HW2: Histogram and Spatial Filtering

DIP Teaching Stuff, *Sun Yat-sen University*

Welcome to your second DIP homework! Histogram and Filtering are two of the **core** components of this course (the other core component is Fourier Transform in Chapter 4), hence you need to pay more attentions to this homework. All right, we know you can finish it well! Similar to your first homework, the second homework is composed of several simple questions and certain programming works. Please submit a report (in **PDF** format) and all relevant codes as the homework solution. Warning: We encourage discussions among students, but homework solutions should be written and submitted **individually**, without copying existed answers. Plagiarism = Fail. Besides, there may be at least 30% penalty for late homework.

1 Exercises

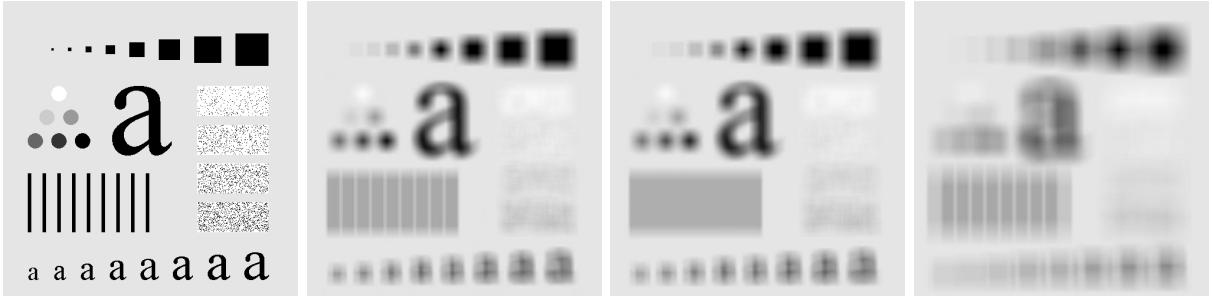
Please answer the following questions in the report.

1.1 Histogram Equalization (10 Points)

Suppose that you have performed histogram equalization on a digital image. Will a second pass of histogram equalization (on the histogram-equalized image) produce exactly the same result as the first pass? Prove your answer.

1.2 Spatial Filtering (10 Points)

The three images shown in Fig. 1(b)∼(d) were blurred using square averaging filters of sizes $n = 23, 25, 45$ respectively. The original image is shown in Fig. 1(a). The vertical bars on the left lower part of Fig. 1(a) are 5 pixels wide, 100 pixels high, and separated by 20 pixels. We can see that though the vertical bars in Fig. 1(b) and Fig. 1(d) are blurred, a clear separation still exists between them. However, the bars have merged in Fig. 1(c), in spite of the fact that the filter that produced this image is significantly smaller than the filter that produced Fig. 1(d). Explain the reason for this.



(a) (b) (c) (d)

Figure 1: *Spatial Filtering.*

2 Programming Tasks

Write programs to finish the following three tasks, and answer questions in your report. Don't forget to submit all relevant codes.

2.1 Pre-requirement

Input Please download the archive "hw2.zip", unzip it and choose the image corresponding to the last two digits of your student ID. This image is the initial input of your programming tasks in HW2. For example, if your student ID is "12110349", then you should take "49.png" as your input. You can convert the image format (to BMP, JPEG, ...) via Photoshop if necessary.

Make sure that you have selected the correct image. Misusing images may result in zero scores.

Language Any language is allowed.

Others There remain some issues that you should pay attention to:

1. You can use third-party packages for operating images. But you should manually implement your programming tasks. For example, though you can use "imread" of Matlab to load an image, you cannot invoke "conv2" or "filter2" or so forth of Matlab for spatial filtering.
2. Good UX (User Experience) is encouraged, but will only bring you negligible bonuses. Please don't spend too much time on it, since this is not an HCI course.
3. Keep your codes clean and well-documented. Bad coding styles will result in 20% penalty at most.

2.2 Histogram Equalization (35 Points)

1. Write a function named "**plot_hist**" for computing and displaying the histogram of a gray scale image.
2. Write a function that applies histogram equalization on a gray scale image. The function prototype is "**equalize_hist(input_img)** $\rightarrow$ **output_img**", returning a gray scale image whose histogram is approximately flat. You can modify the prototype if necessary.

For the report, please load your input image and use your program to:

1. Compute and display its histogram. Manually paste the histogram on your report. (5 Points)
2. Equalize its histogram. Paste the histogram-equalized result and the corresponding histogram on your report. (10 Points)
3. Analyze your histogram-equalized result in **less** than 1 page. (8 Points)
4. Detailedly discuss how you implement the histogram equalization operation, i.e., the "equalize_hist" function, in **less** than 2 pages. Please focus on the algorithm part. Don't widely copy/paste your codes in the report, since your codes are also submitted. (12 Points)

2.3 Image Patch Extraction (10 Points)

Write a function that extracts all patches of a gray scale image. Here we define a "patch" as an overlapping window view of an image, with adjacent patches shifted by a single row or column. The function prototype is "**view_as_window(img, patch_size) → arr_patch**", where "img" is an input image, "patch_size" is a tuple indicating the width and the height of a patch, and "arr_patch" is a list of 2-D patches (a better way for storage is using a 3-D array instead of a list, but the choice is yours). Modify the prototype if necessary.

For example, suppose you are given an image:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

Let the patch size be 2×2. At this time the returned patches would be:

$$\begin{bmatrix} 0 & 1 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \begin{bmatrix} 4 & 5 \\ 8 & 9 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 9 & 10 \end{bmatrix} \begin{bmatrix} 6 & 7 \\ 10 & 11 \end{bmatrix} \begin{bmatrix} 8 & 9 \\ 12 & 13 \end{bmatrix} \begin{bmatrix} 9 & 10 \\ 13 & 14 \end{bmatrix} \begin{bmatrix} 10 & 11 \\ 14 & 15 \end{bmatrix}$$

If the patch size is 3×3, then 4 patches would be returned:

$$\begin{bmatrix} 0 & 1 & 2 \\ 4 & 5 & 6 \\ 8 & 9 & 10 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 10 & 11 \end{bmatrix} \begin{bmatrix} 4 & 5 & 6 \\ 8 & 9 & 10 \\ 12 & 13 & 14 \end{bmatrix} \begin{bmatrix} 5 & 6 & 7 \\ 9 & 10 & 11 \\ 13 & 14 & 15 \end{bmatrix}$$

In a conclusion, given a $W \times H$ image, you can extract $(W - w + 1) \times (H - h + 1)$ patches from it, with the size of each patch being $w \times h$.

For the report, please load your input image and use your "view_as_window" function to:

1. Extract all 96×64 (Width: 96, Height: 64) patches. Randomly paste 8 patches on your report. (5 Points)

2. Extract all 50×50 patches and randomly paste 8 of them. (5 Points)

2.4 Spatial Filtering (35 Points)

Write a function that performs spatial filtering on a gray scale image. The function prototype is "**filter2d(input_img, filter) → output_img**", where "filter" is the given filter. Modify the prototype if necessary. Note: It should be easy to implement "filter2d" based on "view_as_window", since spatial filtering is almost equivalent to the dot product of each patch and the filter (Why? Think about it.), though you have to take care of image boundaries. Anyway, if you wanted, you can implement "filter2d" in the way you like.

For the report, please load your input image and use your "filter2d" function to:

1. Smooth your input image with 3×3, 7×7 and 11×11 averaging filters respectively. Paste your three results on the report. (9 Points)

2. Sharpen your input image with a 3×3 Laplacian filter and then paste the result (There are four variants of Laplacian in Fig. 3.37 of the textbook. Pick any one you like). (3 Points)

3. Filter your image with two 3×3 Sobel filters (See Fig. 3.41(d)$\sim$(e) of the textbook) and then paste the two results. (6 Points)

4. Describe some applications of different filters based on your own knowledge in **less** than 2 page. (7 Points)

5. Detailedly discuss how you implement the spatial filtering operation, i.e., the "filter2d" function, in **less** than 2 pages. (10 Points)