EE 569: Homework #1

Issued: 01/08/2024 Due: 11:59 PM, 01/28/2024

General Instructions:

1. Read *Homework Guidelines* and *MATLAB Function Guidelines* for information about homework programming, write-up, and submission.
2. If you make any assumptions about a problem, please clearly state them in your report.
3. 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 Demosaicing and Histogram Manipulation (30%)

(a) Bilinear Demosaicing (10%)

To capture color images, digital camera sensors are usually arranged in the form of a color filter array (CFA), called the Bayer array, as shown in Figure 1. 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, and B values at each pixel.

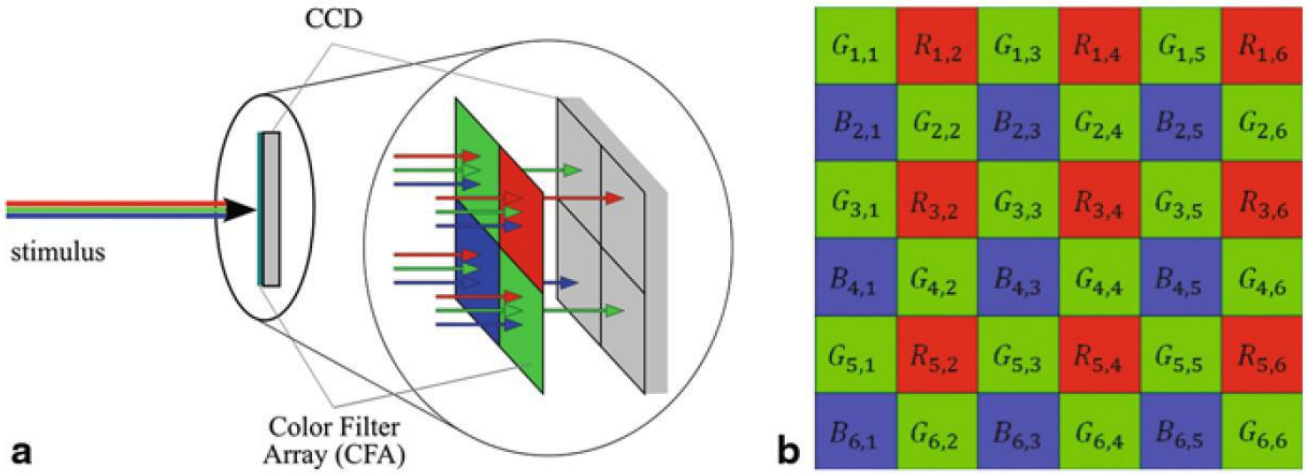


Figure 1: (a) Single CCD sensor covered by a CFA and (b) Bayer pattern [1].

Implement the simplest demosaicing method based on bilinear interpolation. Exemplary demosaicing results are given in Figure 2. With this method, the missing color value at each pixel is approximated by bilinear interpolation using the average of its two or four adjacent pixels of the same color. To give an example, the missing blue and green values at pixel $R_{3,4}$ are estimated as:

$$\hat{B}_{3,4} = \frac{1}{4}(B_{2,3} + B_{2,5} + B_{4,3} + B_{4,5})$$

$$\hat{G}_{3,4} = \frac{1}{4}(G_{3,3} + G_{2,4} + G_{3,5} + G_{4,4})$$

As for pixel $G_{3,3}$, the blue and red values are calculated as:

$$\hat{R}_{3,3} = \frac{1}{2}(R_{3,2} + R_{3,4})$$
$$\hat{B}_{3,3} = \frac{1}{2}(B_{2,3} + B_{4,3})$$

(a)

(b)

Figure 2: (a) The original image and (b) the demosaiced image by bilinear interpolation.

Figure 3: The *House* image before demosaicing.

(1) Apply the bilinear demosaicing to the *House* image in Figure 3 and show your results.
(2) Compare your demosaiced image with the color image *House_ori* which is obtained from a more advanced demosaicing algorithm. Do you observe any artifacts? If yes, explain the cause of the artifacts and provide your ideas to improve the demosaicing performance.

(b) Histogram Manipulation (15%)

Implement two histogram equalization techniques:

- Method A: the transfer-function-based method,
- Method B: the bucket-filling method

to enhance the contrast of the *DimLight* image in Figure 4 below.

(1) Plot the histograms 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.
(3) Apply Method B to the original image and show the enhanced image. Plot the cumulative histograms before and after enhancement.
(4) Discuss your observations on these two enhancement results. Which one do you think is better and why?

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

Figure 4: DimLight image

(c) Contrast Limited Adaptive Histogram Equalization (15%)

Instead of histogram equalization using global statistics from the entire image, adaptive histogram equalization breaks one image into several regions (tiles) and finds the histogram in each separately. Contrast Limited Adaptive Histogram Equalization (CLAHE) is one of the most popular algorithms. Please read the paper [2] carefully to learn about CLAHE.

In this problem, you will apply the CLAHE to do image haze removal, where you remove the fog in the provided image taken in bad weather. Figure 5 shows an example before and after the haze removal. The process can be done through the following 3 steps.

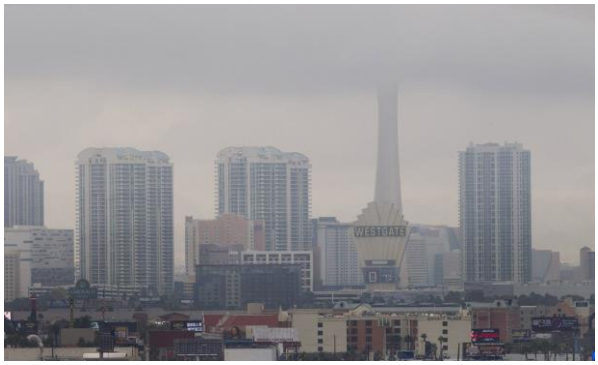
Step 1: Transform the image from *RGB* color space to *YUV* color space. The transformation can be found in the Appendix.
Step 2: Apply a histogram equalization algorithm on the *Y* channel to get *Y'*.
Step 3: Combine *Y'* with *U* and *V*, and transform them back to RGB color space. The resulting image is your haze-removed image.

(1) Explain CLAHE in your own words.

(2) Perform the above three steps on *City.raw* by applying the two histogram equalization methods in Problem 1(b) in Step 2. Show the resulting images for both methods.

(3) Repeat the three steps but apply the CLAHE in Step 2. Here, you are allowed to use open-source code for CLAHE. For C++, you can use ***OpenCV***. For Matlab, you can check function ***adapthisteq***. Tune the hyperparameters including the number of tiles and the clip limit. Show the resulting image that you think is the most pleasant subjectively.

(4) Compare your results between (2) and (3). Discuss your observations.



(a) Original foggy image

(b) Defogged image

Figure 5: An example of image haze removal.

Problem 2: Image Denoising (40%)

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, and 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

Remove noise in the image in Figure 6(b), compare it with the original image in Figure 6(a), and answer Questions (a)-(c):

(a) Basic denoising methods (10%)

(1) Apply a linear filter to the noisy image. Compare the performance of two choices of the filter parameters – the uniform weight function and the Gaussian weight function under different filter sizes.

Figure 6: (a) Original Flower_gray image

(b) Flower_gray_noisy image

(b) Bilateral Filtering (10%)

In most low-pass linear filters, we often see degradation of edges. However, using some nonlinear filters, we can preserve the edges. Bilateral filters are such kinds of filters. A discrete bilateral filter is given by:

$$Y(i,j) = \frac{\sum_{k,l} I(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_c^2} - \frac{\|I(i,j) - I(k,l)\|^2}{2\sigma_s^2}\right)$$

where (k,l) is the neighboring pixel location within the window centered around (i,j), I is the image with noise, Y is the filtered image. σ_c and σ_s are two spread parameters.

(1) Implement the bilateral denoising filter and apply it to the noisy image.
(2) Explain the roles of σ_c and σ_s. Discuss the change in the filter's performance concerning the values of σ_c and σ_s.
(3) Does this filter perform better than linear filters you implemented in Problem 2(a)? Justify your answer in words.

(c) Non-Local Means (NLM) Filtering (10%)

The non-local mean filter utilizes the pixel value from a larger region rather than the mean of a local window centered around the target pixel. A discrete non-local mean filter with a Gaussian weighting function is as follows:

$$Y(i,j) = \frac{\sum_{k,l} I(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \exp\left(-\frac{\left\|I(N_{i,j}) - I(N_{k,l})\right\|_{2,a}^2}{h^2}\right)$$

where I, Y are the noisy and filtered images, respectively, $N_{x,y}$ is the window centered around location (x,y), and h is the filtering parameter, $N' \leq N$ and $M' \leq M$ denote the window size of your choice.
The Gaussian weighted Euclidian distance between window $I(N_{i,j})$and $I(N_{k,l})$ is defined as:

$$\left\|I(N_{i,j}) - I(N_{k,l})\right\|_{2,a}^2 = \sum_{n_1,n_2 \in \aleph} G_a(n_1,n_2)\left(I(i-n_1, j-n_2) - I(k-n_1, l-n_2)\right)^2$$

$$G_a(n_1,n_2) = \frac{1}{\sqrt{2\pi}a} \exp\left(-\frac{n_1^2+n_2^2}{2a^2}\right)$$

where $\aleph$ denotes the local neighborhood centered at the origin, $n_1, n_2 \in \aleph$ denotes the relative position in the neighborhood window. $a > 0$ is the standard deviation of the Gaussian kernel.

(1) Apply the NLM filter (using open-source code) to the noisy image. Try several filter parameters and discuss their effect on the filtering process. Clearly state your final choice of parameters in your report.
(Note that there are four parameters to discuss: the big search window size $\aleph$, the small neighbor window size N', the Gaussian smoothing parameter for the search window h, and the Gaussian smoothing parameter for the neighbor window a. If the open-source code doesn't provide ways to adjust a certain parameter, just analyze that parameter theoretically.)

(2) Compare the performance of NLM with filters used in Problem 2(a) and Problem 2(b).

(d) Denoising for color images (10%)

Figure 7 (b) is a noisy color image corrupted with mixed types of noises. Please identify noise types in the image and answer the following questions:

(1) What types of noises are there? Justify your answer.
(2) What filters would you like to use to remove mixed noise? Can you cascade these filters in any order? Justify your answer.
(3) 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.

(a) (b)

Figure 7: (a) the original Flower image (b) Flower_noisy image

Problem 3: Special Effect Image Filters: Creating Watercolor Painting Effect (30%)

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

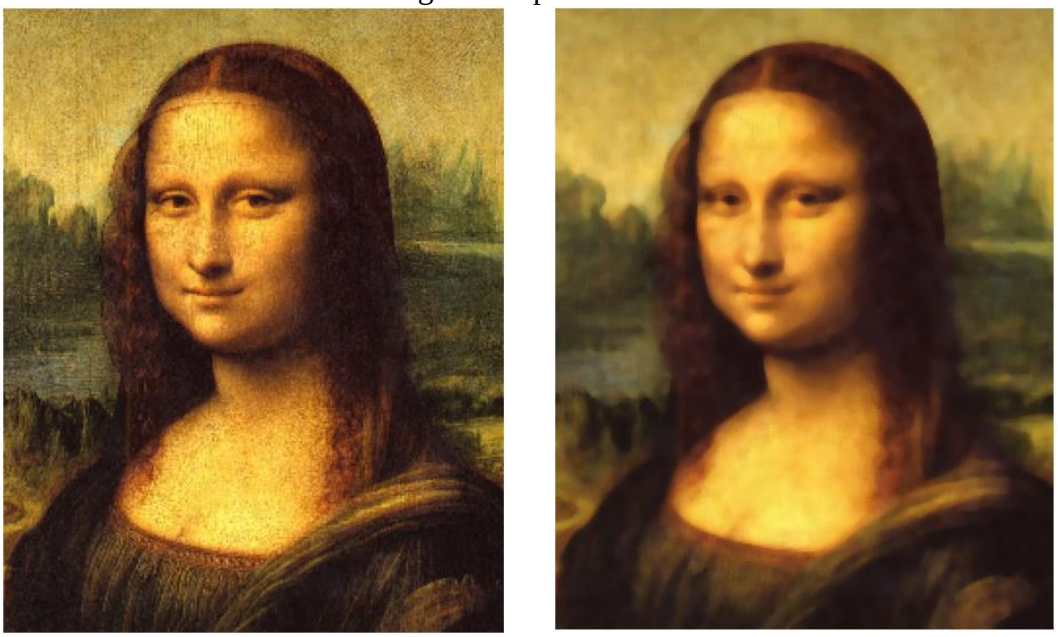


Figure 8 (a) Original color image (b) Intermediate image I_M

Step 1: Median Filtering. For each pixel of the original color image I, select the median pixel value 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_M. The result of $N = 5$shown in Figure 8.

Step 2: Bilateral Filtering. Apply the bilateral filter you implemented in **Problem 2 (b)** to the output image of **Step 1**. Set the window size =, 5, $\sigma_c = 20$ and $\sigma_s = 10$. Reapplying the bilateral filter to for K times (K=5). The output image is denoted by I_b .

Step 3: Gaussian Filtering. Apply a Gaussian blur to the original color image with a window size of 7 and a standard deviation of 2. The output image is denoted by I_G .

Step 4: Linear Combination. Linearly combine the **Step 2** output I_b and **Step 3** output I_G to yield the final output I_{Out} . The output image is calculated by $I_{Out} = 1.4 * I_b - 0.4 * I_G$.

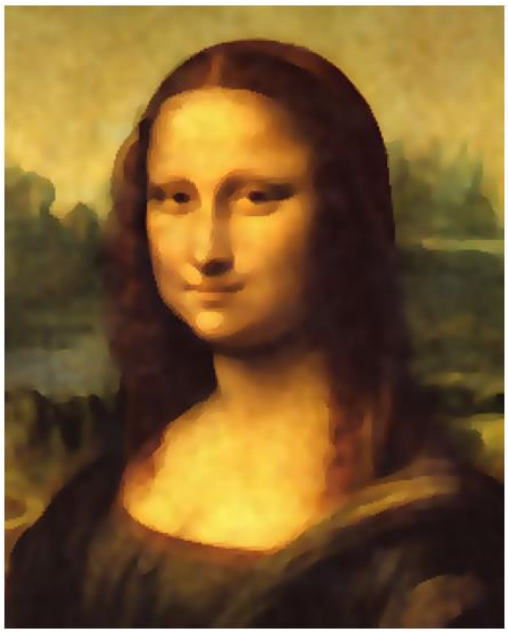


Figure 9: The Watercolor Painting effect

Implement and apply the watercolor painting filter to the image as shown in Fig. 7(a).

(1) Show the intermediate results I_M of **Step 1** and the final output I_{Out}**.**

(2) Implement the watercolor painting process as described with several different window sizes N in **Step 1** with N=7 and N=11. Show the corresponding I_{Out} results and explain your observation.

(3) What happens when the K in step 2 is set to 1 and 10? Show the corresponding I_{Out} results and explain your observation.

(4) Repeat (1) to (3) for the noisy image in Fig. 7(b) with your chosen setting in (2) and (3) for the clean image. Describe your observations and discuss whether the noise leads to the difference and why.

Appendix:

Problem 1: Image Demosaicing and Histogram Manipulation

House.raw	420x288	8-bit	gray
House_ori.raw	420x288	24-bit	color(RGB)
DimLight.raw	596x340	8-bit	gray
City.raw	750x422	24-bit	color(RGB)

Problem 2: Image Denoising & Problem 3

Flower.raw	768x512	24-bit	color(RGB)
Flower_noisy.raw	768x512	24-bit	color(RGB)
Flower_gray.raw	768x512	8-bit	gray
Flower_gray_noisy.raw	768x512	8-bit	gray

Note: “420x288” means “width=420, height=288”.

Convert from RGB to YUV color space

Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16
U = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128
V = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128

Convert from YUV to RGB color space

R = 1.164(Y - 16) + 1.596(V - 128)
G = 1.164(Y - 16) - 0.813(V - 128) - 0.391(U - 128)
B = 1.164(Y - 16) + 2.018(U - 128)

Reference Images

All images in this homework are from Google Images [4], the USC-SIPI image database [5] or others [6].

References

[1] M. E. Celebi et al. (eds.), Color Image and Video Enhancement.
[2] Zuiderveld, Karel. “Contrast Limited Adaptive Histograph Equalization.” Graphic Gems IV. San Diego: Academic Press Professional, 1994. 474–485.
[3] https://www.themathdoctors.org/averaging-angles/
[4] http://images.google.com/
[5] http://sipi.usc.edu/database/
[6] https://www.gettyimages.com