Digital Image Processing
School of Information Science and Technology
ShanghaiTech University

Assignment 1

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 with the results of your codes. Please explain with notes at least at the key steps of 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:

(1.a) $D_4 = 8$, $D_8 = 4$, $D_m = 7$

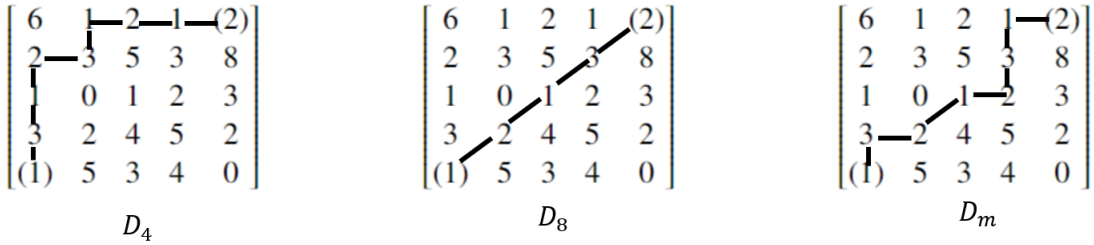


Figure 1: shortest 4-, 8-,m-path respectively

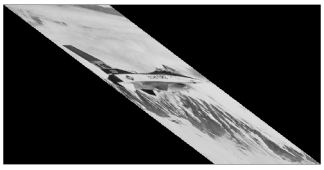
(1.b1) In this case, a horizontal shear would shift pixels vertically based on their y-coordinate.

A horizontal shear by 4 units means that for each pixel at position (x, y) in the original image, we will shift it vertically by 4 times its y-coordinate. So a pixel at position (10, 20) in the original image would be shifted to position (10 + 4*20, 20) = (90, 20) in the transformed image.

The scaling factors of 2 and 3 mean that the image will be stretched along the x-axis by a factor of 2 and along the y-axis by a factor of 3. So a pixel at position (10, 20) in the original image would be mapped to position (2*10 + 4*20, 3*20) = (100, 60) in the transformed image.

The transform matrix is given by

$$\begin{bmatrix} 2 & 4 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



(1.b2)

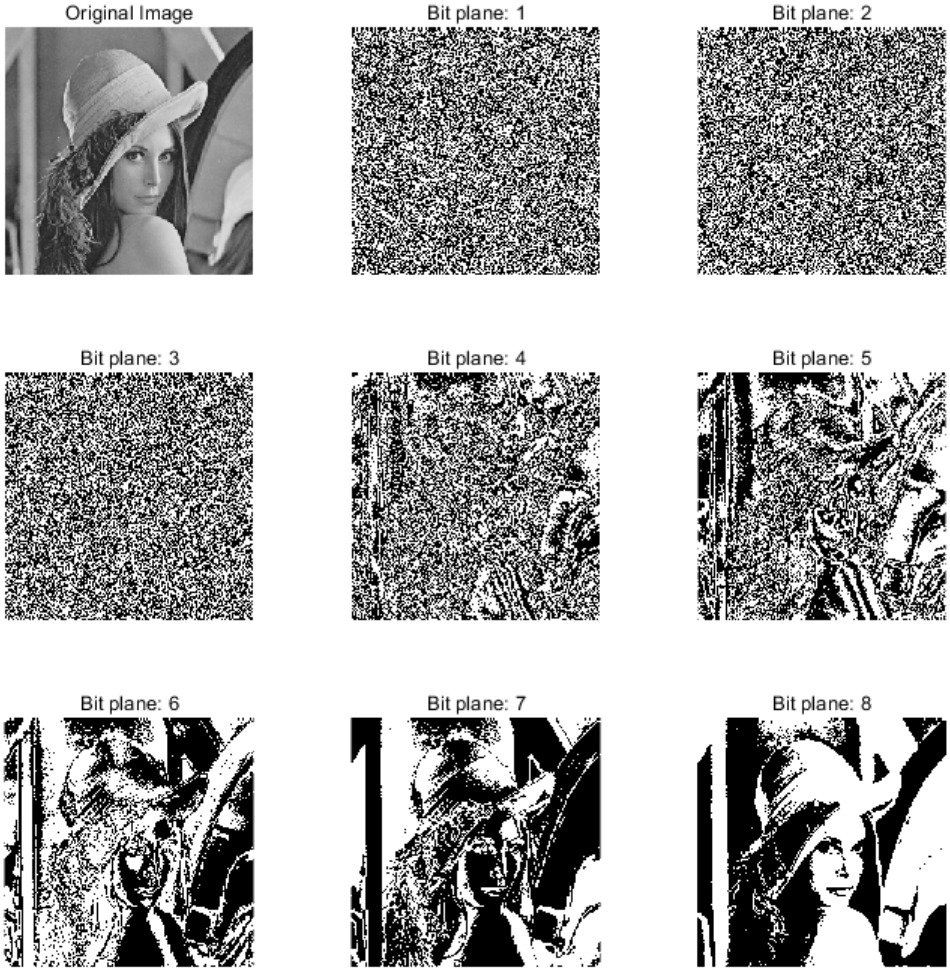
$$\begin{bmatrix} 0.7071 & -0.7071 & 5 \\ 0.7071 & 0.7071 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

where the first row represents the x-axis transformation (which includes a 45 degree clockwise rotation and a translation of 5 units down), the second row represents the y-axis transformation (which includes a 45 degree clockwise rotation and a translation of 2 units to the left), and the third row represents the homogeneous coordinate transformation.

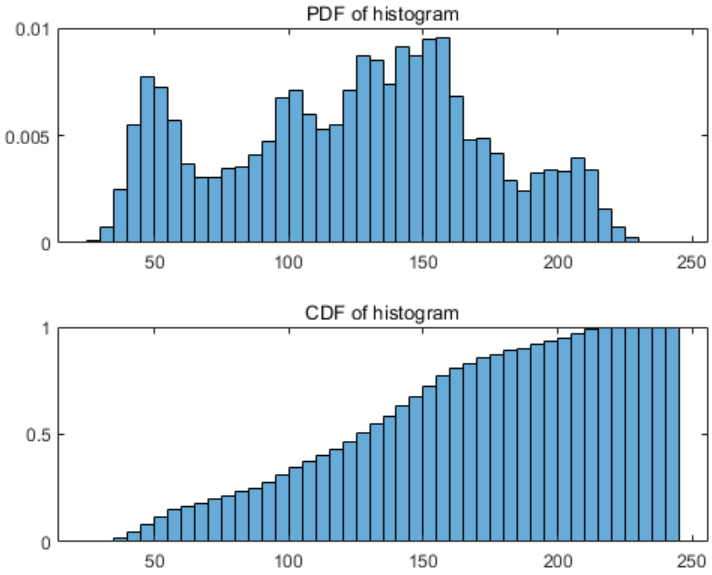
(1.c) The K's bit plane can be given by

$$Bit(k) = I \ \ \& \ \ 2^{k-1}$$

where k ranges from 1 to 8. After performing normalization, the 8 bit planes is shown as follow,



and the bit plane with $k = 7, 8$ contains more effective information. By invesgating the pdf and cdf of historgram of original image, we find that almost half of pixels have the value bigger than 128, and we are always interested at the bright region of original image. Thus, the high bit gives the most reasonable bit plane.


PDF of histogram
0.01
0.005
0
50
100
150
200
250
CDF of histogram
1
0.5
0
50
100
150
200
250

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. Why 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 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:

(2.a) The general mapping function of histogram equalization is given by

$$s = T(r) = (L-1)\int_0^r p_r(w)dw = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j k$$

Histogram equalization works by redistributing the pixel intensities in an image so that they are more uniformly distributed across the entire intensity range. This can enhance the contrast of an image by making the darker areas darker and the brighter areas brighter. By increasing the contrast of an image, details that were previously difficult to see may become more visible.

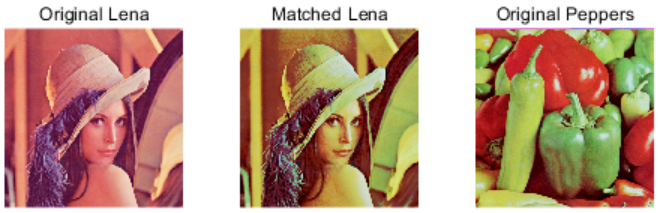
(2.b) The general mapping function of histogram matching is given by

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

$$G(z) = (L-1)\int_0^z p_z(t)dt = s$$

$$z = G^{-1}(s) = G^{-1}[T(r)]$$

And we perform the histogram matching for three channels of color image individually.



(2.c) Although naive histogram equalization is a good way to adjust the distribution of brightness, HE algorithm does not always work well. Considering one case, when there are some large very dark regions or large very bright regions, HE algorithm has no idea to deal with these problems due to cdf math prior. In other words, in this case, HE algorithm tends to make the middle gray regions very sparse and the image becomes very noisy, which can be seen in this problem.

However, contrast limited HE introduces a method to limit the contrast, which is achieved by adding a limit to the histogram. And this method allows new generated image has more flatten histogram.

Adaptive HE divides the original image into sub-grids. And it calculates each region's histogram individually. This method avoids the large dark-bright regions to affect the whole generated image. Also, we implement the interpolation to avoid the checkboard effect. For each pixel, we first locate it at the belonglong sub-grid. Then, we further divide the sub-grid into four regions, and we perform the interpolation with four neighhour sub-grids w.r.t which sub-sub-grid (four in total) the pixel locates in. Importantly, we DONOT interpolate the neighbor pixel values directly, Instead, we DO inference the four neightbor sub-grid's histogram, and we DO interpolate the value coming from the four histograms.

We provide a self-implemented matlab version of CLAHE algorithm. And we compare the effect of our algorithm as bellow:

Figure 2: Original Image

Figure 3: CLAHE: No interpolation for each pixel

Figure 4: CLAHE: With interpolation for each pixel. For each pixel, we first locate it at the belonglong sub-grid. Then, we further divide the sub-grid into four regions, and we perform the interpolation with four neighhour sub-grids w.r.t which sub-sub-grid (four in total) the pixel locates in. Importantly, we DONOT interpolate the neighbor pixel values directly, Instead, we DO inference the four neightbor sub-grid's histogram, and we DO interpolate the value coming from the four histograms.

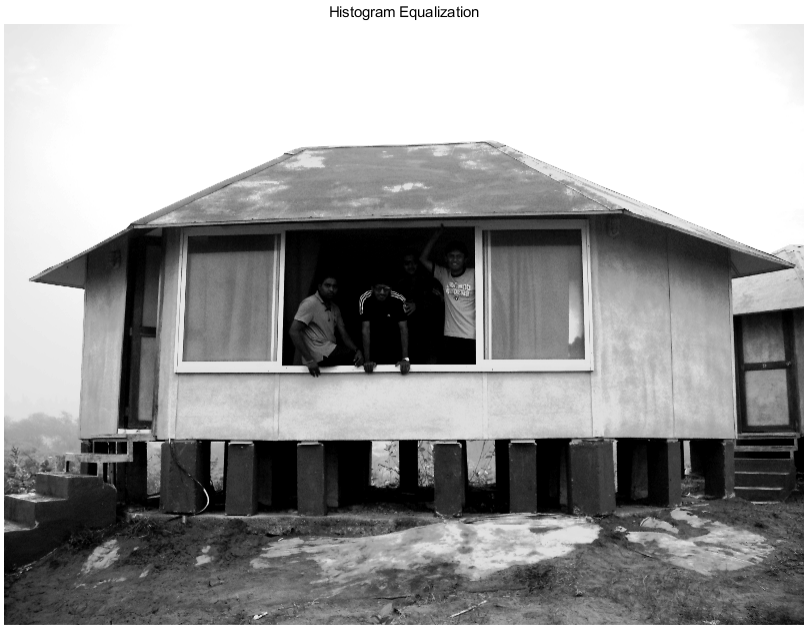


Figure 5: Naive HE

Then, we try to further improve the contrast of the image. And we hope to see the men in house very clearly. Without changing any core part of our CLAHE, we can just change the two parameters, and we find that

numTiles=64, clipLimit=0.08 can give a reasonable clear image without too much checkerboard effect.

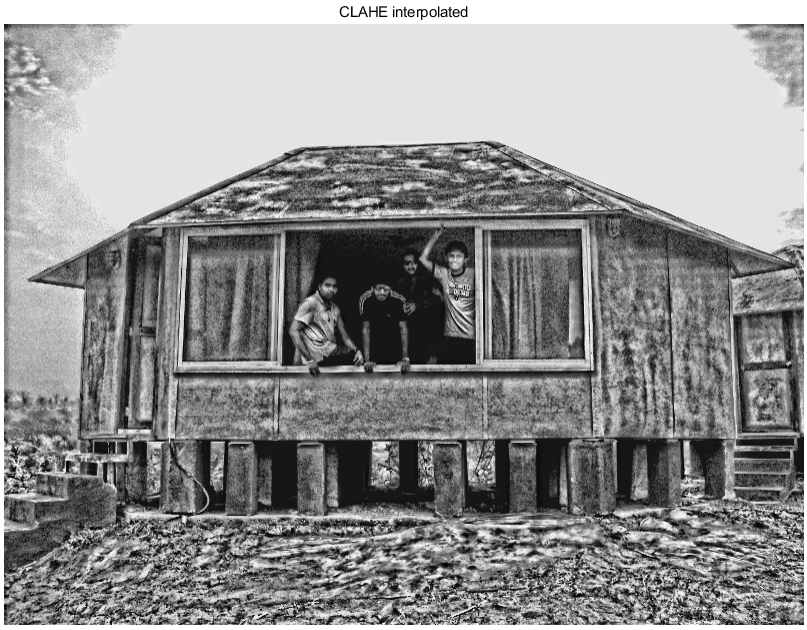


Figure 6: Improved result (CLAHE): we try to further improve the contrast of the image. And we hope to see the men in house very clearly. Without changing any core part of our CLAHE, we can just change the two parameters, and we find that numTiles=64, clipLimit=0.08 can give a reasonable clear image without too much checkerboard effect.

However, there are still several improvements that can be done to improve the final result.

- Firstly, we can introduce **multi-scale tile** to CLAHE. For example, we can assign a large tile to the region with similar grays and samll tile to the intereted low-contrast region. However, this can leads to difficulty to interpolate multi-scale tiles.
- Secondly, we can directly combine the results by **different-tile** CLAHE together. For example, in this case, we can combine the central men-house part resulted with small tile and other uninterested part resulted with large tile together. In this way, we can get more reasonable result-image. Also, we set the clipLimit to small value to avoid checker board effect.

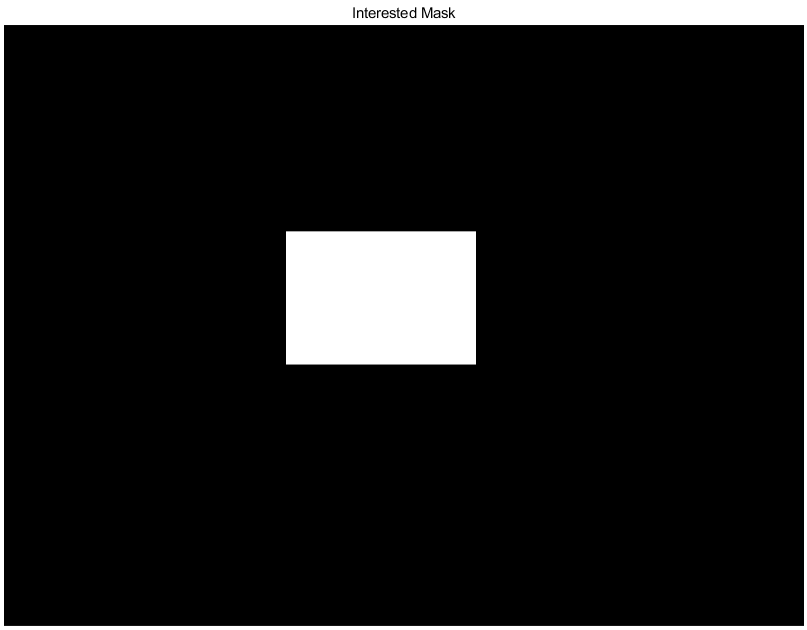
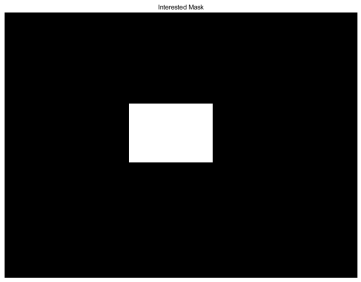


Figure 7: Mask (CLAHE): Since multi-scale tile introduces diffculty into interpolation, we implement the second improvement. In this step, we generate mask for more detailed small-tile based CLAHE algorithm.

Figure 8: Final combined result: Then we use the result from ”numTiles=64, clipLimit=0.08” to fill up the mask, and the other part is filled by ”numTiles=8, clipLimit=0.01”. Please see my code for more details.

(a) Naive HE

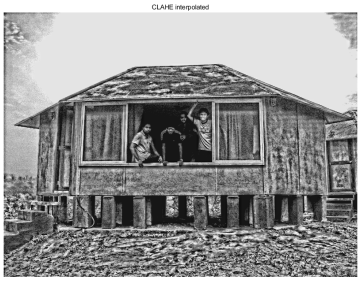
(b) Original Image



(c) Interested Mask

(d) CLAHE: numTiles=8, clipLimit=0.01

(e) Combined CLAHE



(f) CLAHE: numTiles=64, clipLimit=0.08

Figure 9: For better visualization and evaluation, we put some results together for conenvinence. The combined CLAHE: use the result from ”numTiles=64, clipLimit=0.08” to fill up the mask, and the other part is filled by ”numTiles=8, clipLimit=0.01”. And (e) Combined CLAHE is our final result.

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:

(3.a) We use 'padarray' to pad the martix with '0' for full convolution crop convolution. Here is our result,

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

And

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

(3.b)3.b.A For circuitboard-a.tif, the noisy image is caused from underexposure (pepper noise). We find that there are many random black noise points in whole image. It seems that some sensors do not work. Thus, we can apply a mediean filter to the image to get the denoised image. Also, since almost black noise point can not contribute to the original image, we can set a simple threshold to our median filter. Although this leads to some biased to original black regions, it helps to remove all black noise points.

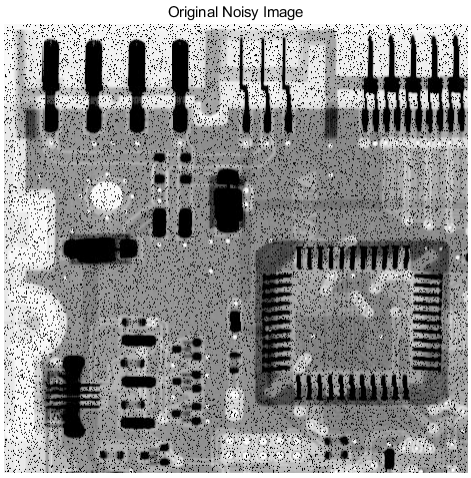


Figure 10: Original image: For circuitboard-a.tif, we find that there are many random black noise points in whole image.

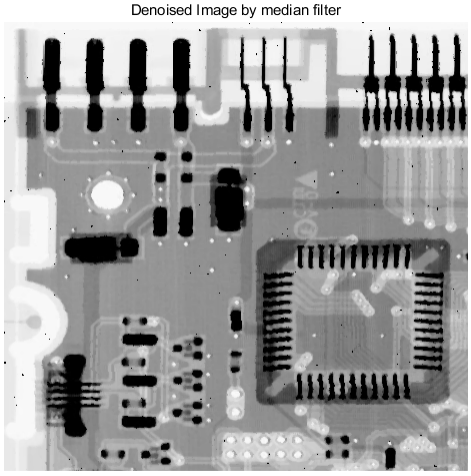


Figure 11: Unbiased Median filter: Thus, we can apply a mediean filter to the image to get the denoised image.

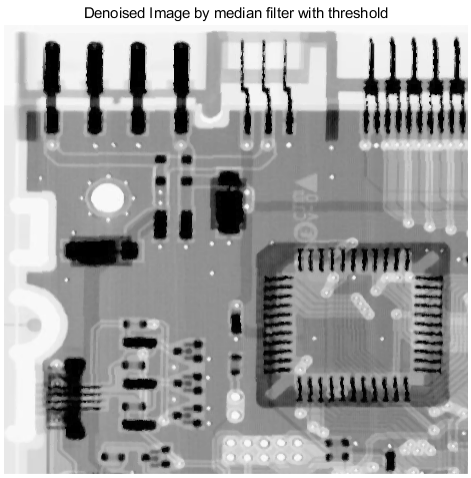


Figure 12: Biased Median filter (circuitboard-a.tif): Also, since almost black noise point can not contribute to the original image, we can set a simple threshold to our median filter. Although this leads to some biased to original black regions, it helps to remove all black noise points.

3.b.B For circuitboard-b.tif, the noisy image is caused from overexposure (salt noise). We find that there are many random white noise points in whole image. We also apply a median filter in this case. Also, we notice that biased median filter can result in very clean image.

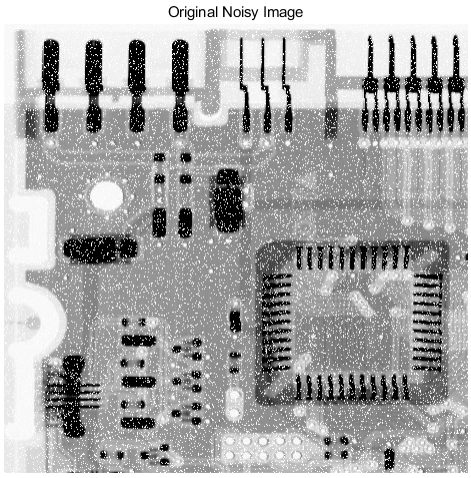


Figure 13: Original image: For circuitboard-b.tif, the noisy image is caused from overexposure. We find that there are many random white noise points in whole image.

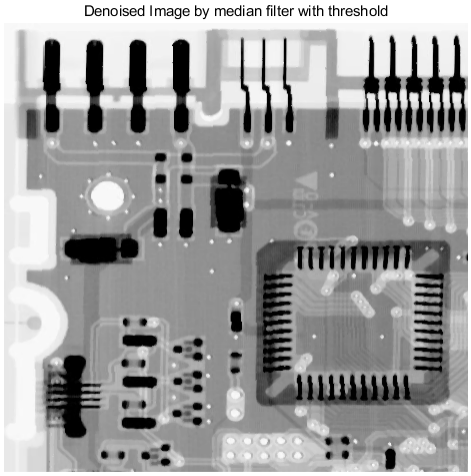


Figure 14: Biased Median filter (circuitboard-b.tif)

3.b.C For circuitboard-c.tif, the noisy image is caused from both underexposure and overexposure. We find that there are many random black and white noise points in whole image. We also apply a median filter in this case. Also, we notice that biased median filter can result in very clean image.

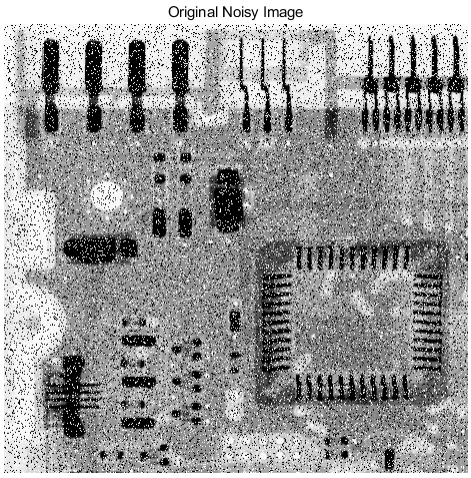


Figure 15: Original image: For circuitboard-c.tif, the noisy image is caused from both underexposure and overexposure. We find that there are many random black and white noise points in whole image.

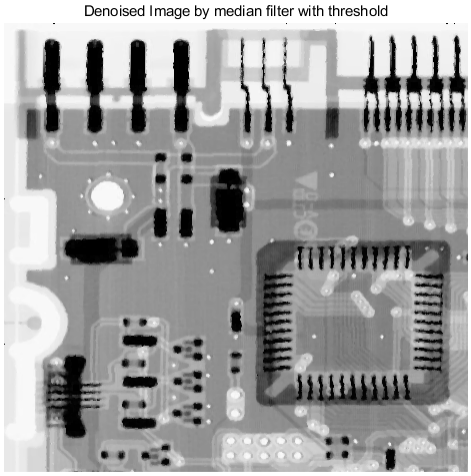


Figure 16: Biased Median filter (circuitboard-c.tif)

3.b.D For cirbuitboard-d.tif, the noisy image , the noise is viewed as uniform noise. Thus, we can adapt alpha trimmed mean filter to remove the uniform noise. We set the windowsize=5, and the factor $d = 6$.

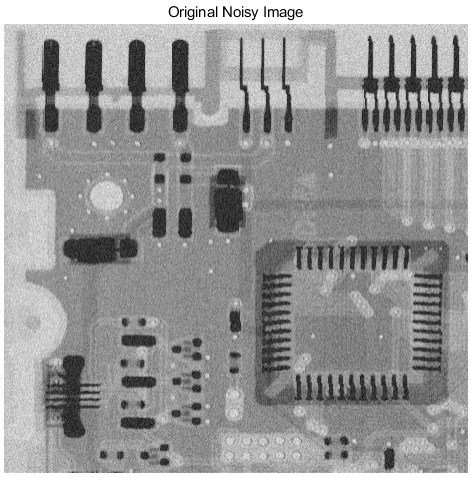


Figure 17: Original image: For circuitboard-d.tif, the noisy image is caused from both underexposure and overexposure. We find that there are many random black and white noise points in whole image.

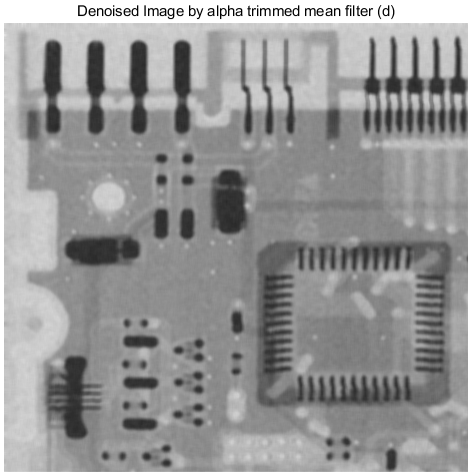


Figure 18: Alpha trimmed mean filter (circuitboard-d.tif)

(3.c) Here is our result,

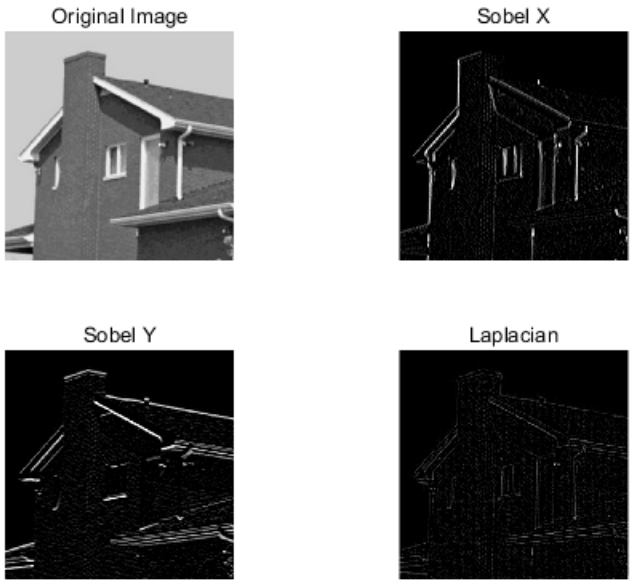


Figure 19: Filtered with both x direction and y direction with 3*3 Sobel mask and Laplacian mask.

(3.d) The function of Laplacian of Gaussian (LoG) Filter is given by

$$G''(x,y) = -\frac{1}{\pi\sigma^4}(1 - \frac{x^2+y^2}{2\sigma^2})e^{-\frac{x^2+y^2}{2\sigma^2}}$$

We also set the windowsize of LoG filter by ksize= $2 * \lfloor 3 * \sigma \rfloor + 1$, which makes sure our LoG filter is stable. Since there are no extra requirements for enhancing factor, we just add the original image and filtered mask by LoG filter directly together to get final sharpened image.

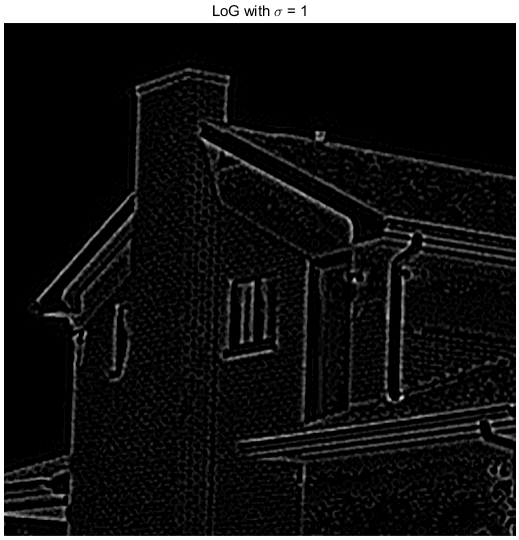


Figure 20: LoG filter, with $\sigma = 1$

Figure 21: Enhanced by LoG filter, with $\sigma = 1$

Figure 22: LoG filter, with $\sigma = 2$

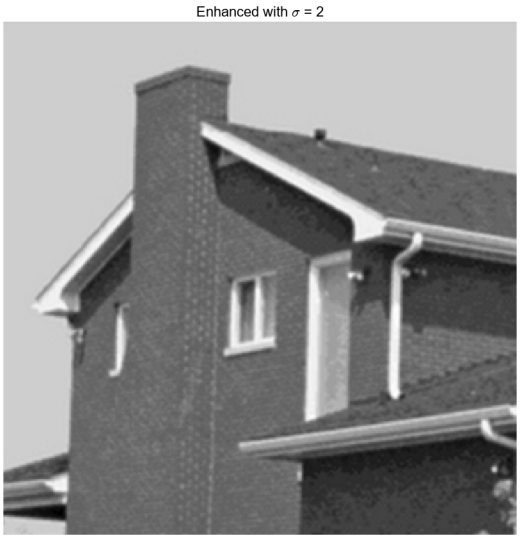


Figure 23: Enhanced by LoG filter, with $\sigma = 2$

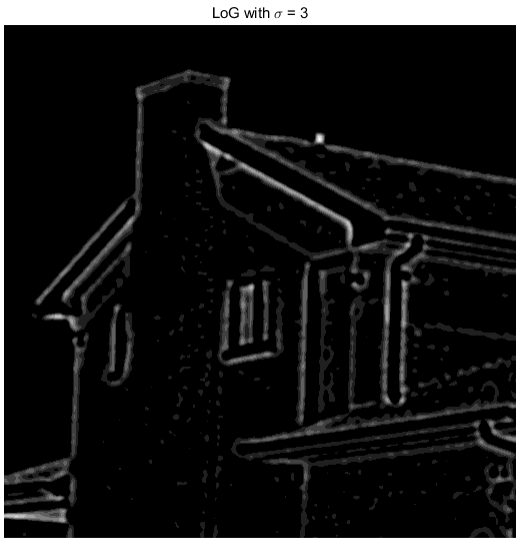


Figure 24: LoG filter, with $\sigma = 3$

Figure 25: Enhanced by LoG filter, with $\sigma = 3$

To perform image enhancing by unsharpen mask method, we first need to define our smoothing filter. We choose Gaussian filter with 'ksize=5' and '*sigma* = 3'. Then we define our original image by f, the smoothing image by s, the mask can be generated by $m = f - s$. And the mask m always implies the boundary information of original image f. Thus, we can generate our enhancing image by $f' = f + k * m$, where k is the enhancing factor. Secondly, we choose our enhancing factor $k = 5$. Finally, we complete the enhancing.

Figure 26: Original image

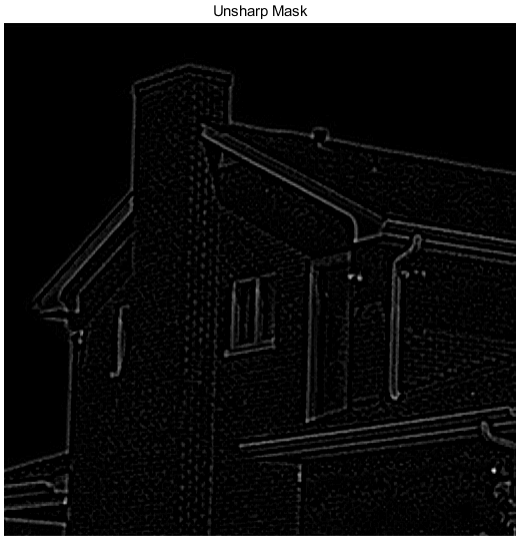


Figure 27: Mask generate from smooth Gaussian filter with 'ksize=5' and '$sigma$ = 3'

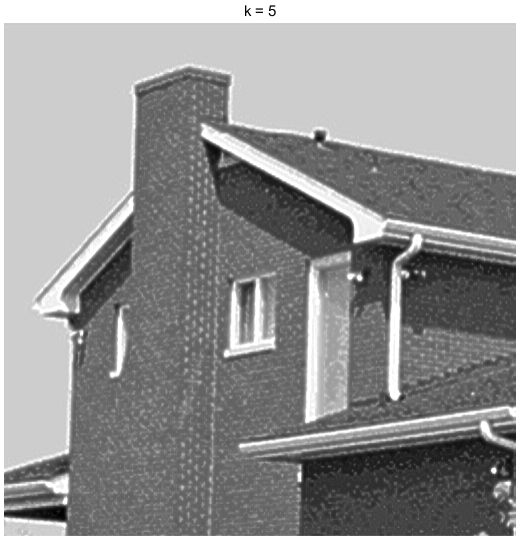


Figure 28: Sharpening Image by Unsharping Mask with $k = 5$, We choose $k = 5$ because of two reasons. On one hand, the image is not sharpened enough when k is small. On the other hand, the image becomes noisy when k is big. Smooth Gaussian filter with 'ksize=5' and '$sigma = 3$'.