EE 569: Homework #1

Issued: 8/31/2013 Due: 11:59PM, 9/22/2013

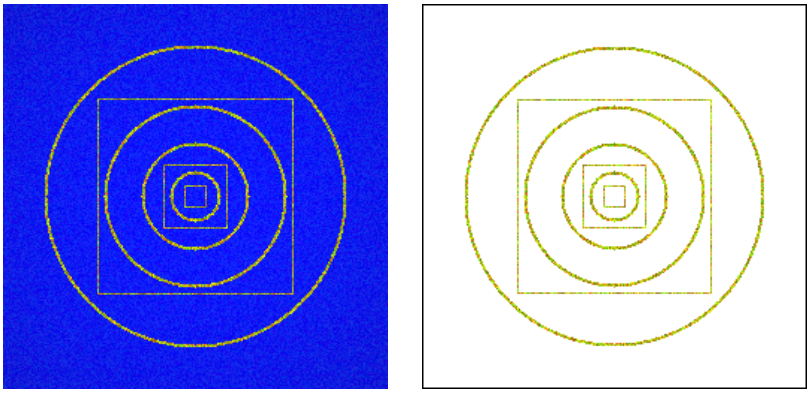
General Instructions:

Please refer to Homework Guidelines and MATLAB Function Guidelines for more information about how to complete and submit the homework. Also, refer to the USC policy on academic integrity and penalties for cheating and plagiarism - these rules will be strictly enforced. **If you make any assumptions about a problem, please clearly state them in your report.**

Problem 1: Image Manipulation (33 %)

In this problem, you will perform simple image manipulations such as accessing pixel values, implementing and interpreting histograms, etc.

1(a) Blue Screen Technique (Basic: 10%)



(a) Input Image: shapes.raw (b) Desired Output

Figure 1: Blue Screen Technique

The blue screen technique is often used to shoot special effect scenes in movies. The scene is shot with blue background that can be easily separated from the foreground. For example, Figure 1(a) has blue background that is different from the foreground consisting of geometrical shapes/objects. For this problem, you are required to separate the foreground from the background, and Figure 1(b) shows the desired output image. Note that the outside black border in Figure 1(b) is just used to indicate the image size. You do not have to add such a border in your output image. Please discuss your algorithm in detail with the aid of pertinent images/plots.

Hint: In order to identify the background color, you can analyze histograms of the three channels of the original image. Let us denote the background color by (R_b, G_b, B_b)*. It is quite possible that there is lot of*

variation in the background color. Then, instead of using single value of background color, you can select a range:

$$\{(R_b, G_b, B_b) \mid R_b \in [r_1, r_2], G_b \in [g_1, g_2], B_b \in [b_1, b_2]\}$$

Please clearly discuss your approach in the report.

1(b) Identifying Geometrical Objects (Basic: 6%)

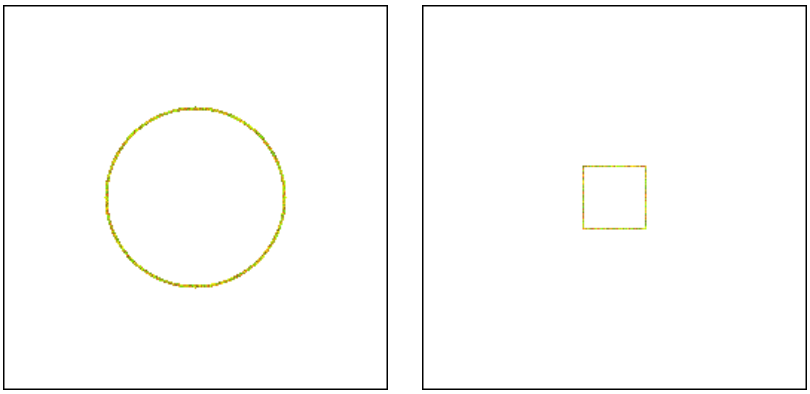
For this part, you need to write a function which counts the total number of geometrical objects in shapes.raw. The function should give three outputs:

1. The total number of circles, N_c, and their respective radii.
2. The total number of squares, N_s, and their respective side lenghts.
3. The total number of geometrical objects, $N_g = N_c + N_s$.

It is recommended that you use the output from Problem 1(a) as the input to this function. You can assume that both circles and squares are positioned at the center of the image. Please clearly discuss you algorithm and show intermediate results, if any, in your report.
Hint: You can traverse along the image center horizontally or vertically pixel by pixel to identify the cross point of each object with the x-axis or the y-axis. To identify the shape of the object, you can traverse the neighborhood of a cross point.

1(c) Extracting Geometrical Objects (Basic: 6%)

Besides identifying the number and the shape of objects in the image, you can separate different objects. Please write a function to extract any given object, circle or square, from shapes.raw (or your output from Problem 1(a)). The same function should be able to extract squares and circles. Use this function to extract the second largest circle and the second largest square from shapes.raw (as shown in Figure 2). Please discuss your algorithm in detail in your report and show intermediate results, if any.



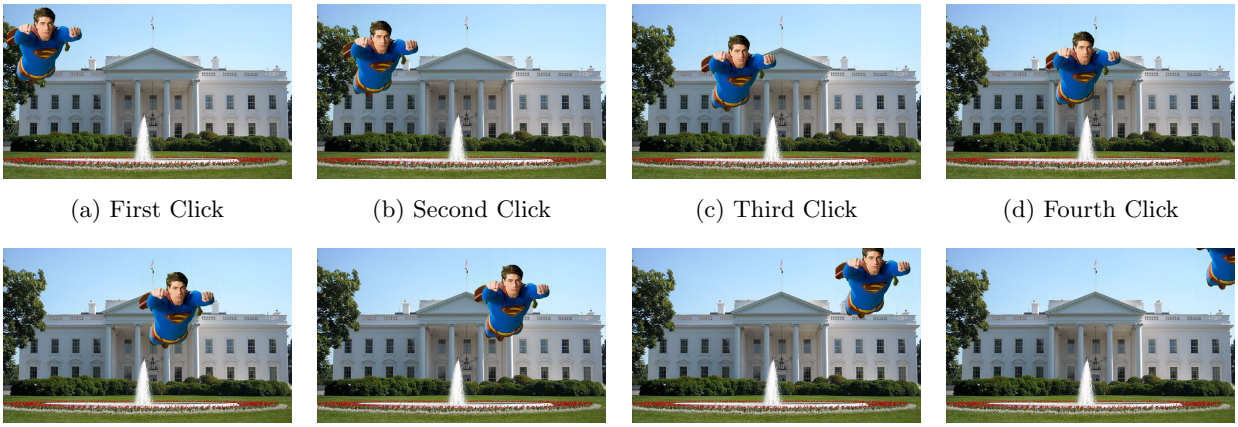
(a) Extracted second largest circle (b) Extracted second largest square

Figure 2: Desired Outputs

Hint: From Problem 1(b), you can deduce the diameter/side of circles/squares and determine the order in which they appear. Then, you can traverse a circle around the center of the image or a square object horizontally or vertically pixel by pixel. The traversal of an object is completed by revisiting the start pixel. You can use the traversal pattern to identify the shape of an object.

1(d) Smart Camera - Moving Objects Removal (Advanced: 11%)

Tommy Trojan, on his trip to Washington, snapped a few shots of the White House. After returning home, he realized that all photos have been spoiled by a faux superman (see Figure 3).



(a) First Click (b) Second Click (c) Third Click (d) Fourth Click

(e) Fifth Click (f) Sixth Click (g) Seventh Click (h) Eighth Click

Figure 3: Tommy's Mishap

Tommy does not like faux-man spoiling his White House pictures. Please help Tommy generate a picture of the White House without the faux-man in it (See the desired output in Figure 4). Note that many cameras nowadays come with such a feature. However, the actual algorithms used by these cameras are beyond the scope of this course. However, you can still use simple image manipulations to solve the problem.
Hint: You may first detect the common content of these multiple images and, then, fill in the area occupied by the faux-man with proper content. This is also known as the image imparting technique.
Aside Note: Images in Figure 3 were generated using the blue screen technique.

Figure 4: Desired Output of The White House

Finally, Tommy wants to create a fun photograph in which the superman is being jumped over by a dog. Use dogjump.raw to create the desired photograph as shown in Figure 5(b). To achieve this goal, you should first extract superman's photo from Figure 3, which would be a byproduct of cleaning the White House image. The exact placement of the superman image in dogjump.raw is less a concern, and you can choose the desired location with proper co-ordinates manually. Please clearly discuss your approach in the report along with the final used co-ordinates.

(a) dogjump.raw (b) Desired Output

Figure 5: Tommy's fun photography!

Problem 2: Image Enhancement (32 %)

In this problem, you would apply the histogram manipulation for image enhancement.

2(a) Histogram Equalization for Grayscale Images (Basic: 10%)

Please implement the full-range linear scaling method and the cumulative histogram equalization method to enhance the contrast of three Barbara images in Figure 6. Describe the procedure and show the resulting images. Comment and compare the performance of these two methods. Plot the histograms of all images (3 input and 6 output images), and the transfer functions (6 functions). Please note that MATLAB users CANNOT use functions like *imhist()*, *hist()*, and other similar functions from Image Processing Toolbox. Use of *imhosw()*, however, is permitted.

(a) barbara_dark.raw (b) barbara_mid.raw (c) barbara_bright.raw

Figure 6: Barbara Image - (a) Dark, (b) Low Contrast, and (c) Bright

2(b) Histogram Equalization for Color Images (Basic: 10%)

Please implement two histogram equalization techniques (the equalized "cumulative" histogram method and the equalized histogram method) to enhance the contrast of Ocean image as shown in Figure 7. Histogram equalization can be performed separately on the red, green and blue channels for color images. Compare and comment on the output images and equalized histograms of these methods.

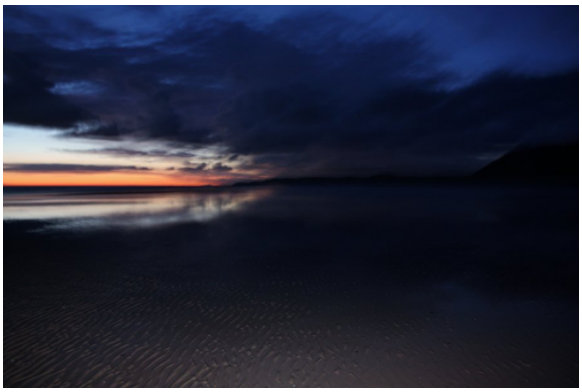


Figure 7: Ocean Low Contrast Image (ocean_contrast.raw)

2(c) Histogram Transform (Advanced: 12%)

All histogram equalization methods you have implemented up to this point can be considered as histogram matching, where you try to "match" the histogram of the source image to a "target histogram" which happens to be uniformly distributed, *i.e.* an equalized histogram with probability of any intensity i, $p(i) = 1/N$, where N is the number of intensity values (For an 8-bit image, $N = 2^8 = 256$, *i.e.* intensity $i \in [0, 255]$). However, it is sometimes desirable that the target histogram be of a specific shape (not necessarily flat).

Figure 8: Barbara high contrast (barbara_contrast.raw)

Here, you will implement one such algorithm by matching the histogram of a high contrast Barbara image shown in Figure 8 to that of a Gaussian distribution function. The probability denisty function of a Gaussian distrubition with mean μ and standard deviation σ is given by

$$p(x/\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{1}$$

Construct and plot a target histogram for 8-bit image using Equation (1) with $\mu = 70$ and $\sigma = 20$. Since we have an 8-bit image, we will truncate the pdf outside $[0, 255]$. Due to truncation, the pdf will not sum to 1 and, hence, we must renormalize it so that

$$\sum_{x=0}^{255} p(x/\mu, \sigma) = 1. \tag{2}$$

Plot the histogram of the high contrast Barbara image as shown in Figure 8. Please conduct the histogram transform so that the contrast enhanced Barbara image has a histogram similar to $p(x/\mu, \sigma)$. Please discuss your method with the help of plots for the output image, its transfer function, and the enhanced histogram. Repeat the same procedure with $\mu = 120$ and $\sigma = 50$. Compare the results of histogram equalization for $p(x/\mu = 70, \sigma = 20)$ and $p(x/\mu = 120, \sigma = 50)$. How do the modified mean and standard deviation affect your results and why? Feel free to play with the parameters if needed.

Problem 3: Noise Removal (35%)

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

In this part, you will perform noise removal on a color image corrupted with "mix" type of noise. The original and noisy pepper images are shown in Figure 9. Since the image has mixed noise, you will need more than one noise removal filters.

(a) The Pepper image

(b) The noisy Pepper image (peppers_mix.raw)

Figure 9

1) Please identify noise types in the 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 use to remove mixed noise? d) Can you cascade these filters in any order (justify your answer). e) Please discuss the affect of different filter window sizes.
2) Try to get the best results in removing mixed noise: a) describe your method and discuss its short-comings. b) Give some suggestions to improve its performance. There is no need to implement but explain them along with your motivation. You can use suitable plots/images to support your arguments.

3(b) Bilateral Filtering (Basic: 8%)

In most low-pass linear filters, we often see degradation of edges. However, using some non-linear filters, we can preserve the edges. Bilateral filters are one such kind of filters. A bilateral filter, in its most general form, can be given by [1]:

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \mathbf{f}(\xi)\ c(\xi, \mathbf{x})\ s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))\quad d\xi \tag{3}$$

where $c(\xi, \mathbf{x})$ and $s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))$ measures, respectively, the spatial closeness and the pixel intensity closeness (*photometric* similarity) between point $\mathbf{x}$ and its nearby point ξ and

$$k(\mathbf{x}) = \int_{-\infty}^{\infty} c(\xi, \mathbf{x})\ s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x}))\quad d\xi$$

is a normaliztion factor. The discrete version of a bilateral filter is given by:

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \sum_{\xi \in \Omega_{\mathbf{x}}} \mathbf{f}(\xi)\ c(\xi, \mathbf{x})\ s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) \tag{4}$$

where $\Omega_{\mathbf{x}}$ is a finite window centered around $\mathbf{x}$. The Gaussian function is often chosen for both $c(.)$ and $f(.)$ so that we get

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \sum_{\xi \in \Omega_{\mathbf{x}}} \mathbf{f}(\xi) \ \exp\left(-\frac{\|\xi - \mathbf{x}\|_2^2}{2\sigma_c^2}\right) \ \exp\left(-\frac{\|\mathbf{f}(\xi) - \mathbf{f}(\mathbf{x})\|_2^2}{2\sigma_s^2}\right), \tag{5}$$

where $\|\cdot\|_2^2$ denotes the square 2-norm (Eucledean norm), and σ_c and σ_s are spread parameters (analogous to standard deviation in Eq. (1)).

1) Implement the Bilateral filter and apply it to lena.raw (shown in Figure 10).

2) Explain the role of functions c and s. Discuss the change in filter's performance with respect to the values of σ_c, σ_s and the window size (*i.e.* size of $\Omega_{\mathbf{x}}$).

3) Does this filter perform better than linear filters you implemented in Problem 3(a)? Justify your answer in words.

Figure 10: Noisy Lena (lena.raw)

3(c) Non-Local Means (NLM) Filtering (Advanced: 12%)

For this part, you may refer to [2].

1) Briefly explain the Non-Local Means Algorithm (NLM) and implement it. Please DO NOT use any code from the Internet or other sources, as it would be considered as plagiarized.

2) Apply NLM filtering to lena.raw (Figure 10). Try several filter parameters and discuss their effect on filtering process. Clearly state your final choice of parameters in your report.

3) Compare the performance of NLM with filters used in Problem 3(a) and Problem 3(b). Can you deduce a relationship between the NLM filter and the bilateral filter (with respect to s and c in Eq. (4))?

Note: DO NOT quote statements directly from [1, 2] or any other online source(s). Explain in your own words. Reports and source codes are subject to verification for any plagiarism.

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

In this part, you would get familiar with a state-of-the-art filtering method 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 lena.raw

(Figure 10). 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). Discuss the effects of several tunable parameters on the denoising result.
2) Both NLM and bilateral filter are spatial domain filters. How would you classify BM3D:- spatial domain, frequency domain, or both? Justify your answer.
3) Conduct qualitative performance comparison between NLM, bilateral filtring, and BM3D.

Appendix:

Problem 1: Image Manipulation

shapes.raw	$257 \times 257 \times 3$, 8-bit depth, inteleaved RGB image
firstclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
secondclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
thirdclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
fourthclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
fifthclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
sixthclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
seventhclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
eighthclick.raw	$750 \times 453 \times 3$, 8-bit depth, inteleaved RGB image
dogjump.raw	$900 \times 507 \times 3$, 8-bit depth, inteleaved RGB image

Problem 2: Image Enhancement

barbara_dark.raw	512×512, 8-bit depth, grayscale image
barbara_mid.raw	512×512, 8-bit depth, grayscale image
barbara_bright.raw	512×512, 8-bit depth, grayscale image
ocean_contrast.raw	$500 \times 332 \times 3$, 8-bit depth, inteleaved RGB image
barbara_contrast.raw	512×512, 8-bit depth, grayscale image

Problem 3: Noise Removal

peppers_mix.raw	$256 \times 256 \times 3$, 8-bit depth, inteleaved RGB image
lena.raw	256×256, 8-bit depth, grayscale image

Sample Codes Provided

readraw.m	MATLAB source code provided to read in grayscale raw image files
writeraw.m	MATLAB source code provided to output grayscale raw image files
readraw.cpp	C++ code provided to read in and output grayscale raw image files

Reference Images

All images used in this homework were downloaded either from Google image search [5] or USC-SIPI image database [6].

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on.* IEEE, 1998, pp. 839–846.

[2] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 60–65.

[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]. Available: http://images.google.com/

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