EE 569: Homework #2

Issued: 2/4/2018 Due: 11:59PM, 2/25/2018

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. You need to understand the USC policy on academic integrity and penalties for cheating and plagiarism. These rules will be strictly enforced.

Problem 1: Geometric Image Modification (30%)

In this problem, you will need to apply geometric modification and spatial warping techniques to do some interesting image processing tricks. Please note that during these operations, you may need to solve some linear equations to get the matrix parameters.

(a) Geometrical Warping (Basic: 15%)

Design and implement a spatial warping technique that transforms an input square image into an output image of a disk-shaped image. An example is given in Figure 1.

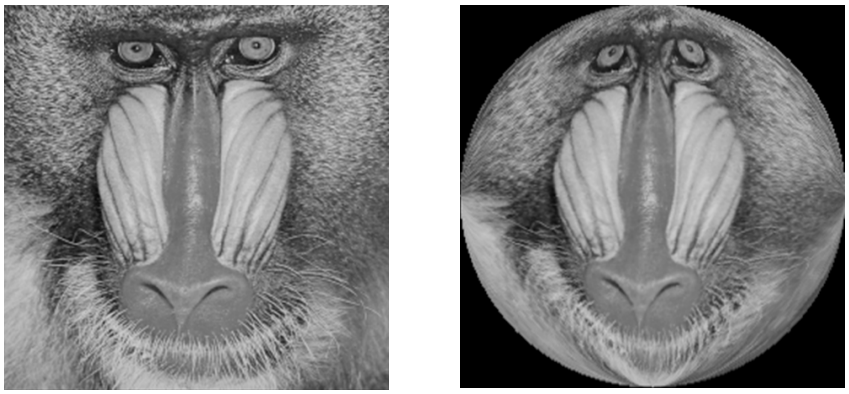

Figure 1: Warp the original image to a disk-shaped image

The wrapped image should satisfy the following three requirements:

- Pixels that lie on boundaries of the square should still lie on the boundaries of the circle.
- The center of original images should be mapped to the center of warped images.
- The mapping should be reversible, i.e. it is a one-to-one mapping.

Apply the same developed spatial warping algorithm to *puppy*, *tiger* and *panda* images in Figure 2. Please first describe your approach as clearly as possible and show the resulting images.

Next, apply the reverse spatial warping to each warped image to recover its original image. Compare the recovered square image with the original square image. Is there any difference between two images? If any, explain sources of distortion in detail.

Figure 2: The puppy, tiger and panda images

(b) Homographic Transformation and Image Stitching (Advanced: 15%)

One can use homographic transformation and image stitching techniques to create panorama that consisting of multiple images. One example (Taken from matlab examples [1]) is shown in Figure. 3. The left image were taken with an uncalibrated smart phone camera by sweeping the camera from left to right along the horizon to capture all parts of the building. The right panorama is the desired output by stitching transformed images.

This example involves five images to composite. However, the basic principle is to process in terms of consecutive pair of images. It could be achieved by following these steps:

- Select control points from both images. You are allowed to do it manually for this step.
- Apply homographic transformation to find a homography mapping (described below).
- Wrap one image onto the other using the estimated transformation.
- Create a new image big enough to hold the panorama and composite the wrapped image into it. You can composite by simply averaging the pixel values where the two images overlap.

The homographic transformation procedure is stated below. Images of points in a plane, from two different camera viewpoints, under perspective projection (pin hole camera models) are related by a homography:

$$P_2 = HP_1$$

where H is a 3x3 homographic transformation matrix, P_1 and P_2 denote the corresponding image points in homogeneous coordinates before and after the transform, respectively. Specifically, we have

$$\begin{bmatrix} x_2' \\ y_2' \\ w_2' \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad and \quad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{x_2'}{w_2'} \\ \frac{y_2'}{w_2'} \end{bmatrix}$$

To estimate matrix H, you can proceed with the following steps:

- Fix $H_{33} = 1$ so that there are only 8 parameters to be determined.

- Select four point pairs in two images to build eight linear equations.
- Solve the equations to get the 8 parameters of matrix H.
- After you determine matrix H, you can project all points from one image to another by following the backward mapping procedure and applying the interpolation technique.

Implement above homographic transformation and stitching techniques to composite the *room* images in Figure 4. Show the results and make discussion on the following questions.

1) How many control points were used? What if we use more than four control points?
2) How did you select control points? Clearly specify your observations and strategies.

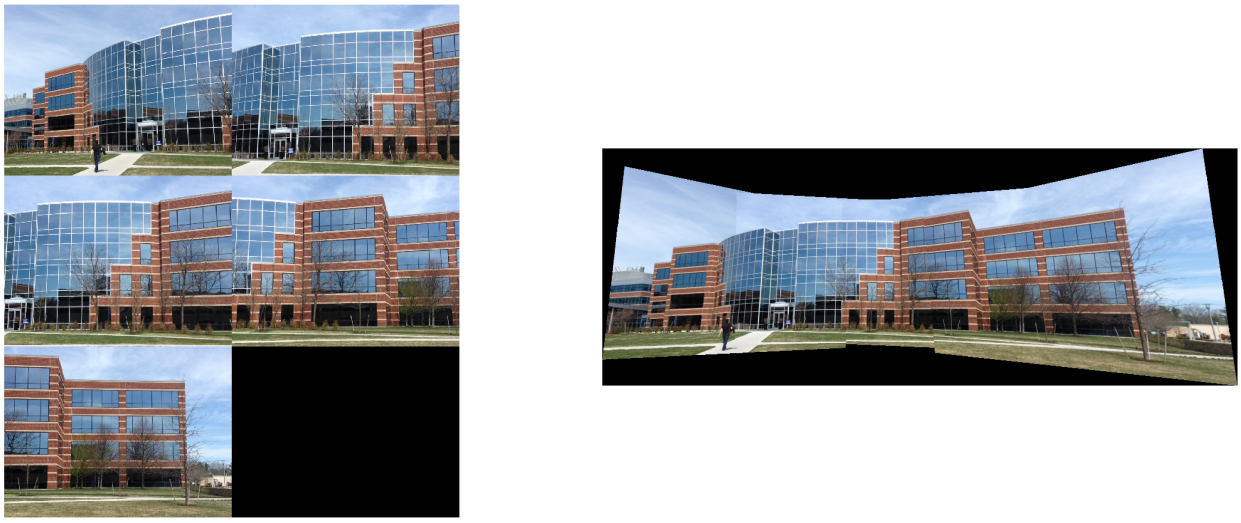

Figure 3: An example of homographic transformation and image stitching

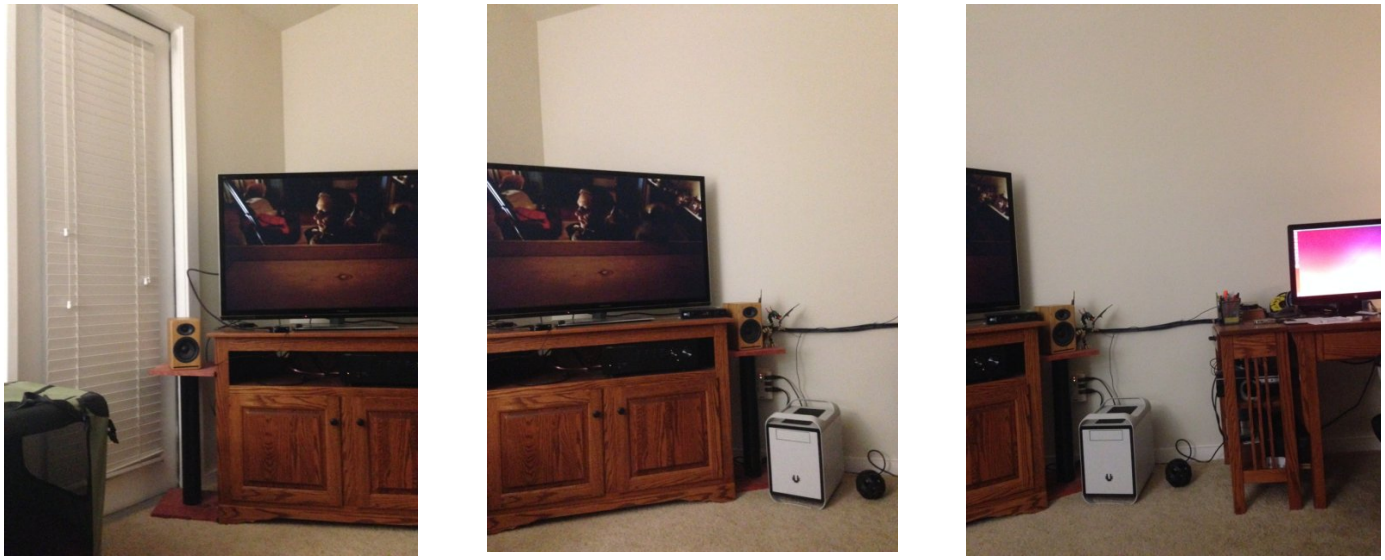

Figure 4: The room images (left, middle and right).

Problem 2: Digital Half-toning (30+10%)

3(a) Dithering (Basic: 20%)

Implement the following four methods to convert *colorchecker* to half-toned images. There are 256 gray levels for pixels in *colorchecker* image. In the following discussion, *F(i,j)* and *G(i,j)* denote the pixel of the input and the output images at position (*i,j*), respectively. **Compare the results obtained by these algorithms in your report.**

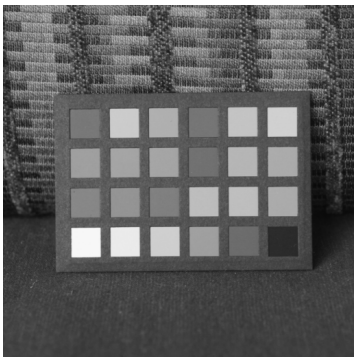

Figure 5: The colorchecker image

1. Fixed thresholding

Choose one value, T, as the threshold to divide the 256 levels into two ranges. An intuitive choice of T would be 127. For each pixel, map it to 0 if it is smaller than T, otherwise, map it to 255, i.e.

$$G(i,j) = \begin{cases} 0 & \text{if } 0 \le F(i,j) < T \\ 255 & \text{if } T \le F(i,j) < 256 \end{cases}$$

2. Random thresholding

In order to break the monotones in the result from fixed thresholding, we may use a 'random' threshold. The algorithm can be described as:

- For each pixel, generate a random number in the range 0 ~ 255, so called *rand(i,j)*
- Compare the pixel value with *rand(i,j)*. If it is greater, then map it to 255; otherwise, map it to 0, i.e.

$$G(i,j) = \begin{cases} 0 & \text{if } rand(i,j) \le F(i,j) \\ 255 & \text{if } rand(i,j) > F(i,j) \end{cases}$$

The built-in rand function in C/C++/Matlab generate numbers in uniform distribution.

3. Dithering Matrix

Dithering parameters are specified by an index matrix. The values in an index matrix indicate how likely a dot will be turned on. For example, an index matrix is given by

$$I_2(i,j) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

where 0 indicates the pixel most likely to be turned on, and 3 is the least likely one. This index matrix is a special case of a family of dithering matrices first introduced by Bayer [2]. The Bayer index matrices are defined recursively using the formula:

$$I_{2n}(i,j) = \begin{bmatrix} 4 * I_n(x,y) + 1 & 4 * I_n(x,y) + 2 \\ 4 * I_n(x,y) + 3 & 4 * I_n(x,y) \end{bmatrix}$$

The index matrix can then be transformed into a threshold matrix T for an input gray-level image with normalized pixel values (*i.e.* with its dynamic range between 0 and 1) by the following formula:

$$T(x,y) = \frac{I(x,y) + 0.5}{N^2}$$

where N^2 denotes the number of pixels in the matrix. Since the image is usually much larger than the threshold matrix, the matrix is repeated periodically across the full image. This is done by using the following formula:

$$G(i,j) = \begin{cases} 1 & \text{if } F(i,j) > T(i \bmod N, j \bmod N) \\ 0 & otherwise \end{cases}$$

where *G(i, j)* is the normalized output binary image.

Answer the following questions:

1. Please create $I_2(i,j)$, $I_4(i,j)$ and $I_8(i,j)$ thresholding matrices and apply them to halftone the *colorchecker* image.
2. If a screen can only display FOUR intensity levels, design a method to generate a display-ready *Colorchecker* image. Show your best result in gray-scale with four gray-levels (0, 85, 170, 255) and explain your design idea and detailed algorithm.

3(b) Error Diffusion (Basic: 10%)

Convert the 8-bit *colorchecker* image to a half-toned one using the error diffusion method. Show the outputs of the following three variations, and discuss these obtained results. Compare these results with dithering matrix. Which method do you prefer? Why?

1. Floyd-Steinberg's error diffusion with the serpentine scanning, where the error diffusion matrix is:

$$\frac{1}{16}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

2. Error diffusion proposed by Jarvis, Judice, and Ninke (JJN), where the error diffusion matrix is:

$$\frac{1}{48}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

3. Error diffusion proposed by Stucki, where the error diffusion matrix is:

$$\frac{1}{42}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

Describe your own idea to get better results. There is no need to implement it if you do not have time. However, please explain why your proposed method will lead to better results.

3(c) Color Halftoning with Error Diffusion (Bonus 10%)

Figure 6: The flower image

1) Separable Error Diffusion

One simple idea to achieve color halftoning is to separate an image into CMY three channels and apply the Floyd-Steinberg error diffusion algorithm to quantize each channel separately. Then, you will have one of the following 8 colors, which correspond to the 8 vertices of the CMY cube at each pixel:

W = (0,0,0), Y = (0,0,1), C = (0,1,0), M = (1,0,0),
G = (0,1,1), R = (1,0,1), B = (1,1,0), K = (1,1,1)

Note that (W, K), (Y, B), (C, R), (M, G) are complementary color pairs. Please show and discuss the result of the half-toned color *flower* image. What is the main shortcoming of this approach?

2) MBVQ-based Error diffusion

Shakad et al. [3] proposed a new error diffusion method to overcome the shortcoming of the separable error diffusion method. They partition the CMY color space into six Minimum Brightness Variation Quadrants (MBVQ) as shown in Fig. 2 of [3]. Then, the MBVQ-based error diffusion can be conducted as follows. Given the CMY value and the quantization error at a pixel located in (x, y). They are denoted by RGB(x, y) and e(x, y), respectively.

- Determine its MBVQ quadrant based on RGB (x, y);
- Find the vertex V of the MBVQ tetrahedron to closest CMY (x, y) + e (x, y) and output the value of V at location (x, y);
- Compute the new quantization error using CMY (x, y) + e (x, y) - V
- Distribute the quantized error to the future pixel error buckets e using a standard error diffusion process (e.g. the FS error diffusion).

You may refer to [3] for more details.

Implement this algorithm and apply it to the *flower* image in Fig. 6. Compare the output with that obtained in (1). Discuss the difference between these two methods.

Problem 3: Morphological Processing (40%)

In this problem, you will implement three morphological processing operations: shrinking, thinning, and skeletonizing. A pattern table (patterntables.pdf) is attached for your reference. Please show outputs for all following parts in your report and discuss them thoroughly. Please state any assumptions you make in your solution.

(a) Shrinking (Basic: 8%)

Please apply the 'shrinking' filter to the squares image (stars.raw). Please implement the filter, and discuss your solution for the following questions:

- Count the total number of stars in the image.
- How many different square sizes are present in the image? What is the frequency of these star sizes? (Hint: Plot the histogram of the square size with respect to frequency.)

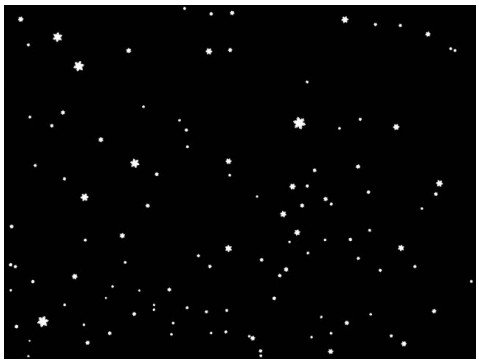

Figure 7: stars.raw

(b) Thinning (Basic: 8%)

Please apply the 'thinning' filter to the jigsaw image (jigsaw_1.raw) and show your result.

Figure 8: jigsaw_1.raw

(c) Skeletonizing (Basic: 8%)

Please apply the 'skeletonizing' filter to the jigsaw image (jigsaw_2.raw) and show your result.

Figure 9: jigsaw_2.raw

(d) Counting game (Advanced: 16%)

Figure 10 (see below) is target image with 16 jigsaw pieces (the background is white). Some of these pieces are obtained from geometrical operations, i.e. rotation, flipping, etc. You need to design an algorithm that uses morphological and logical operations to solve and answer the questions below. You may use any of these operators that you learned in class or ones that you invent, but all of them must be detailed in your report with respect to how they operate and how their results help you solve the problem. In your report, discuss your algorithm, results, and analysis in detail. You should also submit your MATLAB or C/C++ programs along with your source code submission. State any assumptions you make for this problem.

Note: Do NOT use any built-in morphological processing functions. Manual counting is not permitted; your code must automatically do this.

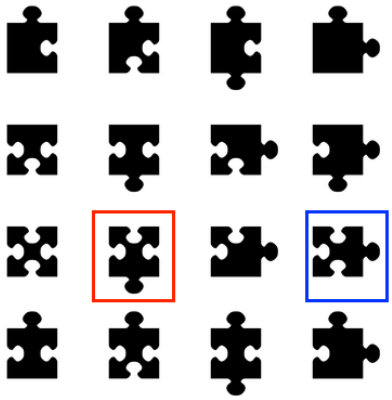

Figure 10: board.raw

- Count the number of pieces in the board image. You may take pieces in red and blue bounding box are different for this question.
- Find the number of unique pieces in the board image. Pieces in red and blue bounding box are identical because they are rotated version of each other.

Note: Hits will be given during discussion sessions.

Appendix:

Problem 1: Geometric image modification			
puppy.raw	512x512	24-bit	color(RGB)
tiger.raw	512x512	24-bit	color(RGB)
panda.raw	512x512	24-bit	color(RGB)
left.raw	480x640	24-bit	color(RGB)
middle.raw	480x640	24-bit	color(RGB)
right.raw	480x640	24-bit	color(RGB)
Problem 2: Digital halftoning			
colorchecker.raw	512x512	8-bit	gray
flower.raw	700x700	24-bit	color(RGB)
Problem 3: Morphological Processing			
stars.raw	640x480	8-bit	gray
jigsaw_1.raw	100x100	8-bit	gray
jigsaw_2.raw	100x100	8-bit	gray
board.raw	372x372	8-bit	gray

Reference Images

All images in this homework are from Google images [4], the MCL-JCI dataset [5] and the USC-SIPI [6] image database.

Reference

[1] https://www.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html
[2] B. E. Bayer, “An optimum method for two-level rendition of continuous-tone pictures,” SPIE MILE- STONE SERIES MS, vol. 154, pp. 139–143, 1999.
[3] D. Shaked, N. Arad, A. Fitzhugh, I. Sobel, "Color Diffusion: Error-Diffusion for Color Halftones", HP Labs Technical Report, HPL-96-128R1, 1996.
[4] http://images.google.com/
[5] http://mcl.usc.edu/mcl-jci-dataset/
[6] http://sipi.usc.edu/database/