EE 569: Homework #1

Issued: 1/20/2021   Due: 11:59PM, 2/7/2021

General Instructions:

1. Read *Homework Guidelines* and *MATLAB Function Guidelines* for the 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 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, B values at each pixel.

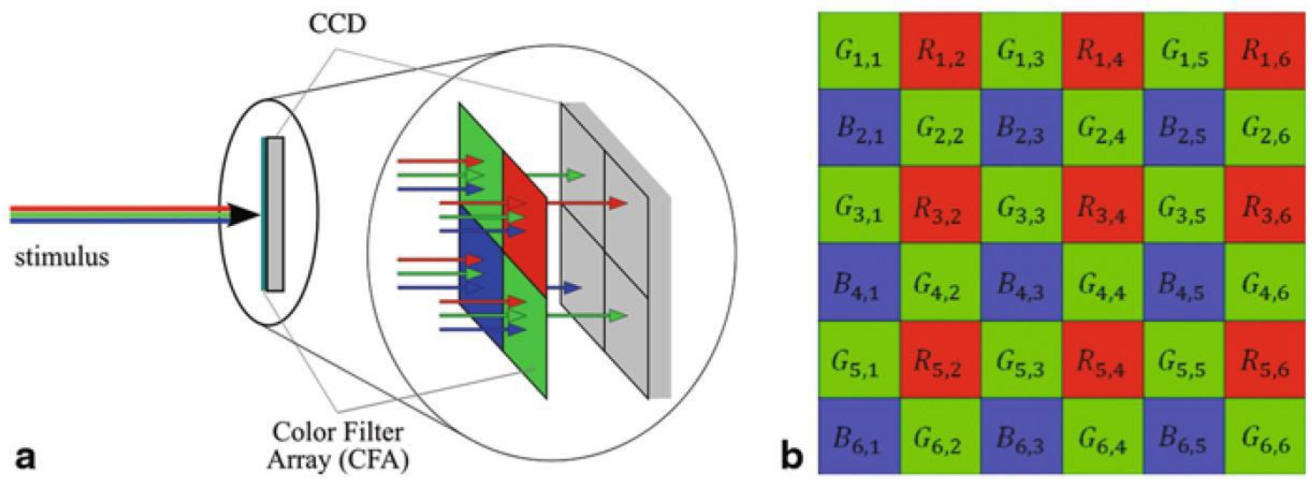

CCD
stimulus
Color Filter Array (CFA)
a
b
$G_{1,1}$ $R_{1,2}$ $G_{1,3}$ $R_{1,4}$ $G_{1,5}$ $R_{1,6}$
$B_{2,1}$ $G_{2,2}$ $B_{2,3}$ $G_{2,4}$ $B_{2,5}$ $G_{2,6}$
$G_{3,1}$ $R_{3,2}$ $G_{3,3}$ $R_{3,4}$ $G_{3,5}$ $R_{3,6}$
$B_{4,1}$ $G_{4,2}$ $B_{4,3}$ $G_{4,4}$ $B_{4,5}$ $G_{4,6}$
$G_{5,1}$ $R_{5,2}$ $G_{5,3}$ $R_{5,4}$ $G_{5,5}$ $R_{5,6}$
$B_{6,1}$ $G_{6,2}$ $B_{6,3}$ $G_{6,4}$ $B_{6,5}$ $G_{6,6}$

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:

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

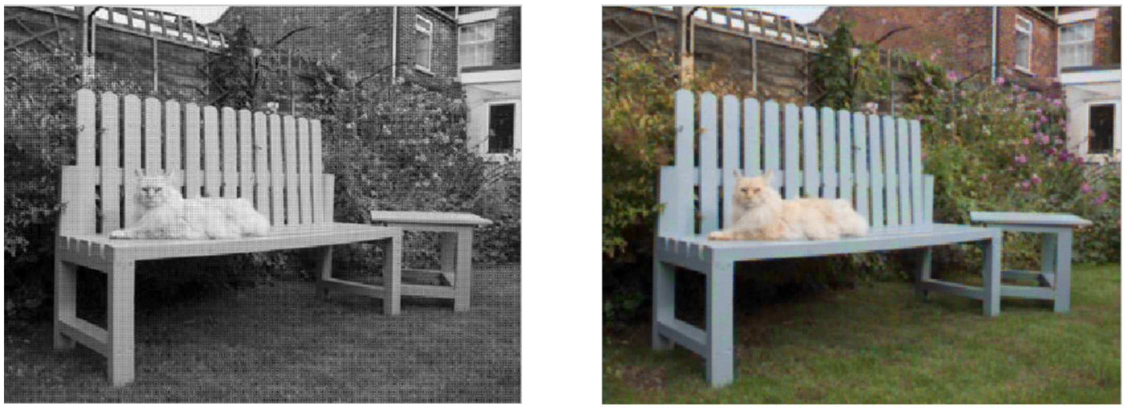

(a) (b)

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

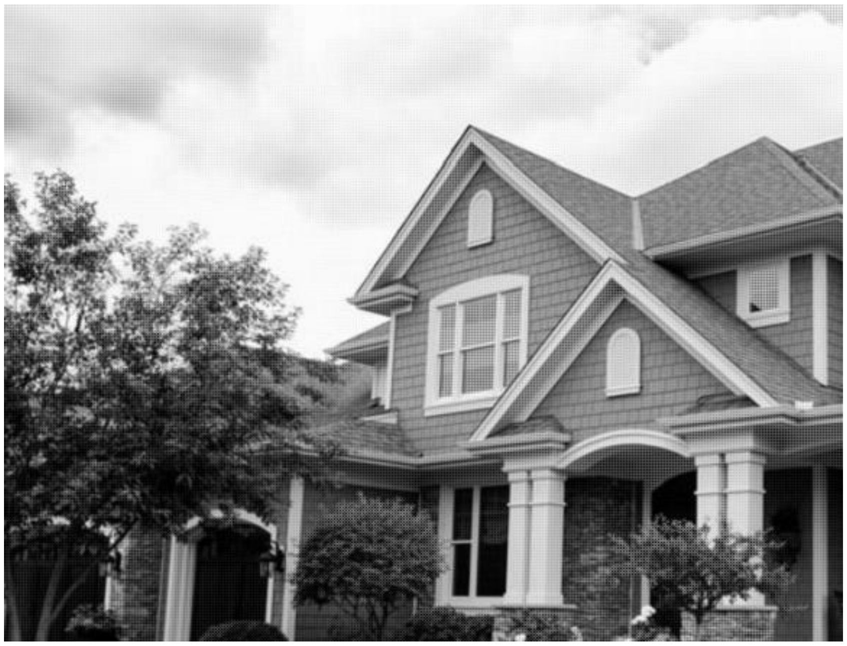


Figure 3: The *House* image before demosaicing.

(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 (20%)

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 *Toy* 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 function like imshow().

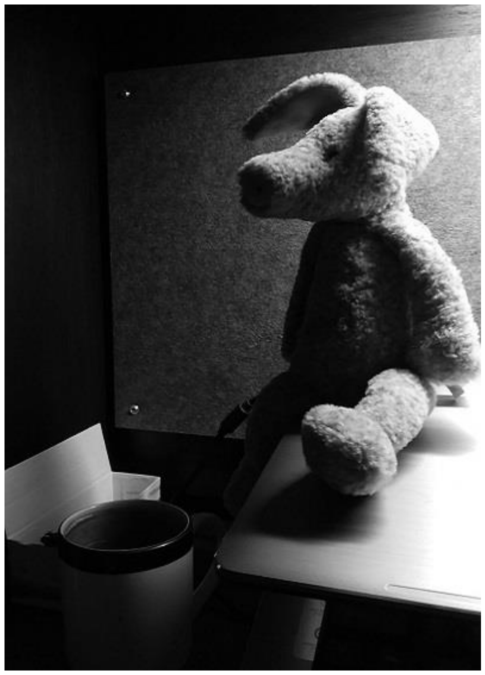


Figure 4: Toy image

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, 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 $N \times M$

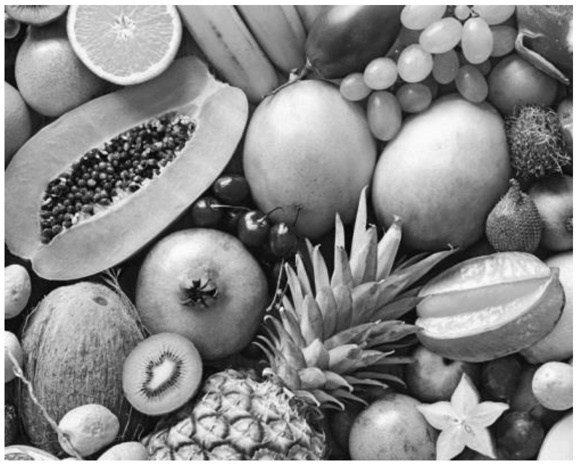
Max: Maximum possible pixel intensity = 255

Remove noise in the image in Figure 5(b), compare it with the original image in Figure 5(a), and answer the following questions:

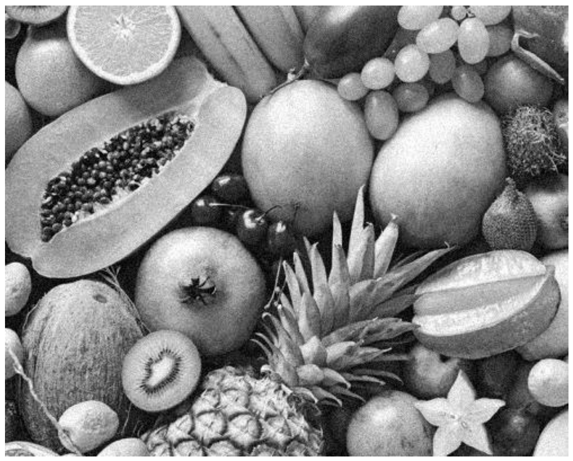
(a) Basic denoising methods (10%)

(1) What is the type of embedded noise in Figure 5(b)? Justify your answer.

(2) 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.



(a) Original image



(b) Noisy image

Figure 5: The original and noisy Fruits images.

(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 one such kind 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 filter's performance with respect to 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 the mean of a local window centered around the target pixel. A discrete non-local mean filter with 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, $N' \le N$ and $M' \le M$ 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:

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

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

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 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, 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) Mixed noises in color image (10%)

Figure 6 (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 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.

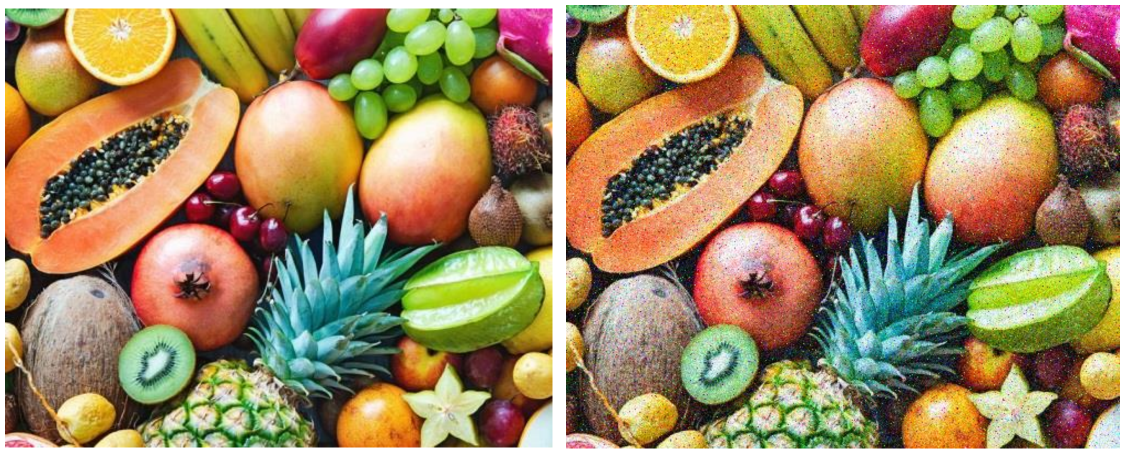


Figure 6: (a) the original Fruits image (b) the Fruits image with mixed noises.

Problem 3: Special Effect Image Filters: Creating Oil Painting Effect (30%)
An exemplary oil-painting effect for the *Flower* image is shown in Figure 8. This effect can be implemented as a filter with the following two steps:

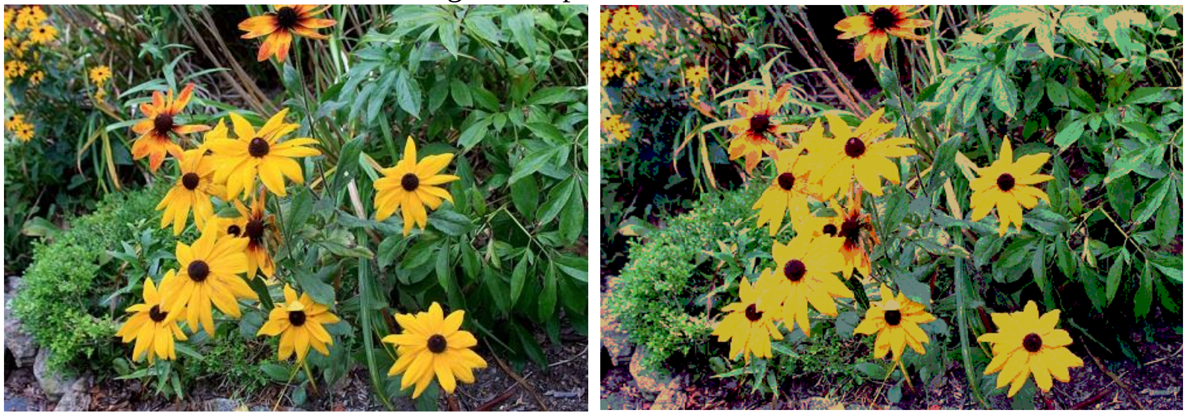

(a) Original color image
(b) Processed image with a reduced color set

Figure 7: images 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 *Flower* images are shown in Figs. 7 (a) and (b), respectively.
Step 2: For each pixel of image I_1, select the most frequent color in its NxN 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 of $N = 5$ is shown in Figure 8.

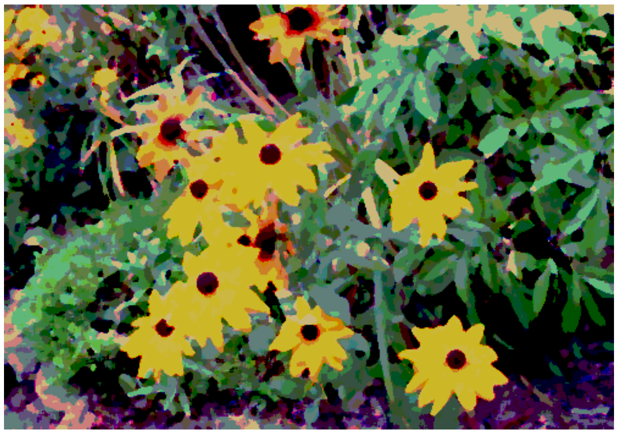


Figure 8: The Oil Painting effect

Implement and apply the oil-painting filter to the image as shown in Fig. 6(a).

(1) Show the 64-color version of the *Fruits* image by following Step 1. Describe how you do the quantization.

(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 when the input image has 512 colors instead? Show the corresponding results for step 1 and step2 and explain your observation.

(4) Repeat (1) to (3) for the noisy image in Fig. 6(b) with your chosen setting in (1) and (2) for the clean image. Describe your observations and discuss how the noise leads to the difference and how the number of colors may affect this.

Appendix:

Problem 1: Image Demosaicing and Histogram Manipulation

House.raw	580x440	8-bit	gray
House_ori.raw	580x440	24-bit	color(RGB)
Toy.raw	400x560	8-bit	gray

Problem 2: Image Denoising & Problem 3

Fruits.raw	500x400	24-bit	color(RGB)
Fruits_noisy.raw	500x400	24-bit	color(RGB)
Fruits_gray.raw	500x400	8-bit	gray
Fruits_gray_noisy.raw	500x400	8-bit	gray

Note: “580x440” means “width=580, height=440”.

Reference Images

All images in this homework are from Google images [2] or the USC-SIPI image database [3].

References

[1] M. E. Celebi et al. (eds.), Color Image and Video Enhancement.
[2] [Online] http://images.google.com/
[3] [Online] http://sipi.usc.edu/database/