Digital Image Processing
School of Information Science and Technology
ShanghaiTech University

Assignment 1

An Lihua, 2022131001

Due time: 23:59, March 15th, 2023

---

1 Notes

This homework has **100 points** in total.

Please submit your homework to blackboard with a zip file named as **DIP2023_ID_Name_hw1.zip**. The zip file should contain three things: **a folder named 'codes' storing your codes**, **a folder named 'images' storing the original images**, and **your report named as report_ID_Name_hw1.pdf**. The names of your codes should look like **'p1a.m'** (for (a) part of Problem 1), so that we can easily match your answer to the question. Make sure all paths in your codes are relative path and we can get the result directly after running the code. Please answer in English.

Please complete all the coding assignments using MATLAB. All core codes are required to be implemented by yourself (without using relevant built-in functions). Make sure your results in the report are the same as the results of your codes. Please explain in notes at least the key steps in your code.

2 Policy on plagiarism

This is an individual homework. You can discuss the ideas and algorithms, but you can neither read, modify, and submit the codes of other students, nor allow other students to read, modify, and submit your codes. Do not directly copy ready-made or automatically generated codes, or your score will be seriously affected. We will check plagiarism using automated tools and any violations will result in a zero score for this assignment.

3 Problem sets

Problem 1 (30 pts)

$$\begin{bmatrix} 6 & 1 & 2 & 1 & (2) \\ 2 & 3 & 5 & 3 & 8 \\ 1 & 0 & 1 & 2 & 3 \\ 3 & 2 & 4 & 5 & 2 \\ (1) & 5 & 3 & 4 & 0 \end{bmatrix}$$

(a) Calculate the D_4, D_8, D_m distance between the pair of points(framed in parentheses) in the matrix above, mark the shortest 4-,8-,m-path respectively. Show the results in your report. (V=1,2,3) (10 pts)

(b) An affine transformation of coordinates is given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Please perform the following transformation on **jetplane.tif**, write out the affine transformation matrix, then show the results:

(1) Scaling and shearing. Set x-scaling factor to 2 and y-scaling factor to 3, then shear the image horizontally 4 units. (5 pts)

(2) Translation and Rotation. Move 2 units to the left and move down 5 units, then rotate 45 degrees clockwise (about the origin). (5 pts)

(Hint: The main focus of this problem is to solve the affine transformation matrix, and using built-in functions like imwarp() to obtain the transformed image is allowed.)

(c) Perform bit-plane slicing on **lena_gray_256.tif** to get bit plane 1-8 and show the results in your report. Which kind of bit-plane usually contains more effective information, low bit or high bit? Why? (10 pts)

Solution:

(a) The position of two points are (5,1) and (1,5).

D_4:

$$D_4 = |5 - 1| + |1 - 5| = 8$$

D_8:

$$D_8 = max(|5 - 1|, |1 - 5|) = 4$$

D_m: Fig. 1 shows all the m-adjacency, $D_m = 8$ when choose path 1 and $D_m = 7$ when choose path 2, so the shortest m-path is path 2.

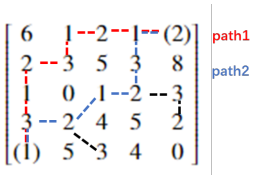

Figure 1: All the m-adjacency.

According to the Fig. 2, I have marked the 8-adjacency, and it is easy to find the red path 1 is the shortest 4-path and the blue path 2 is the shortest 8-path.

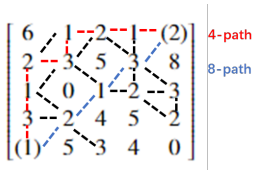

Figure 2: All the 8-adjacency.

(b) (1) For x-scaling we need to set $a_{11} = 2$, and for y-scaling set $a_{22} = 3$, then for the image shearing horizontally we need to set $a_{21} = 4$, then we get the transformation matrix:

$$A_{scaling} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A_{shear} = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The result is shown in Fig. 3:

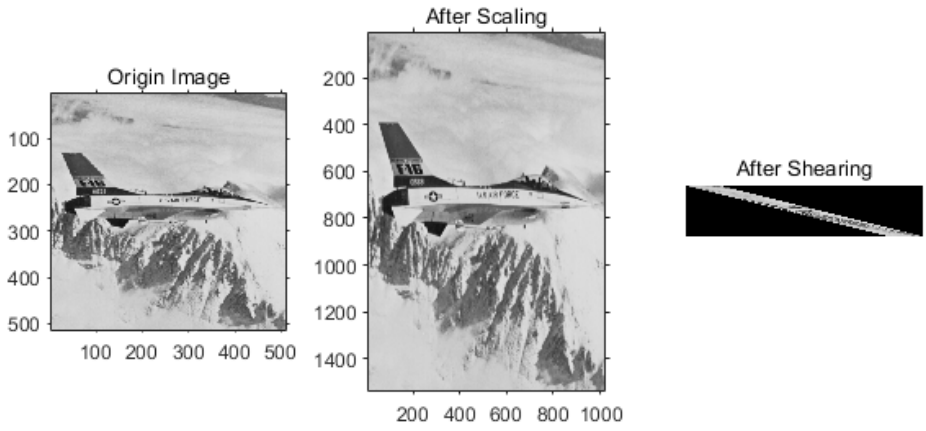

Figure 3

(2) For moving 2 units to the left we need to set $a_{23} = -2$, and for 5 units down set $a_{13} = 5$, then for the 45 degrees rotation we need to set $a_{11} = a_{12} = a_{22} = \frac{\sqrt{2}}{2}$ and $a_{21} = -\frac{\sqrt{2}}{2}$, then we get the transformation matrix:

$$A_{trans} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A_{rot} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The result is shown in Fig. 4:

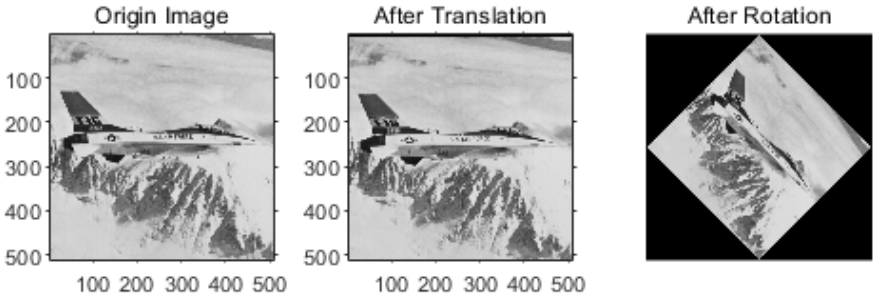

Figure 4

(c) Fig. 5 shows the bit-plane slicing results and Fig. 6 is the code of BitPlaneSlicing function in MATLAB. In this function, I translate the image from decimal to 8 bit binary, then reshape it to a MxNx8 array, then decide every bit whether it is equal to '1'. If it is '1', set the output to 1, otherwise set it to 0.

From Fig. 5, we can easily get that the bit 7 and 8 (**high bit**) contains more effective information, since higher bits have longer intervals and contain more elements, thus carrying more effective information.

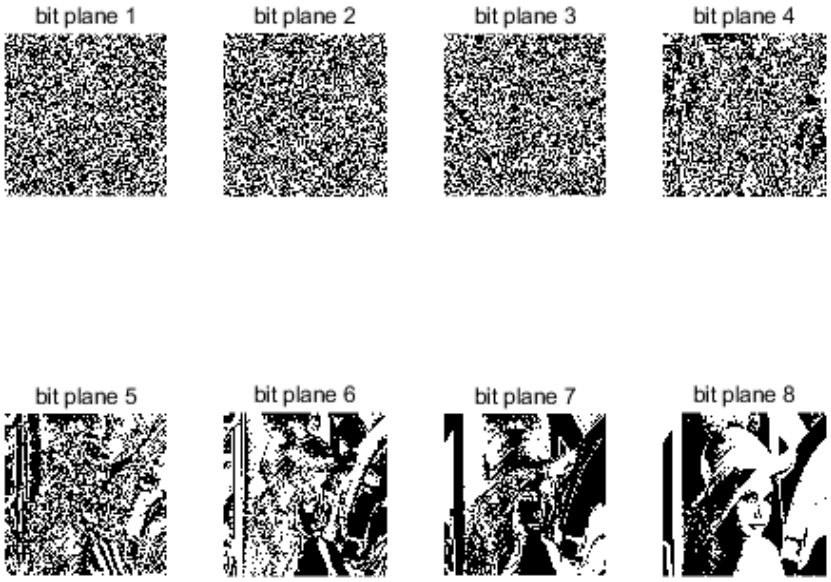

Figure 5: Bit plane slicing results.

Function: Bit Plane Slicing

```
36  function output = BitPlaneSlicing(input)
37      input = double(input);
38      [row,col] = size(input);
39      output = zeros(row,col,8);
40
41      input_bin = dec2bin(input,8);
42      input_bin = reshape(input_bin(:),row,col,8);
43      for i = 1:row
44          for j = 1:col
45              for k = 1:8
46                  if input_bin(i,j,9-k) == '0'
47                      output(i,j,k) = 0;
48                  else
49                      output(i,j,k) = 1;
50                  end
51              end
52          end
53      end
54  end
```

Figure 6: BitPlaneSlicing code.

Problem 2 (30 pts)

(a) Compute the histogram of **einstein_low_contrast.tif**. Perform histogram equalization (HE) on it, show the result. Show the histogram of the processed image, too. How can HE enhance the contrast? (10 pts)

(b) Match the histgram of **lena_color.tif** to **peppers_color.tif**. Show the result in your report. (8 pts)

(c) Perform contrast limited adptive histogram equalization(CLAHE) on **man_in_house.png**. Contrast the result to the HE method. Show the results in your report. Explain why CLAHE has a better effect.(Reference: http://cas.xav.free.fr/Graphics%20Gems%204%20-%20Paul%20S.%20Heckbert.pdf, page474-485) (12 pts)

(Hint: If you can effectively eliminate the checkerboard effect and balance the efficiency of the algorithm, you will get a bonus. Built-in functions like hist(), histogram(), histeq() and adapthisteq() are not allowed.)

Solution:

(a) Fig. 7 shows the results of histogram and equalization, and Fig. 8 is the transformation function $s_k = T(r_k) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$ (k = 0, 1, 2, ..., L-1). Fig. 9 shows the key functions of p2a, (b) is **my_hist()** function, which realize the same function as imhist() in MATLAB, (a) is the histogram equalization function by realizing the transformation equation talked before.

Histogram equalization makes the number of pixel values in a certain gray range approximately equal by redistributing image pixel values. In this way, the contrast of the peak part in the middle of the original histogram is enhanced, while the contrast of the valley part on both sides is reduced, and the histogram of the output image is more uniform. For an image, when the distribution of each gray value is more balanced, the amount of information contained in the image is greater.

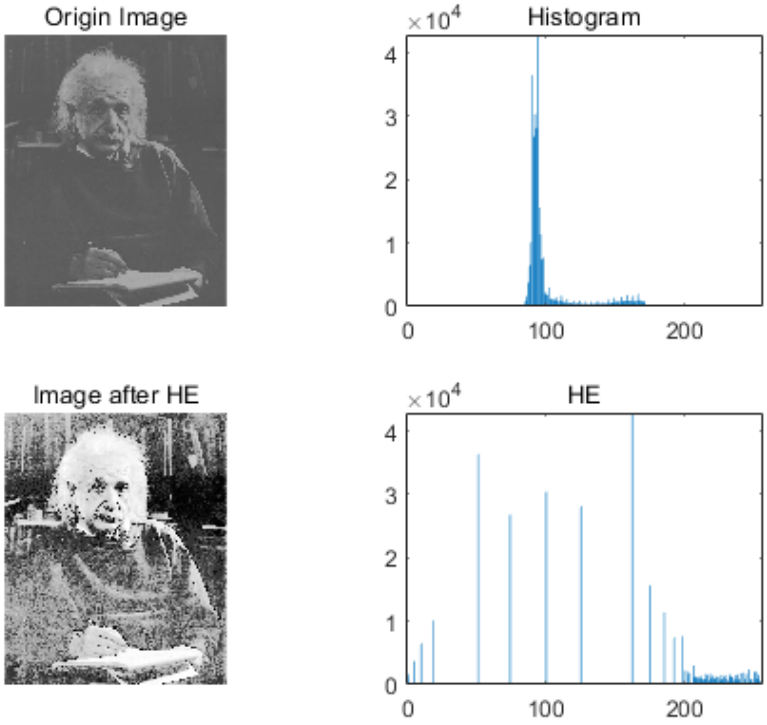

Figure 7: Histogram and histogram equalization.

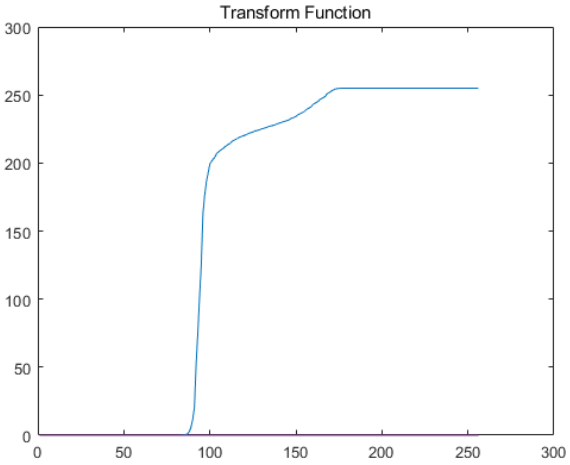

Figure 8: Transform function.

```matlab
% HE function
T = zeros(1,256);
for k = 1:256
    if k == 1
        T(k) = 255*I_hist(k)/M/N;
    else
        T(k) = T(k-1) + 255*I_hist(k)/M/N;
    end
end
```

(a)

```matlab
function output = my_hist(input)
[M,N] = size(input);
output = zeros(1,256);

for k = 1:256
    for m = 1:M
        for n = 1:N
            output(k) = output(k) + (input(m,n) == k);
        end
    end
end
end
```

(b)

Figure 9: Key functions of p2a. (a) histogram equalization, (b) show histogram.

(b) Fig. 10 shows the matching results. To match the whole image, we need to do matching for every channel, which are R, G, B channels respectively.

The main idea is shown in Fig. 11. Based on the functions in q2a, it needs to find the minimum distance between source histogram and destined histogram, then set it to the source histogram to realize matching.

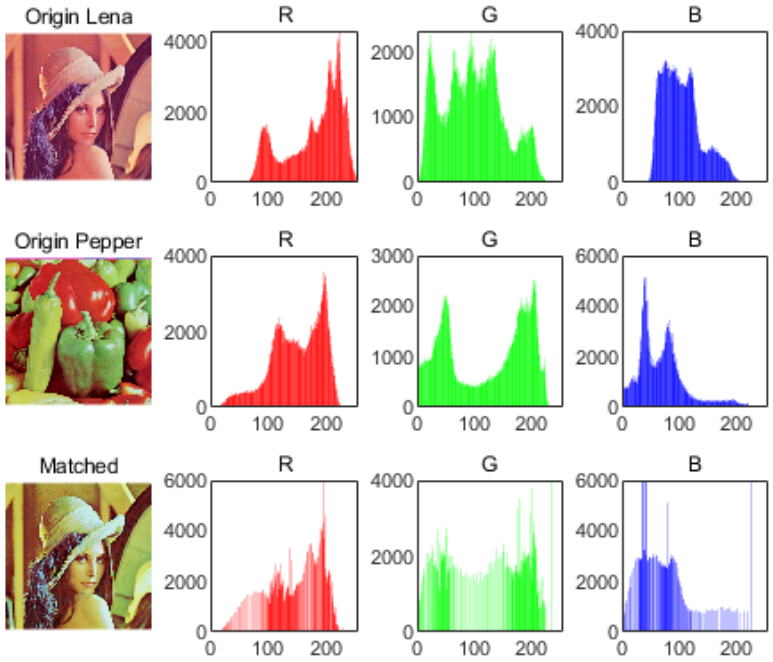

Figure 10: Histogram matching.

```
% finding minimum points
idx1 = zeros(1,256);
idx2 = zeros(1,256);
idx3 = zeros(1,256);
for i=1:256
    [min1,idx1(i)]=min(abs(T_P1-T_L1(i)));
    [min2,idx2(i)]=min(abs(T_P2-T_L2(i)));
    [min3,idx3(i)]=min(abs(T_P3-T_L3(i)));
end

% matching
match1=zeros(m,n);
match2=zeros(m,n);
match3=zeros(m,n);
for i=1:m
   for j=1:n
      match1(i,j)=idx1(L1(i,j)+1)-1;
      match2(i,j)=idx2(L2(i,j)+1)-1;
      match3(i,j)=idx3(L3(i,j)+1)-1;
   end
end
```

Figure 11: Key function for histogram matching.

(c) Histogram equalization (HE) has two problems:

1. Histogram equalization is global, and the effect is not very good when there is too bright or too dark in local areas of the image;

2. Histogram equalization will enhance the background noise

In order to solve the above problems, a method of image segmentation is proposed. Each region is subject to histogram equalization separately, so that local information can be used to enhance the image, which can solve the global problem; Aiming at the problem of background noise enhancement, a method to limit the contrast is proposed. The combination of the above two is CLAHE method.

The results are shown in Fig. 12. For CLAHE, the slicing window is [12,16] and the clip limit is set to 0.1. The function **my_CLAHE()** is reference to the source code of MATLAB.

To clip the histograms, we need to trim the histogram in the tiles to make its amplitude lower than a certain upper limit. At the same time, we need to distribute this part of the clipping value on the entire gray range to ensure that the total area of the histogram remains unchanged. Then match the new histogram to the source histogram.

Set the clipping value as **ClipLimit**, and calculate the sum of excesses of the part higher than this value in the histogram. Then the excesses are divided into all gray levels uniformly, and calculate the height of the overall rise of the histogram caused by this L=excess/N, and take upper=ClipLimit-L as the boundary to process the histogram as follows:

(1) If the amplitude is higher than ClipLimit, it is directly set to ClipLimit;

(2) If the amplitude is between Upper and ClipLimit, fill it to ClipLimit;

(3) If the amplitude is lower than Upper, directly fill L pixels;

After the above operation, the number of pixel points to be filled is usually slightly less than the total excess, that is, there are still some remaining pixel points that have not been divided out, and the rest comes from (1) (2). At this time, we can distribute these points evenly to those gray values whose current amplitude is still less than ClipLimit.

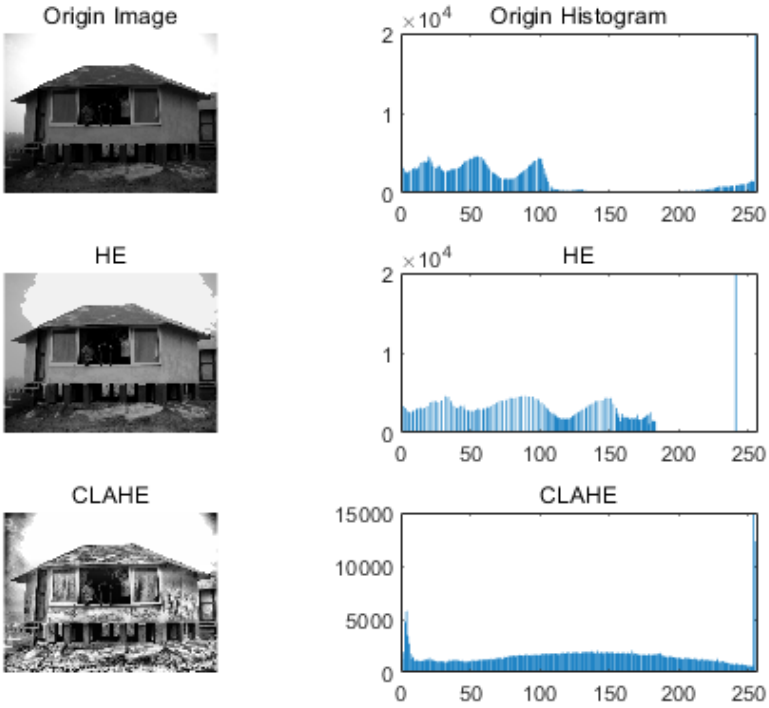

Figure 12: CLAHE.

Problem 3 (40 pts)

(a) Implement full convolution and cropped convolution of the given matrices, show the results in your report. (8 pts)

$$\begin{bmatrix} 6 & 4 & -1 & 0 & 1 \\ 1 & -3 & -4 & 3 & 2 \\ 0 & 3 & 5 & -2 & 1 \\ 9 & -1 & -3 & 4 & 5 \\ -2 & -5 & 2 & 3 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 4 \\ -3 & -1 & 1 \\ 5 & 2 & -1 \end{bmatrix}$$

(Hint: Pay attention to the difference between convolution and correlation.)

(b) Describe the noise type on **circuitboard-a.tif**, **circuitboard-b.tif**, **circuitboard-c.tif** and **circuitboard-d.tif**, then choose the best filter you think for each of them to reduce the noise. Show the results in your report. (12 pts)

(Hint: Choose the best kernel size you think.)

(c) Filter the image **house.tif** in both x direction and y direction with 3*3 Sobel mask and Laplacian mask. Show the results in your report. (8 pts)

(d) Perform image sharpening on **house.tif** using LoG filter(with the $\sigma^2 = 1$ and another two values you choose yourself) and unsharpen mask method(choose the best k you think). Show the results in your report. (12 pts)

Solution:

(a) Full convolution:

$$\begin{bmatrix} 6 & 16 & 31 & 14 & -3 & 2 & 4 \\ -17 & -19 & -1 & -12 & -12 & 15 & 9 \\ 27 & 43 & 24 & 6 & 10 & -3 & 5 \\ 14 & -5 & -14 & 8 & 25 & 24 & 19 \\ -29 & 0 & 34 & -26 & -12 & 15 & 4 \\ 51 & 30 & -29 & -1 & 35 & 9 & -5 \\ -10 & -29 & 2 & 24 & 4 & -3 & 0 \end{bmatrix}$$

Cropped convolution:

(1) With padding

$$\begin{bmatrix} -19 & -1 & -12 & -12 & 15 \\ 43 & 24 & 6 & 10 & -3 \\ -5 & -14 & 8 & 25 & 24 \\ 0 & 34 & -26 & -12 & 15 \\ 30 & -29 & -1 & 35 & 9 \end{bmatrix}$$

(2) Without padding

$$\begin{bmatrix} 24 & 6 & 10 \\ -14 & 8 & 25 \\ 34 & -26 & -12 \end{bmatrix}$$

(b) **circuitboard-a.tif** is pepper-noise, use a 5x5 median filter.

circuitboard-b.tif is salt-noise, use a 5x5 median filter.

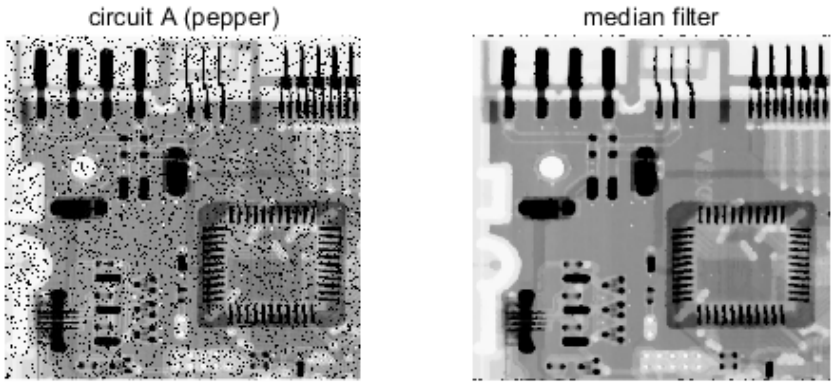

Figure 13: Circuit A.

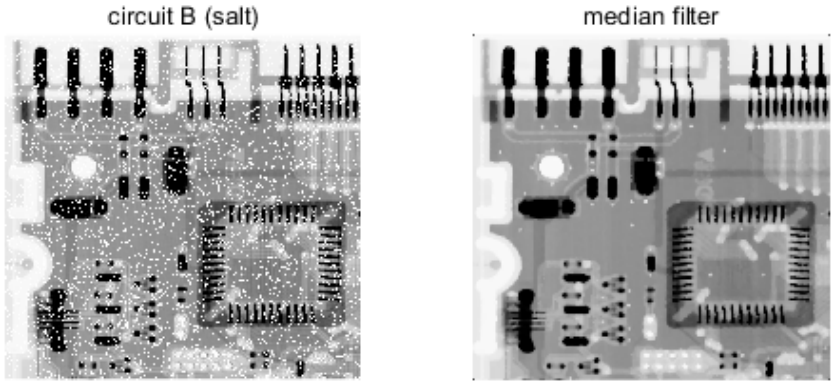

Figure 14: Circuit B.

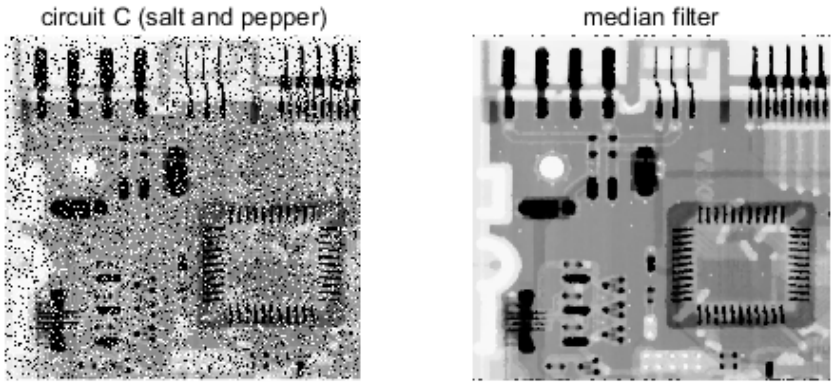

Figure 15: Circuit C.

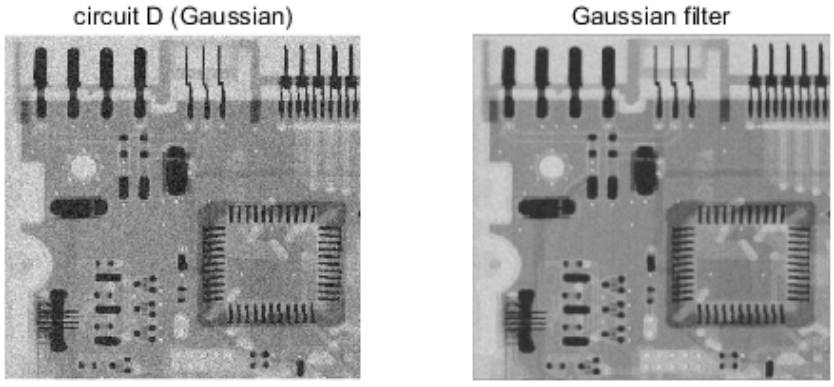

Figure 16: Circuit D.

circuitboard-c.tif is salt-pepper noise, use a 5x5 median filter.

circuitboard-d.tif is Gaussian noise, use a 5x5 Gaussian filter.

Fig. 17 is the key functions. (a) is the median filter, in this function we use the bubble sort algorithm in (b) to find the median value. (c) is the Gaussian filter.

```
function output=my_median_filter(I,kernal)
[M,N] = size(I);
output = ones(M,N);
I_tmp = zeros(M+kernal-1,N+kernal-1);
I_tmp(1+(kernal-1)/2:M+(kernal-1)/2,1+(kernal-1)/2:N+(kernal-1)/2) = I;

for i = 1:M
    for j = 1:N
        tmp = I_tmp(i:(i+kernal-1), j:(j+kernal-1));
        tmp = tmp(:);
        output(i,j) = med_sort(tmp);
    end
end
end
```

(a)

```
%=========================
% bubble sort
function medvalue=med_sort(buff)
V = length(buff);

for m=1:V-1
    for n=1:V-1
        if buff(n)>buff(n+1)
        temp = buff(n);
        buff(n) = buff(n+1);
        buff(n+1) = temp ;
        end
    end
end
medvalue = buff((V+1)/2);
end
```

(b)

```
%=========================
% Gaussian Filter
function output = Gaussian(hsize, sigma)
hhsize = (hsize-1)/2;
[x,y] = meshgrid(-hhsize(2):hhsize(2),-hhsize(1):hhsize(1));
output = exp(-(x.*x + y.*y)/(2*sigma*sigma));

cum = sum(output(:));
output = output/cum;
end
```

(c)

Figure 17: Key functions of p3b. (a) histogram equalization, (b) show histogram.

(c) For Sobel mask, we have:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2}$$

For Laplace mask, I use below one to filter both x and y direction:

$$G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The results are shown in Fig. 18.

(d) Fig. 19 shows the results. (b) is the Laplace mask in p3c, we can see there is a lot of noise at the recognized edge, so here introduced Laplace of Gaussian mask.

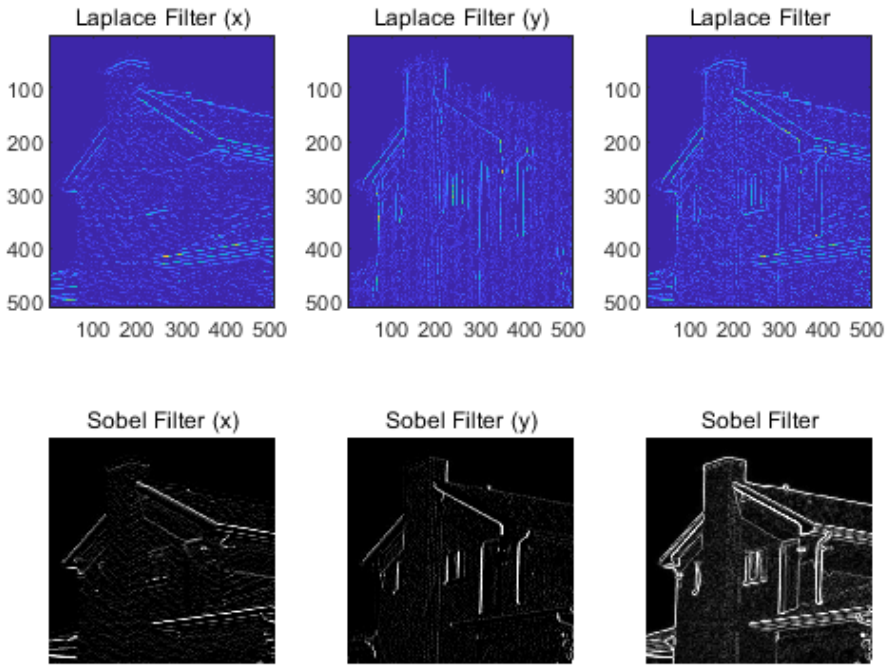

Figure 18: Filter with two different masks.

LOG mask is very sensitive to edge noise. This method combines Gaussian filter and Laplace filter for edge detection. This method first eliminates noise through filtering before edge detection, so the detection effect is very good and the image is very clear after processing. The LOG filter method judges the edge points by detecting the zero-crossing point of the second derivative.

(c) and (d) are the recognized edge using Gaussian filter and second derivative Laplace filter and its result. Compared with the Laplace mask, the noise is significantly reduced, but the edge is still very rough. Then we try the LOG operator template:

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

(e) and (f) are the results using LOG template, it has a much better performance.

For unsharpen mask, firstly use the Gaussian low-pass filter to get the blurred image, then subtract the blurred image from the original image to get the mask, and finally multiply the mask by a positive coefficient k to add to the original image to get the enhanced sharpening result.

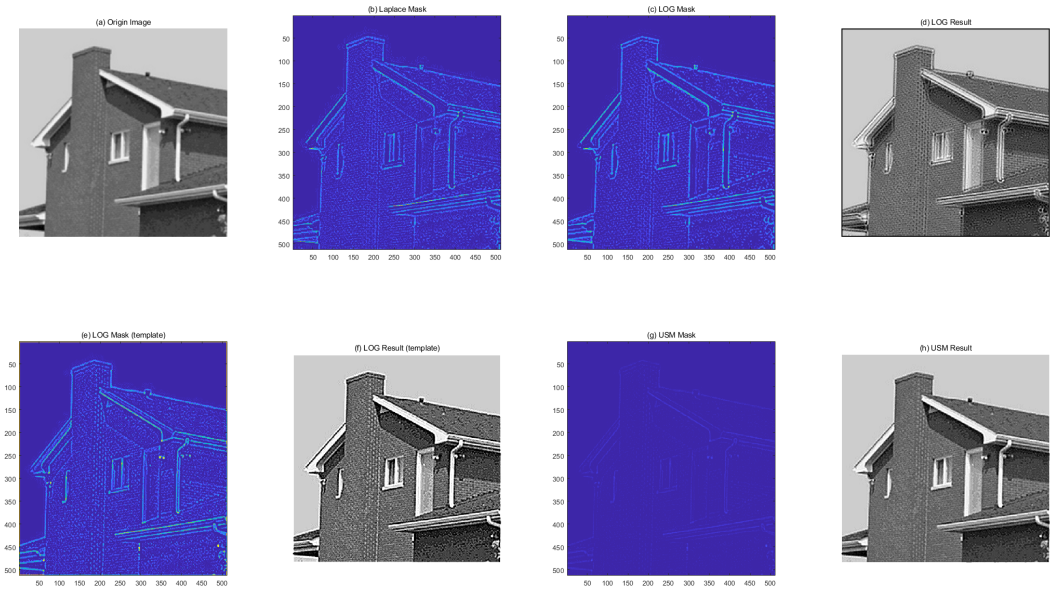

Figure 19: Sharpening the image with two different method.