 2: Histogram Equalization and Image Filtering			
jet.raw	512x512	24-bit	Color (RGB)
yosemite.raw	410x327	24-bit	Color (RGB)
yosemite64.raw	410x327	24-bit	Color (RGB)
yosemite_oil.raw	410x327	24-bit	Color (RGB)
barn.raw	380x275	24-bit	Color (RGB)
coliseum.raw	580x247	24-bit	Color (RGB)
Problem 3: Noise Removal			
peppers.raw	512x512	24-bit	Color (RGB)
peppers_noisy.raw	512x512	24-bit	Color (RGB)

Reference Images

All images in this homework are from Google images [5] or the USC-SIPI image database [6].

References

[1] 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.

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

[3] 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.

[4] [Online]. Available: http://www.cs.tut.fi/~foi/GCF-BM3D/

[5] [Online] http://images.google.com/

[6] [Online] http://sipi.usc.edu/database/