EE 569: Homework #2

Issued: 9/20/2013 Due: 11:59PM, 10/13/2013

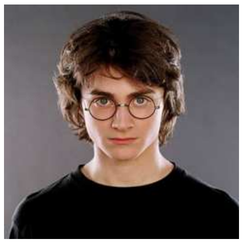
General Instructions:

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: Special Effect Image Filters (35 %)

In this problem, you will implement special effect filters, such as the pencil sketch filter, using edge detection operators. Then, you should apply all special effect filters (Parts [a]-[d]) to three test images, 'naruto.raw','harrypotter.raw' and 'lena-noisy.raw' as shown in Figure 1.

(a) naruto.raw


(b) harrypotter.raw

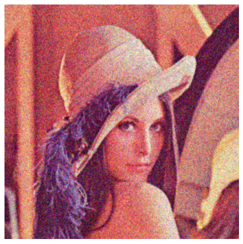

(c) lena-noisy.raw

Figure 1: Test Images*

1(a) Color to Gray-Scale Conversion (Basic: 2%)

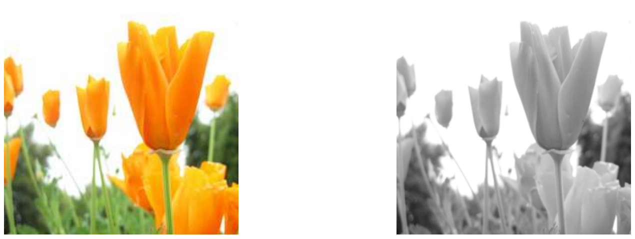
In this part, you will convert a color image into a gray-scale image using the following formula:

$$Y = 0.299R + 0.587G + 0.114B,$$

where Y, R, G, B are gray, red, green and blue channels, respectively. An example for flowers.raw is shown in Figure 2.

1(b) Pencil Sketch using edge detectors (Basic: 8%)

In this part, you will create a pencil sketch filter by applying an edge detection algorithm to the grayscale image. Please implement two basic edge detection algorithms: i) the 1st-order derivative gradient method



(a) Input Image: flowers.raw (b) Grayscale of flowers.raw

Figure 2: Color to Grayscale

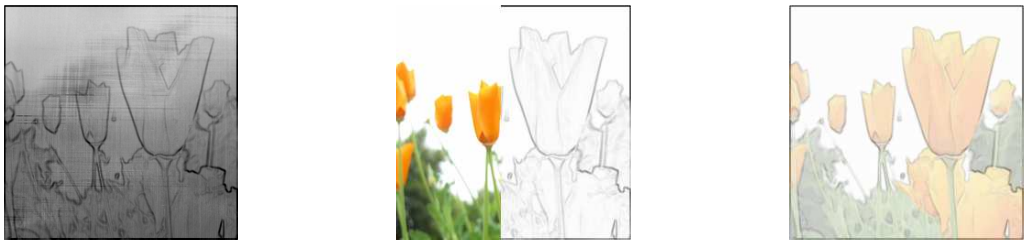
and ii) the 2nd-order derivative plus zero crossing. Please discuss how you choose the threshold values which are required in each of the two algorithms. Also, if needed, perform some simple post-processing such as noise removal on the pencil-sketch images. Please discuss your results on the three test images in Figure 1 and the challenges of using edge detection based algorithms for creating pencil sketch special effect.

1(c) Background Special Effect (Basic: 5%)

In this part, you will improve the pencil-sketch special effect by adding grainy background. An example is shown in Figure 3(a). This process can be described as:

$$I^{'}(i,j,k) = I(i,j,k) + \alpha * G(i,j,k) + \beta, \tag{1}$$

where $I(i,j,k)$ is the value of pixel at (i,j) and channel k for the pencil-sketch image, $G(i,j,k)$ is the value of grainy background image (grainy.raw) at pixel (i,j) and channel k, α and β are parameters with suggested values in the range $\alpha \in (-.75, -0.95)$ and $\beta \in (10, 30)$, respectively. Apply the background effect to the pencil sketch of all three test images. Please feel free to play with these parameters to get a good background effect. Show three different results obtained by varying α and β in your report.



(a) Background Special Effect (b) Sliding Transition Effect (c) Fade-in Transition Effect

Figure 3: Special effects on flowers.raw

1(d) Transition Special Effects (Advanced: 10%)

In this part, you will create two transition effects on the pencil sketch images and produce short video clips to illustrate these transition effects.

Prob 1(d) continued on next page. . .

- The **sliding transition** is a special effect that is obtained by a simple combination of the original color input image and the pencil sketch image. To obtain this effect, you will sweep through the pencil sketch image in a certain order and replace the swept part of the pencil sketch image with the original image. An example of the sweeping snapshot is shown in Figure 3(b). There are several ways to sweep through the pencil sketch image. Some examples are shown in Figure 4. Please implement at least two sweep directions to showcase your sliding transition effects on all three test images.
- **Fade-in transition** is a special effect obtained by a simple additive combination of the original image and its pencil sketch image (Figure 3(c)). To obtain this effect, please use the following:

$$I^{'}(i,j,k) = (0.5+\alpha) \times I(i,j,k) + (0.5-\alpha) \times G(i,j,k), \tag{2}$$

  where $I(i,j,k)$ is the value of pixel at (i,j) and channel k for the pencil-sketch image, $G(i,j,k)$ is the value of the original input image at pixel (i,j) and channel k, and α is a parameter in the range $(-0.5, 0.5)$. Please increase α monotonically in this range to generate your own fade-in transition effect for all three test images.

Please briefly discuss the challenges and your observations on these two transition special effects in your report.

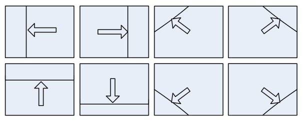


Figure 4: Sweeping directions.

Hint: For Problem 1(d), video clips can be created in Matlab. You could use getframe() and VideoWriter() Matlab functions to create and save the video files. More hints for video clip creation will be provided during discussion sessions.

1(e) Color Pencil Sketch (Bonus: 10%)

Please generalize the pencil sketch technique to color images. One possible generalization is to apply the pencil sketch algorithm to the R, G, B channels, respectively, and then combine their results together. Please try this method and comment on its shortcomings. Please propose another method to achieve the color pencil sketch effect with better visual quality. Please show the results and comment on the reasons of the improvement.

Problem 2: Morphological Processing (35 %)

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.

2(a) Shrinking (Basic: 7%)

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

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

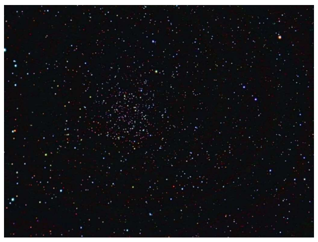


Figure 5: Image: stars.raw

2(b) Thinning (Basic: 7%)

Please apply the 'thinning' filter to the digits image (digits.raw). Please count the total number of digits (0, 1, ..., 9) in the image?

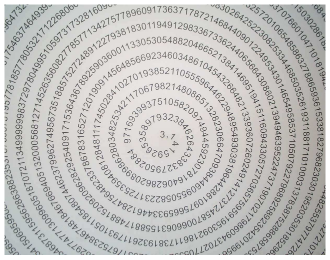


Figure 6: Image: digits.raw

2(c) Skeletonizing (Basic: 8%)

Please apply the 'skeletonizing' filter to the dinosaur image (dinosaur.raw). Please discuss any pre and post processing techniques (if needed) used to solve for this problem.

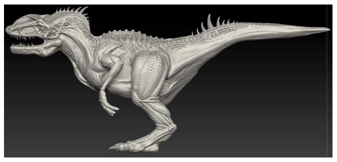


Figure 7: Image: dinosaur.raw

2(d) Pacman Game (Advanced: 13%)

In this part, you should answer the following questions by applying the morphological operators to 'pacman.raw' and describing your methods in detail.

- How many "point-balls" are present in the 'pacman.raw' image? The point-ball as shown in Figure 9 helps increase your score in the game.
- How many "walls" are present in the 'pacman.raw' image? A wall is the structure which the pacman cannot cross. An example of 'wall' is shown in Figure 9.
- How many "turns" can the pacman make in this given image? A turn is done at the corner of the wall. An example of turn is illustrated in Figure 9.

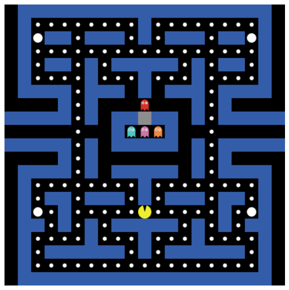


Figure 8: Image: pacman.raw

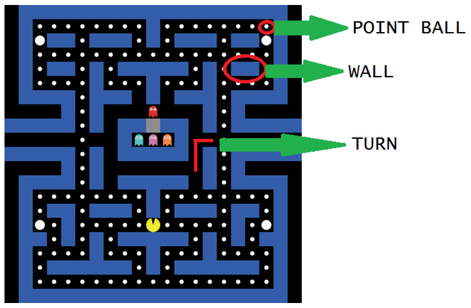


Figure 9: Image: pacman-annotate.raw

Hint: Hints will be given during discussion sessions

Problem 3: Digital Half-toning (40%)

3(a) Dithering (Basic: 13%)

Please implement the following three procedures to convert man.raw to a half-toned image. There are 256 gray levels for pixels in Figure 3 (man.raw). 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 from these three algorithms in your report.

Figure 10: Image: man.raw

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 $0 \sim 255$, 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 build-in rand function in c/c++/Matlab generate numbers in uniform distribution. Please choose two random generators (based on different distribution function), specify what are your choices, compare and discuss the difference in the dither results to justify your choice.
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 [1]. 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:

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

where $G(i,j)$ is the normalized output binary image. Please create 2×2 , 4×4 thresholding matrices and apply them to halftone the man.raw image.

3(b) Error Diffusion (Basic: 12%)

Please apply the error diffusion technique to “gradient.raw”. Show the output images for the three techniques and discuss your results. Also, suggest your own approach to get better results (There is no need to implement your proposed technique. It is fine to describe your solution with justification.)

Figure 11: Image: gradient.raw

1. Floyd-Steinberg error diffusion with serpentine scanning [2].
   Error diffusion matrix for Floyd-Steinberg is

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

2. Jarvis, Judice and Ninke Error diffusion (JJN) [3].

Error diffusion matrix for JJN 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. Stucki Error Diffusion [4].
   Error diffusion matrix for Stucki is

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

Compare the dithering matrix and the error diffusion techniques. Which do you prefer and why?

3(c) Inverse Half-toning (Advanced: 15%)

The process of reconstructing an 8-bit image from its halftoned version is called as “inverse halftoning”. In this part, you will implement one of the classic inverse half-toning techniques described in [5]. Please implement the ‘cascaded inverse-halftone’ algorithm described in Figure 6 of [5]. Please perform ‘inverse-halftoning’ on the half-tone outputs of Dithering matrix and Floyd-Steinberg error diffusion methods. Please show the outputs and discuss your observed results.

Appendix:

Problem 1: Special Effect Image Filters

naruto.raw	$256 \times 256 \times 3$, 8-bit depth, interleaved RGB image
harrypotter.raw	$256 \times 256 \times 3$, 8-bit depth, interleaved RGB image
grainy.raw	$256 \times 256 \times 3$, 8-bit depth, interleaved RGB image

Problem 2: Morphological Processing

stars.raw	$1024 \times 768 \times 3$, 8-bit depth, inteleaved RGB image
digits.raw	$1231 \times 959 \times 3$, 8-bit depth, inteleaved RGB image
dinosaur.raw	$960 \times 444 \times 3$, 8-bit depth, inteleaved RGB image
pacman.raw	$441 \times 441 \times 3$, 8-bit depth, inteleaved RGB image

Problem 3: Digital Half-toning

man.raw	1024×1024, 8-bit depth, grayscale image
gradient.raw	256×256, 8-bit depth, grayscale image

Reference Images

All images used in this homework were downloaded either from Google image search [9] or USC-SIPI image database [8].
*http://harrypotter.wikia.com/wiki/File:Harry-Potter-1-.jpg
**http://desktopvideo.about.com/od/desktopediting/ig/Movie-Maker-Video-Transitions/Fade.htm
***http://ccl.northwestern.edu/netlogo/models/Pac-Man

References

[1] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," *SPIE MILESTONE SERIES MS*, vol. 154, pp. 139–143, 1999.

[2] R. W. Floyd, "An adaptive algorithm for spatial gray-scale," in *Proc. Soc. Inf. Disp.*, vol. 17, 1976, pp. 75–77.

[3] J. F. Jarvis, C. N. Judice, and W. Ninke, "A survey of techniques for the display of continuous tone pictures on bilevel displays," *Computer Graphics and Image Processing*, vol. 5, no. 1, pp. 13–40, 1976.

[4] P. Stucki, *MECCA-A Multiple-Error Correction Computation Algorithm for Bi-Level Image Hardcopy Reproduction.* IBM Thomas J. Watson Research Center, 1991.

[5] C. M. Miceli and K. J. Parker, "Inverse halftoning," *Journal of Electronic Imaging*, vol. 1, no. 2, pp. 143–151, 1992.

[6] Z. Karni, D. Freedman, and D. Shaked, "Fast inverse halftoning," in *31st International Congress on Imaging Science (ICIS 2010), Beijing, China*, 2010.

[7] Z. Xiong, M. T. Orchard, and K. Ramchandran, "Inverse halftoning using wavelets," *Image Processing, IEEE Transactions on*, vol. 8, no. 10, pp. 1479–1483, 1999.

[8] http://sipi.usc.edu/database/

[9] https://www.google.com/imghp