Digital Image Processing Project

Yueming Wei
Student ID: 115033910063

31st Dec. 2015

Contents

Problem 1: Histogram Equalization **4**
Generate the histogram . . . 4
Implement the histogram equalization technique . . . 5
Enhanced images . . . 6
Histogram of enhanced image . . . 8

Problem 2: Combining spatial enhancement methods **9**
Laplacian of the original image . . . 9
Laplacian on the original image . . . 10
Sobel gradient of the original image . . . 12
Sobel image smoothed with a 5x5 averaging filter . . . 14
Mask image formed by the product of Laplacian and smoothed-gradient image . . . 14
Adding the product image above to the original image . . . 15
Final result obtained by applying a power-law transformation . . . 16

Problem 3: Filtering in frequency domain **18**
Functions . . . 18
Ideal Filter . . . 18
Butterworth filter . . . 18
Gaussian Filter . . . 22

Problem 4: Noise Generation and Reduction **24**
Noisy . . . 24
Gaussian noisy . . . 24
Rayleign noisy . . . 24
Exponential noisy . . . 24
Gamma nosiy . . . 24
Uniform nosiy . . . 24
salt-and-pepper noisy . . . 25
Filter . . . 25
Arithmetic mean filter . . . 25
Geometric mean filter . . . 25
Harmonic mean filter . . . 25
Constraharmonic mean filter . . . 25
Test images . . . 25

Problem 5: Image restoration **29**
Function . . . 29
H(u,v) . . . 29
Inverse filter . . . 29
Wiener filter . . . 29
Experiment image results . . . 29

Problem 6: Geometric transform **32**
Algorithm . . . 32
Exoeriment results . . . 32

Problem 8: Morpholigical Processing **38**
Erosion and Dilation . . . 38
Dilation . . . 38
Erosion . . . 38
Opening and closing . . . 38
Opening . . . 38
Closing . . . 39
Boundary extraction . . . 39
Hole filling . . . 40
Connected component extraction . . . 40

Problem 9: Image segmentation **42**
Dectors . . . 42
Roberts . . . 42
Prewitt . . . 42
Sobel . . . 45
marr-Hildreth . . . 45
Canny . . . 45
Thresholding segmentation . . . 45
Global threshold . . . 45
Ostu . . . 49

Problem 1: Histogram Equalization

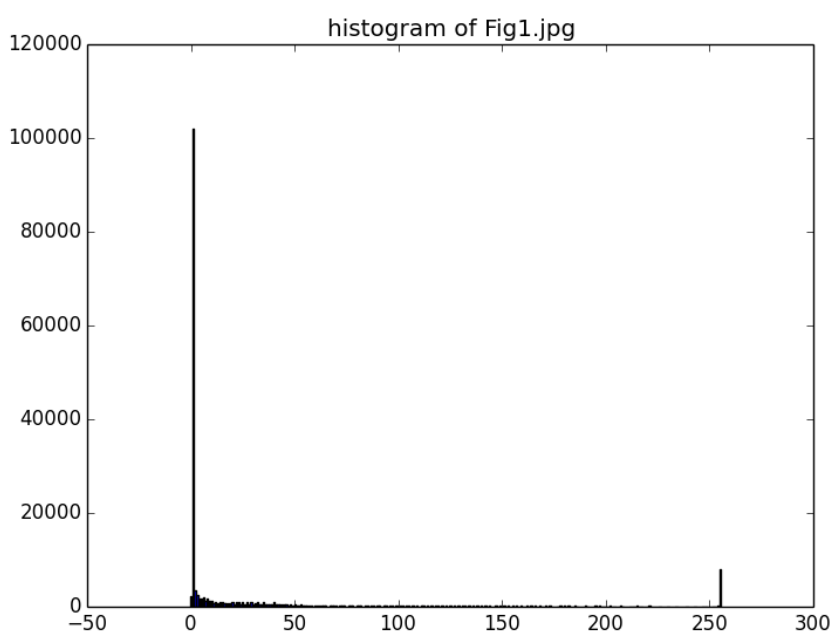
Histogram Equalization (test images: fig1.jpg, fig2.jpg)
(a) Write a computer program for computing the histogram of an image.
(b) Implement the histogram equalization technique.
(c) Your program must be general to allow any gray-level image as its input.
As a minimum, your report should include the original image, a plot of its histogram, a plot of the histogram-equalization, transformation function, the enhanced image, and a plot of its histogram.

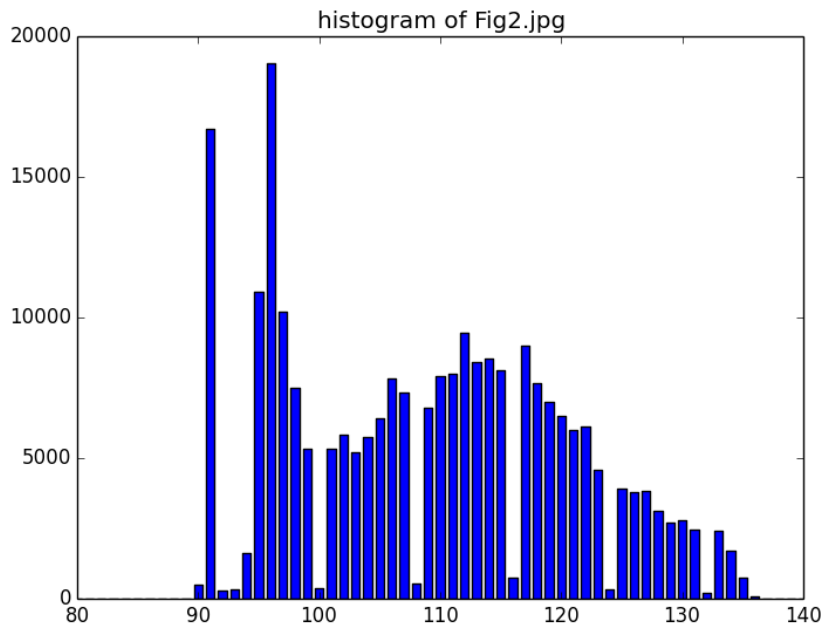
Generate the histogram

Generating the histogram of an image using the cumulative distribution function of each pixel value:

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

The histogram pictures of Fig1.jpg and Fig2.jpg are as follows:





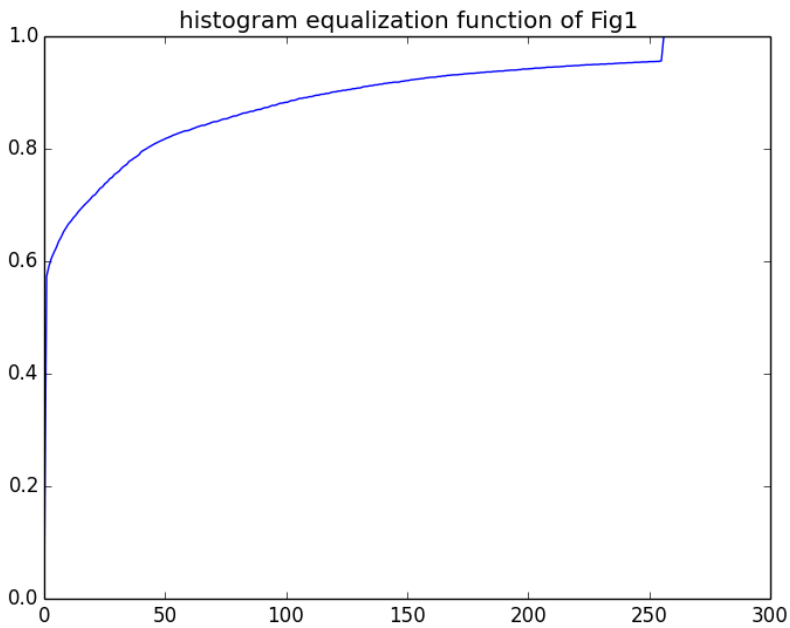
Implement the histogram equalization technique

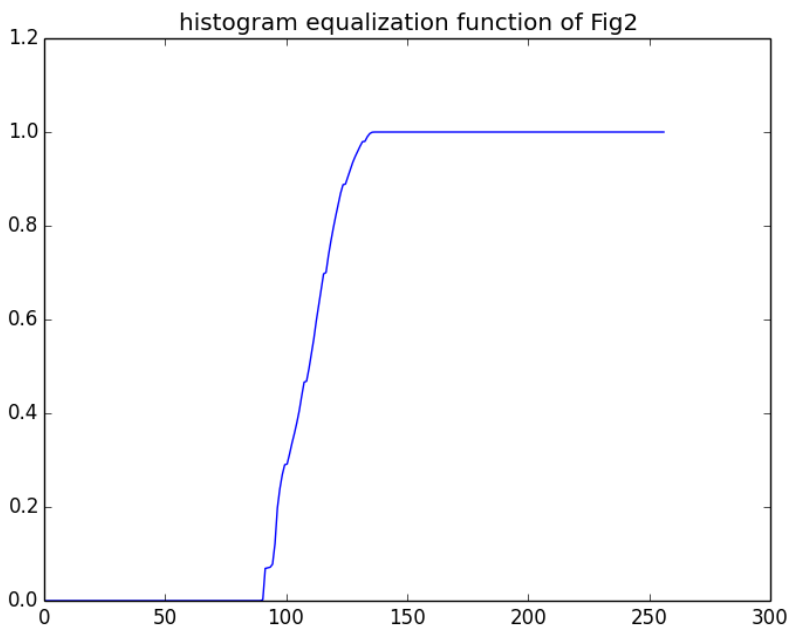
We use those functions to calculate the histogram equalization:

$$P_y(i) = T[cdf_x(i)]$$

$$P_x(i) = round[\frac{cdf(v) - cdf_{min}}{M * N - cdf_{min}} \times (Graylevel - 2)] + 1$$

The histogram equalization of Fig1.jpg and Fig2.jpg are as follows:





Enhanced images

We use the function

$$Image(i, j) = cdf_y(k)$$

to enhance the original images. The original images and enhanced images are as follows.

Figure 1: Original Fig1

Figure 2: Enhanced Fig1

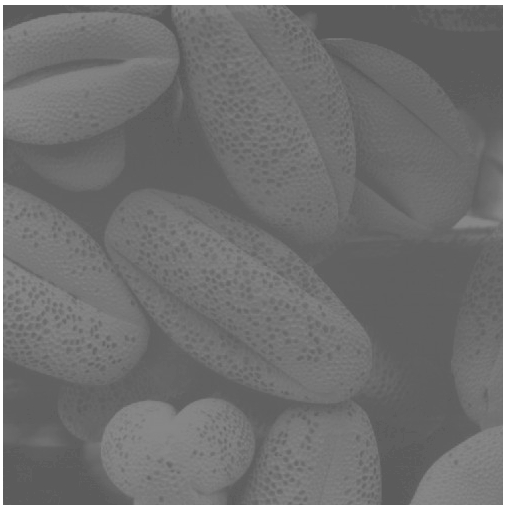


Figure 3: Original Fig1

Figure 4: Enhanced Fig1

Histogram of enhanced image

The histogram of original image and enhanced images are as follows.

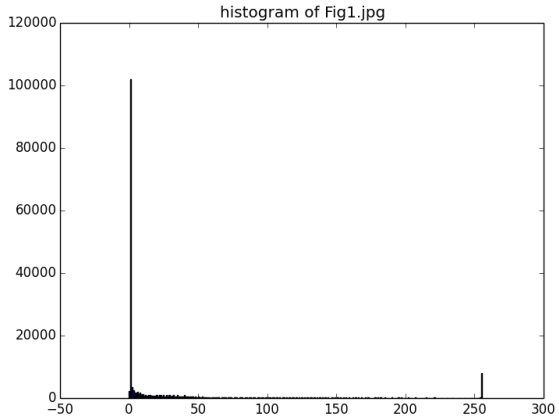


Figure 5: Histogram of original Fig1

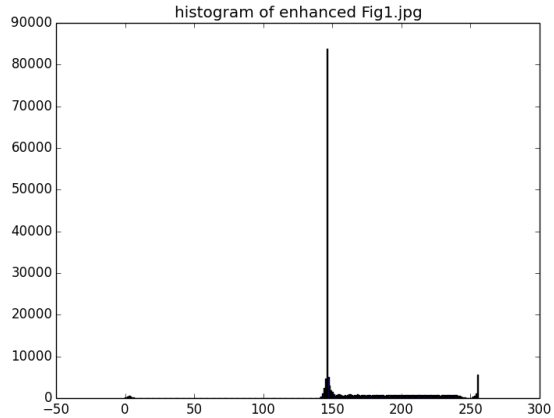


Figure 6: Histogram of enhanced Fig1

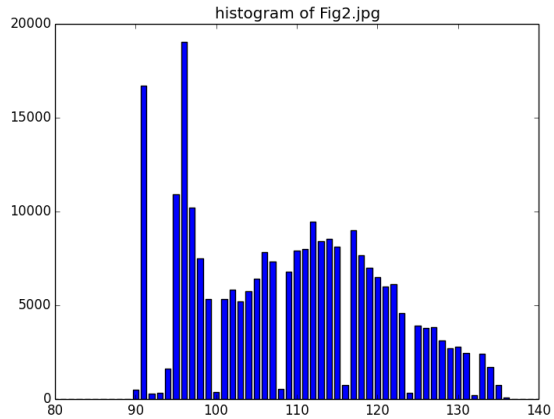


Figure 7: Histogram of original Fig2

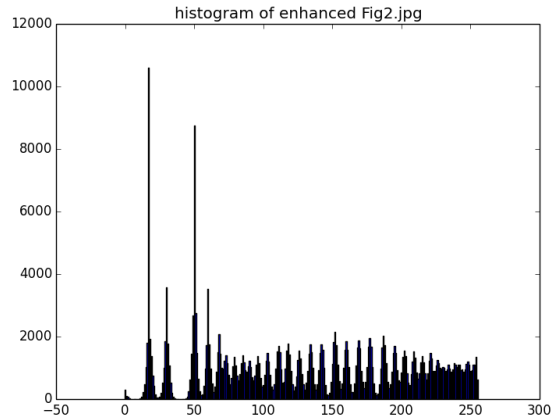


Figure 8: Histogram of enhanced Fig2

Problem 2: Combining spatial enhancement methods

Combining spatial enhancement methods

Implement the image enhancement task of Section 3.7 (Fig 3.43) (Section 3.8, Fig. 3.46 in our slides). The image to be enhanced is skeleton_orig.tif. You should implement all steps in Figure 3.43. (You are encouraged to implement all functions by yourself, attempting not to directly use Matlab functions such as imfilter or fspecial.)

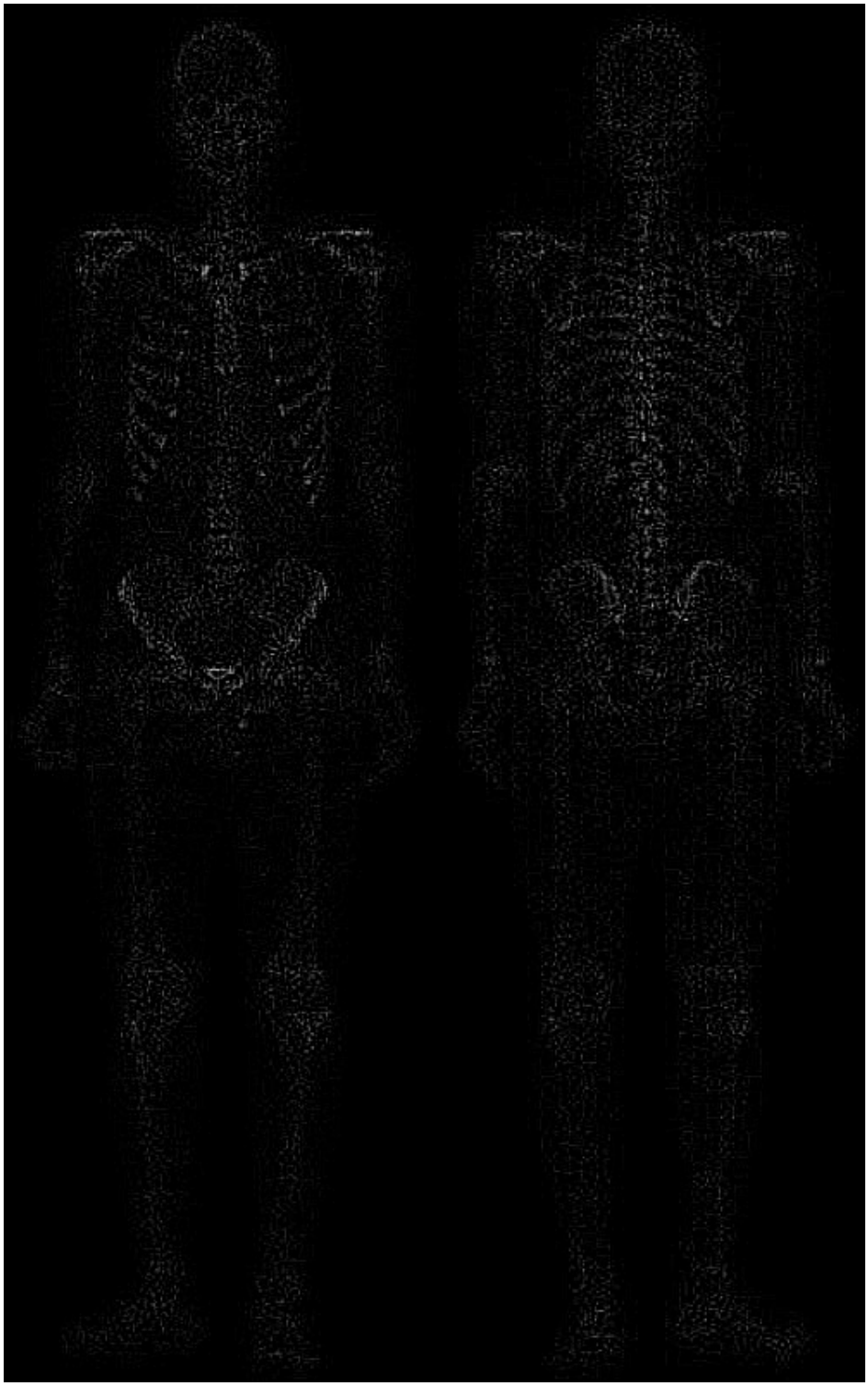
Laplacian of the original image

We will apply Laplacian on the original image to get the image (b). The filter used is as Table 1.

Table 1: Laplacian filter mask

-1	1	-1
-1	8	-1
-1	-1	-1

The processed image is as follows:



Laplacian on the original image

The we add the Laplacian of the original image to the original image, we will get the new image. The new image is a rather noisy sharpened image

The we add the Laplacian of the original image to the original image, we will get the new image. The new image is a rather noisy sharpened image

Sobel gradient of the original image

We will use two mask to separately get the components gx and gy. Then add the two components together, we will get the the sober gradient of the original image. The new image is as follows. As we can see, edges are much more dominant in this image than in the Laplacian image.

Table 2: Sobel gx mask

-1	-2	-1
0	0	0
-1	-2	-1

Table 3: Sobel gy mask

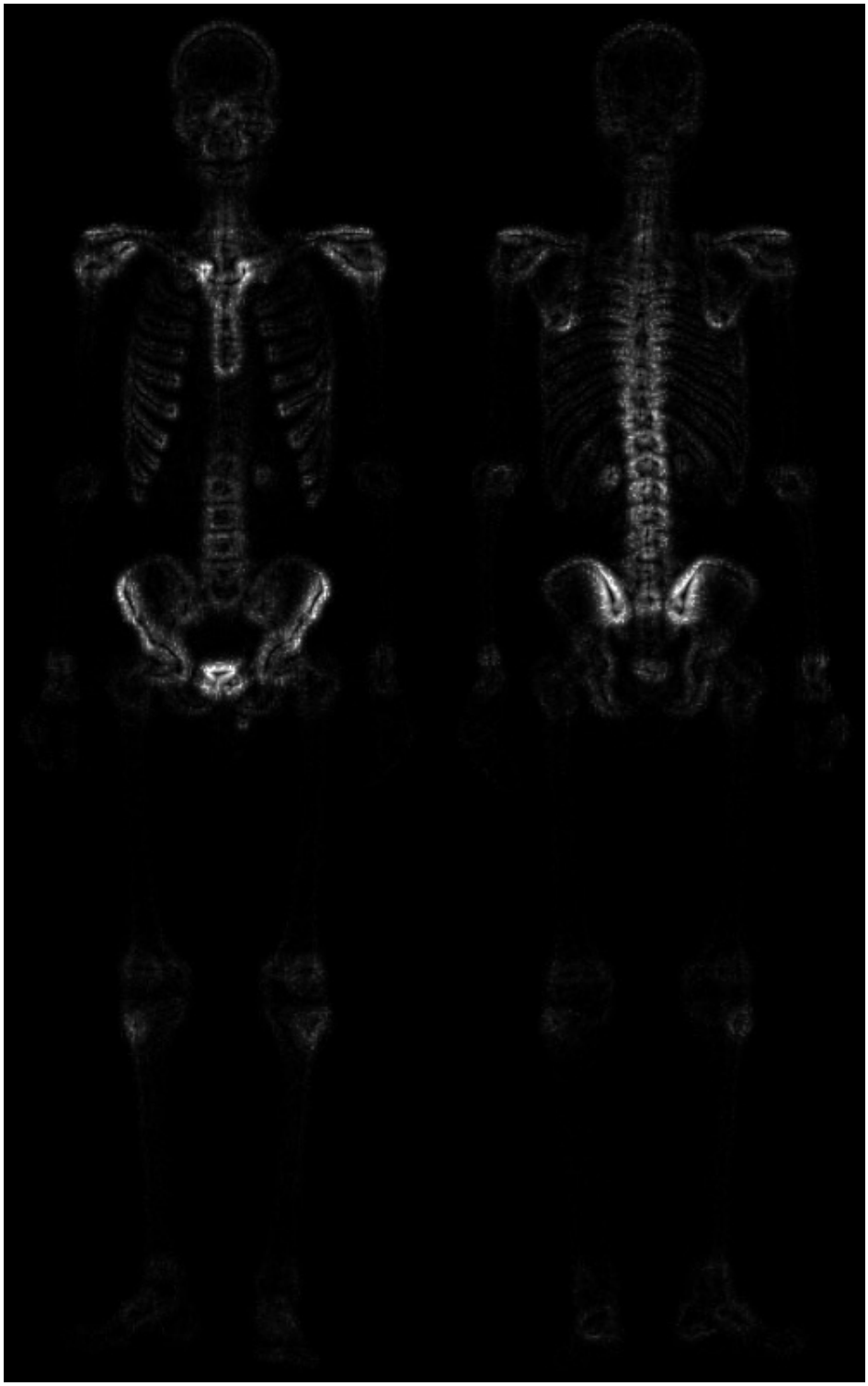
-1	0	-1
-2	0	-2
-1	0	-1

The processed image is as follows:

Sobel image smoothed with a 5x5 averaging filter

Mask image formed by the product of Laplacian and smoothed-gradient image

The dominance of the strong edges and the relative lack of visible noise, which is the key objective behind masking the Laplacian with a smoothed gradient image.



Adding the product image above to the original image

This operation results in the sharpened image below. The significant increase in sharpness of detail in this image over the original is evident in most parts of the image, including the ribs, spinal cord, pelvis and skull. This type of improvement would not have been possible by using the Laplacian or the gradient alone.

Final result obtained by applying a power-law transformation

The sharpening procedure does not affect in an appreciable way the dynamic range of the intensity levels in an image. We apply the power-law transformation on the processed image and got the image below.

Problem 3: Filtering in frequency domain

Filtering in frequency domain
Implement the ideal, Butterworth and Gaussian lowpass and highpass filters and test them under different parameters using characters_test_pattern.tif.

Functions

We first need transform the image to gray image and then use the 2D Fourier transformation belw to change it to the frequence domain.

$$F(u,v) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0} N-1 f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

The inverse funtion of FFT is as folows:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0} N-1 F(u,v)e^{-j2\pi(ux/M+vy/N)}$$

Ideal Filter

The functions we implement are as follows:

$$H(u,v) = \begin{cases} 1 & D(u,v) \leq D_0 \\ 0 & D(u,v) > D_0 \end{cases}$$

Here, $D(u,v)$ is defined as below:

$$D(u,v) = [(u-M/2)^2 + (v-N/2)^2]^{1/2}$$

After applying Ideal filter to the image with different parameters, we get the following images.

Figure 9: Ideal filter with 10 radius lowpass

Figure 10: Ideal filter with 30 radius lowpass

Butterworth filter

$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$

Figure 11: Ideal filter with 60 radius lowpass

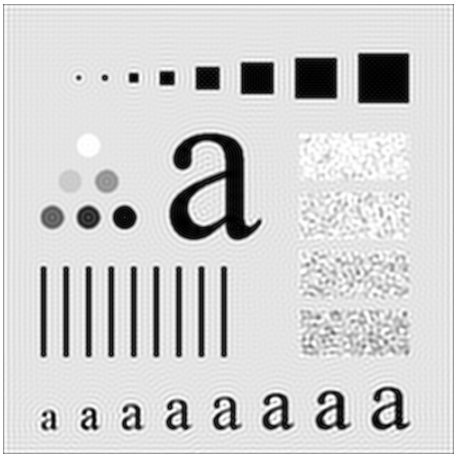


Figure 12: Ideal filter with 160 radius lowpass

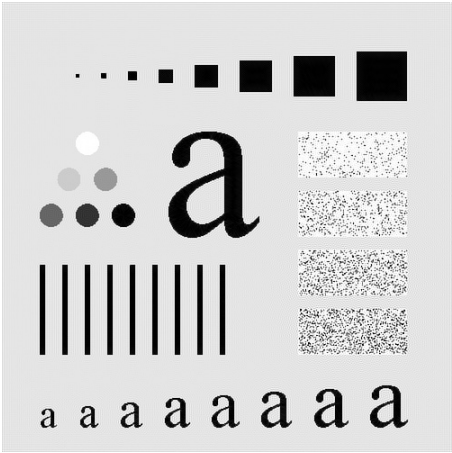


Figure 13: Ideal filter with 460 radius lowpass

Figure 14: Ideal filter with 10 radius highpass

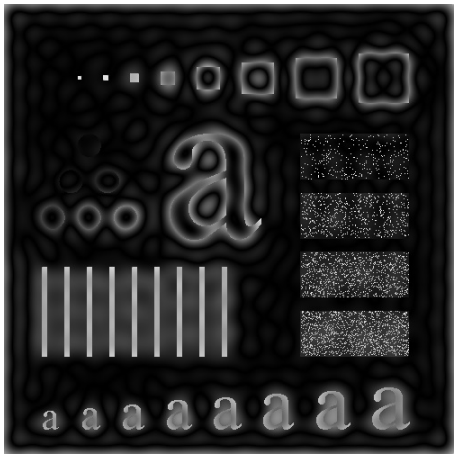


Figure 15: Ideal filter with 30 radius highpass

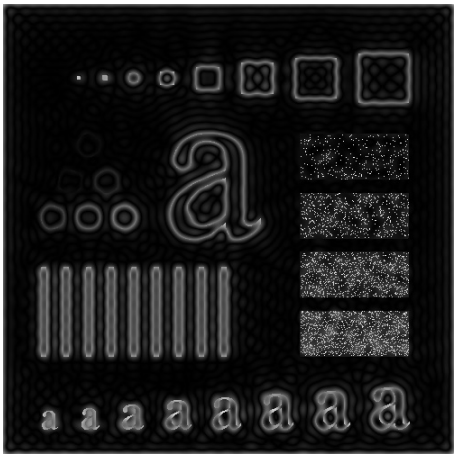


Figure 16: Ideal filter with 60 radius highpass

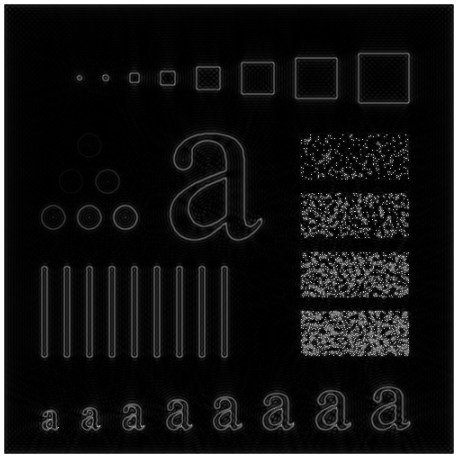


Figure 17: Ideal filter with 160 radius highpass

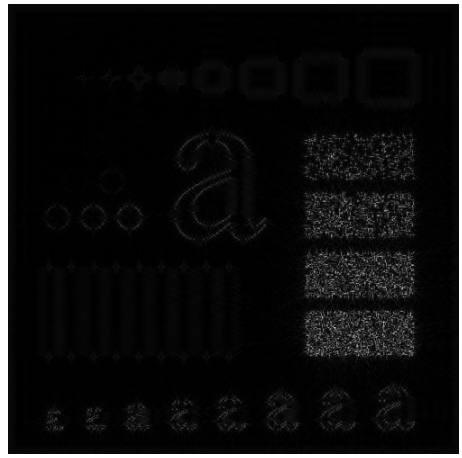


Figure 18: Ideal filter with 460 radius highpass

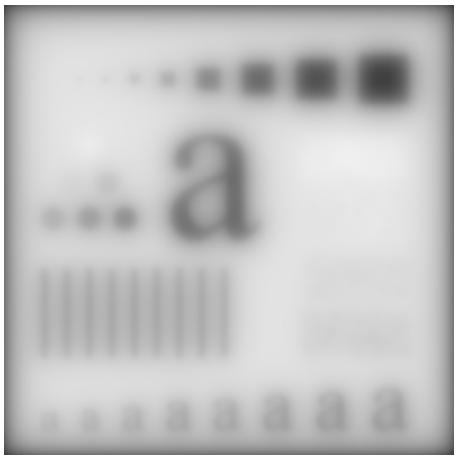


Figure 19: HButterwoth filter with 10 radius low-pass

Figure 20: Butterwoth filter with 30 radius low-pass

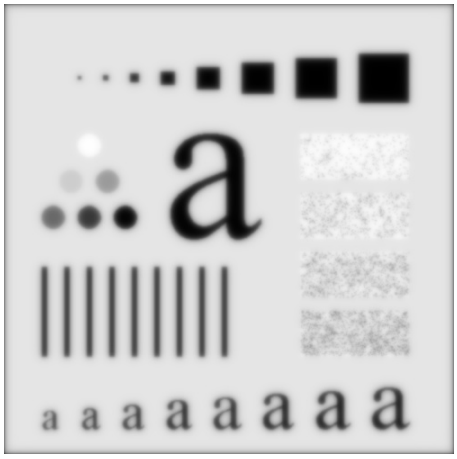


Figure 21: Butterwoth filter with 60 radius low-pass

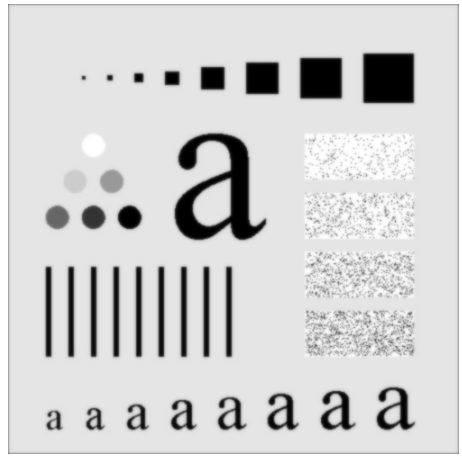


Figure 22: Butterwoth filter with 160 radius low-pass

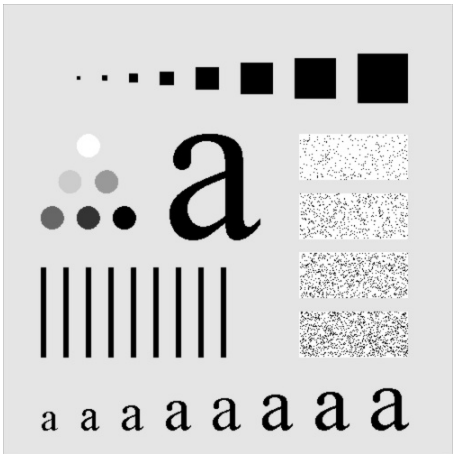


Figure 23: Butterwoth filter with 460 radius low-pass

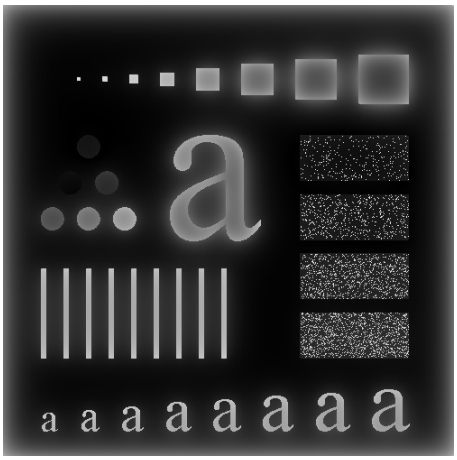


Figure 24: Butterwoth filter with 10 radius high-pass

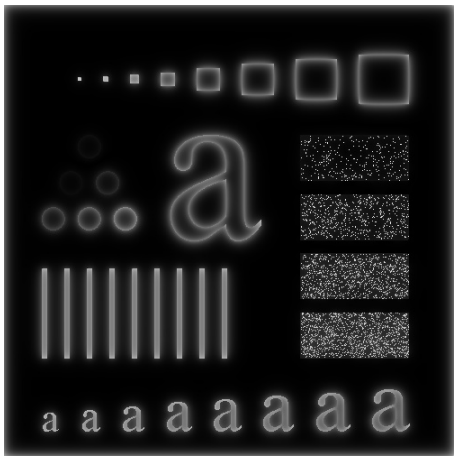


Figure 25: Butterwoth filter with 30 radius high-pass

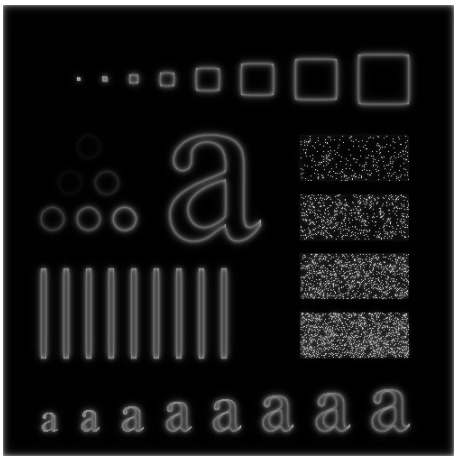


Figure 26: Butterwoth filter with 60 radius high-pass

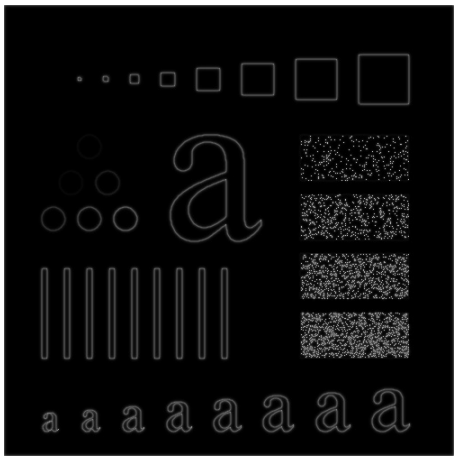


Figure 27: Butterwoth filter with 160 radius highpass

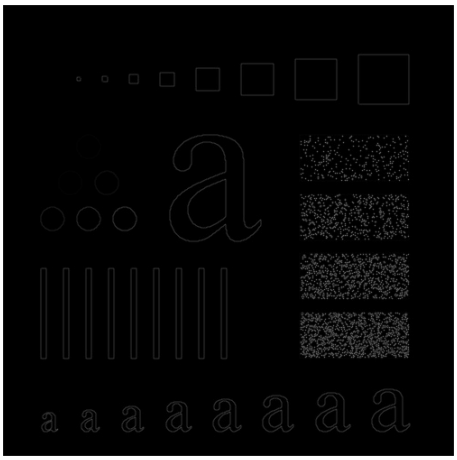


Figure 28: Butterwoth filter with 460 radius highpass

Gaussian Filter

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$

Figure 29: Gaussian filter with 10 radius lowpass

Figure 30: Gaussian filter with 30 radius lowpass

Figure 31: Gaussian filter with 60 radius lowpass

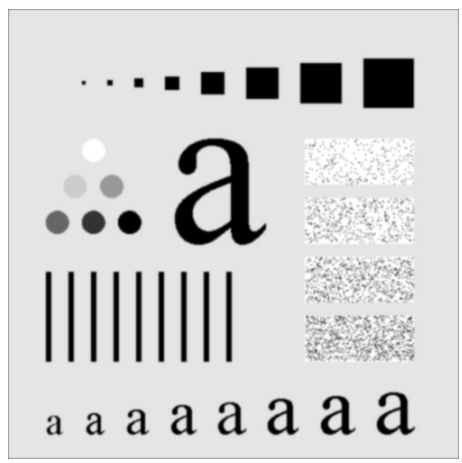


Figure 32: Gaussian filter with 160 radius lowpass

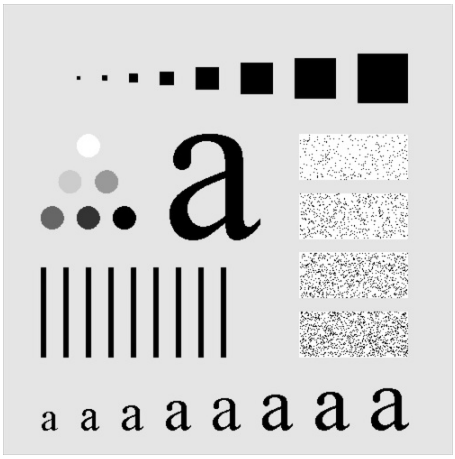


Figure 33: Gaussian filter with 460 radius low-pass

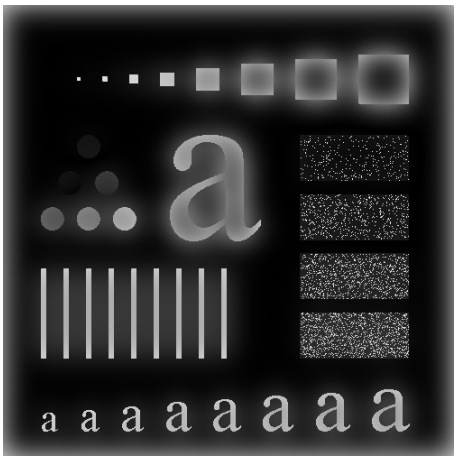


Figure 34: Gaussian filter with 10 radius high-pass

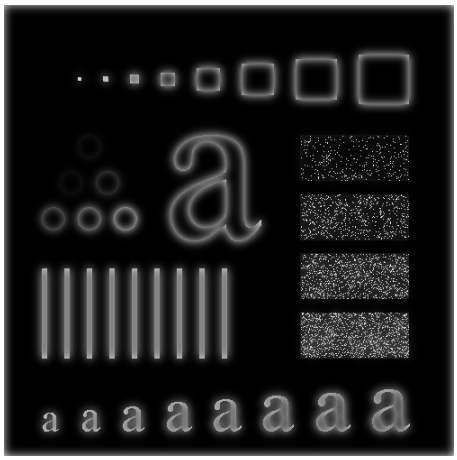


Figure 35: Gaussian filter with 30 radius high-pass

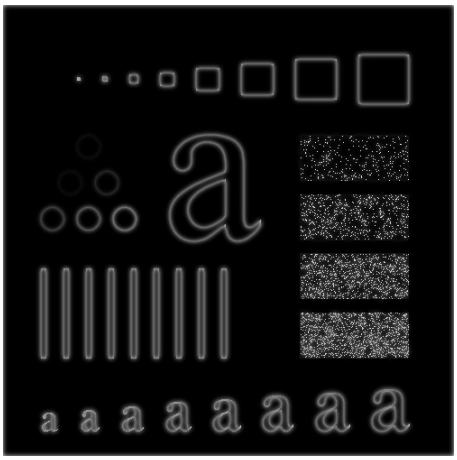


Figure 36: Gaussian filter with 60 radius high-pass

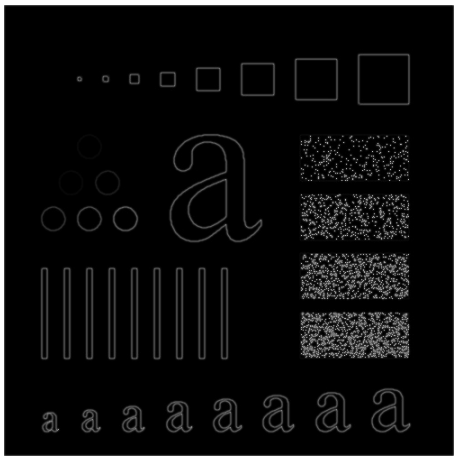


Figure 37: Gaussian filter with 160 radius high-pass

Figure 38: Gaussian filter with 460 radius high-pass

Problem 4: Noise Generation and Reduction

Generating different noise and comparing different noise reduction methods
In this problem, you are required to write a program to generate different types of random noises started from the Uniform noise and Gaussian noise. (one of the reference may be Digital image processing using Matlab PP.143-150. And then add some of these noises to the circuit image (Circuit.tif) and investigate the different mean filters and order statistics as the textbook did at pages 344-352.

Noisy

Gaussian noisy

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2}$$

Here, z is the gray value and μ is the expected value of z, σ is the standard deviation of z.

Rayleign noisy

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & z \geq a \\ 0 & z < a \end{cases}$$

Exponential noisy

$$p(z) = \begin{cases} ae^{-az} & z \geq a \\ 0 & z < a \end{cases}$$

Here,

$$\mu = \frac{1}{a}$$

$$\sigma^2 = \frac{1}{a^2}$$

Gamma nosiy

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & z \geq a \\ 0 & z < a \end{cases}$$

Here,

$$\mu = \frac{b}{a}$$

$$\sigma^2 = \frac{b}{a^2}$$

Uniform nosiy

$$p(z) = \begin{cases} \frac{1}{b-a} e^{-az} & a \leq z \leq b \\ 0 & otherwise \end{cases}$$

Here,

$$\mu = \frac{a+b}{2}$$

$$\sigma^2 = \frac{(b-a)^2}{12}$$

salt-and-pepper noisy

$$p(z) = \begin{cases} P_a & z = a \\ P_b & z = b \\ 0 & otherwise \end{cases}$$

Filter

Arithmetic mean filter

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

Geometric mean filter

$$\hat{f}(x,y) = [\prod_{(s,t)\in S_{xy}} g(s,t)]^{\frac{1}{mn}}$$

Harmonic mean filter

$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

Constraharmonic mean filter

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{x,y}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{S_{xy}}} g(s,t)^{Q}}$$

Test images

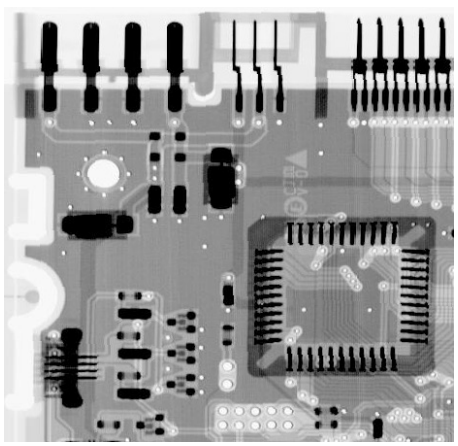


Figure 39: X-ray image

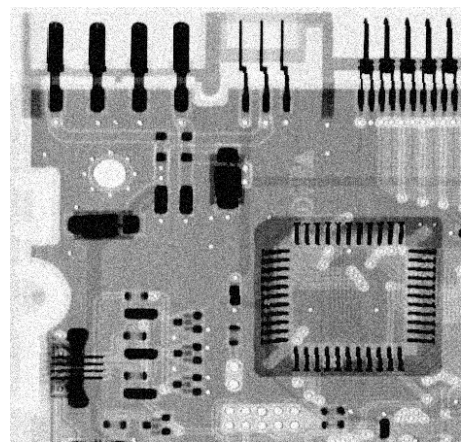


Figure 40: Image corrupted by additive

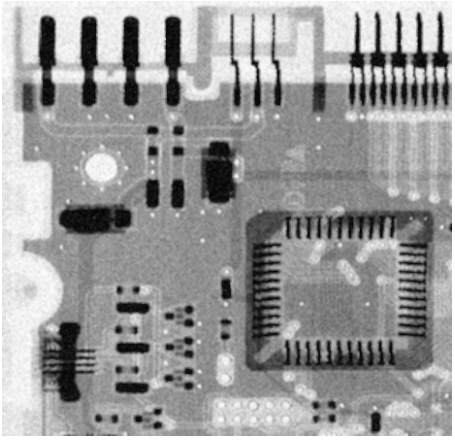


Figure 41: Result of fitering with an arithmetic mean filter of the size 5×5

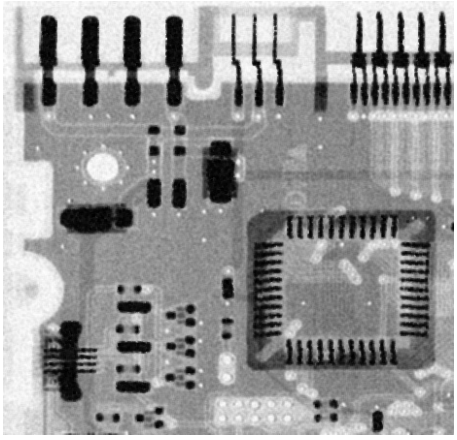


Figure 42: Result of fitering with a geometric mean filter of the same size

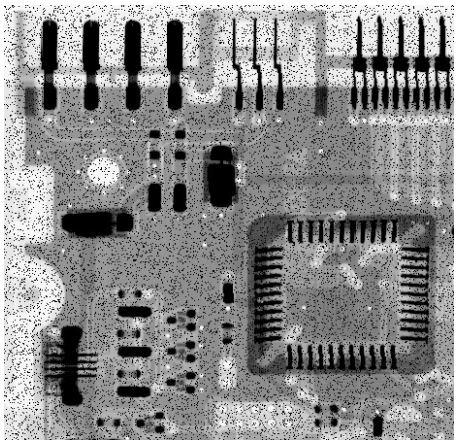


Figure 43: Image corrupted by pepper noise with a probability of 0.1

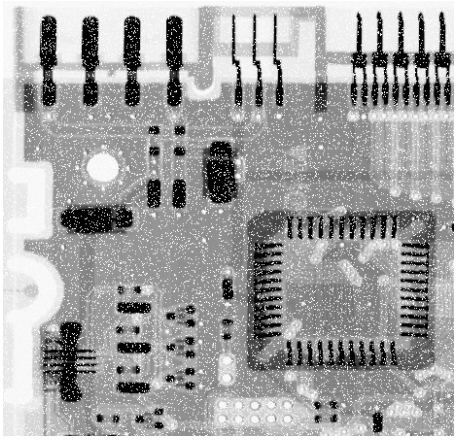


Figure 44: Image corrupted by salt noisy with the sma e probabilty

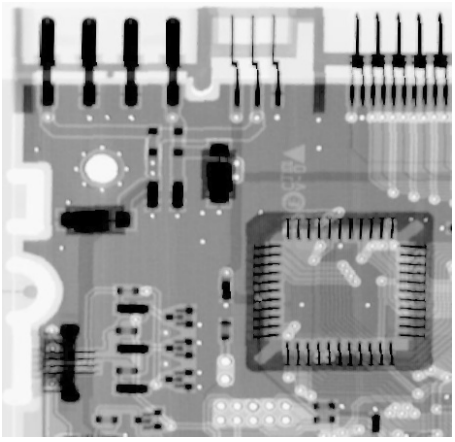


Figure 45: Result of fitering with a 3×3 contra-harmonic filter of order 1.5

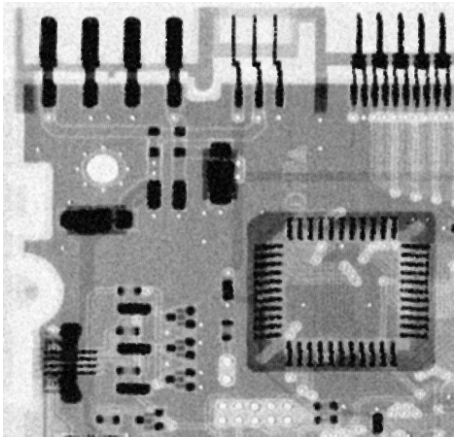


Figure 46: Result of fitering with a Q = -1.5

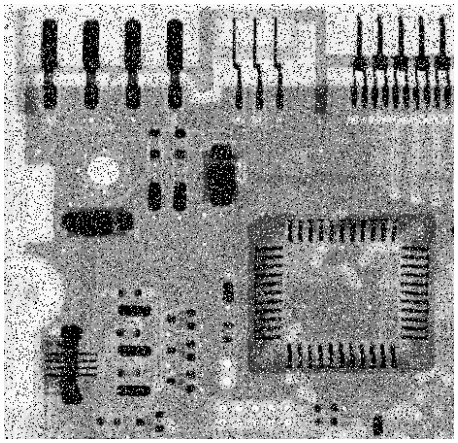


Figure 47: Image corrupted by salt-and-pepper noise with probabilities $P_a = P_b = 0.1$

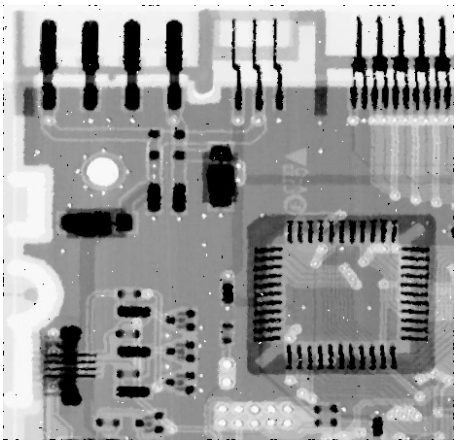


Figure 48: Result of one pass with a median filter of size 3× 3

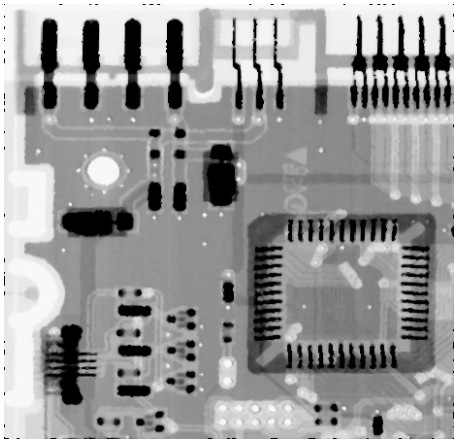


Figure 49: Result of processing the above image with this filter

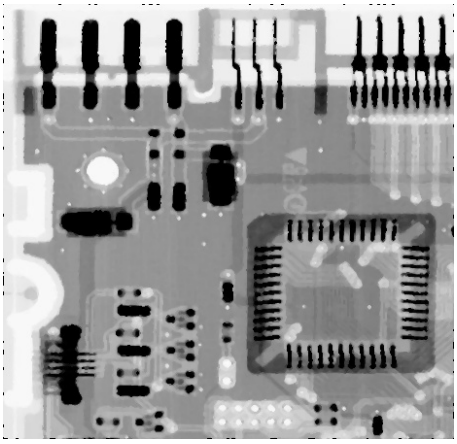


Figure 50: Result of processing with the same filter

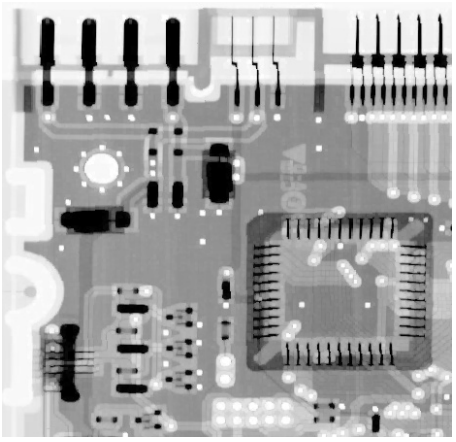


Figure 51: Result of filtering with a max filter of size 3×3

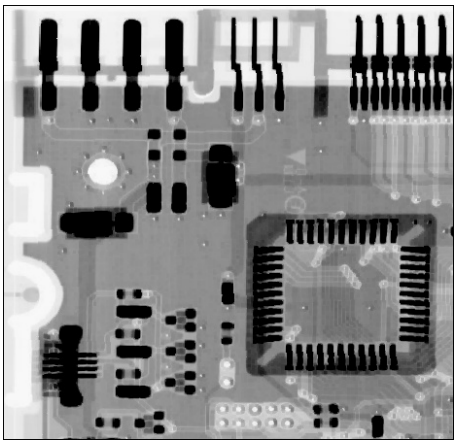


Figure 52: Result of filtering with a min filter of the same size

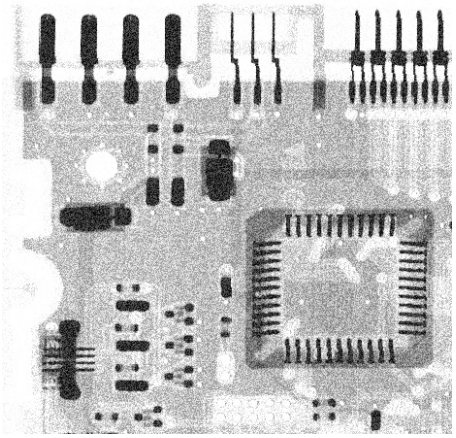


Figure 53: Image corrupted by additive uniform noise

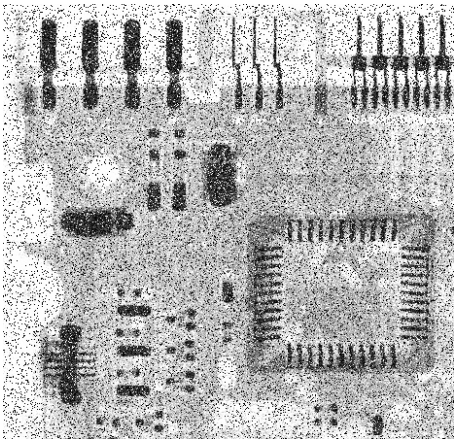


Figure 54: Image additionally corrupted by additive salt-and-pepper noise

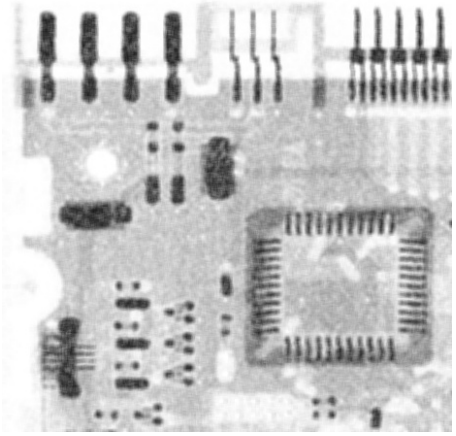


Figure 55: Arithmetic mean filter

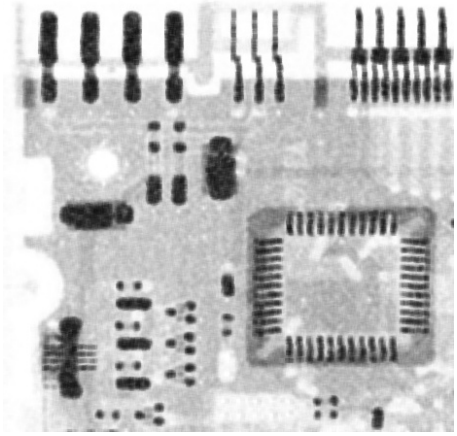


Figure 56: geometric mean filter

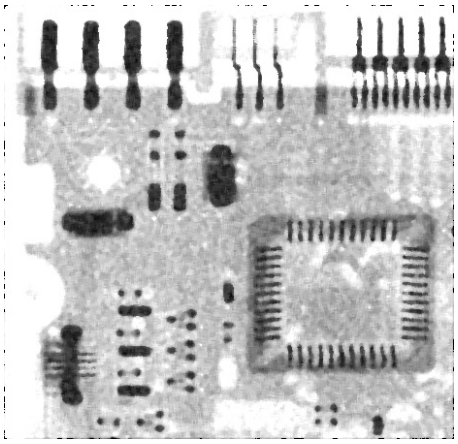


Figure 57: median filter

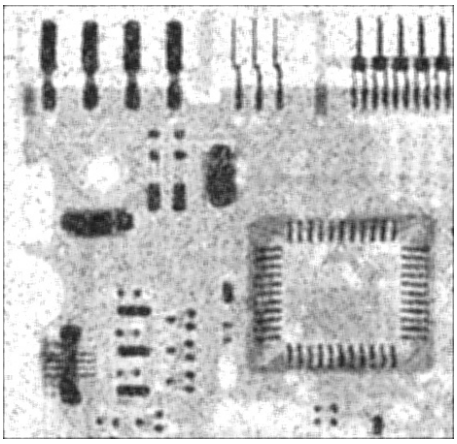


Figure 58: alpha-trimmed mean filter with d=5

Problem 5: Image restoration

Image restoration (Test image: book_cover.jpg)
Suppose a blurring degradation function as

$$H(u,v) = \frac{T}{\pi(ua+vb)} sin[\pi(ua+vb)]e^{-j\pi(ua+vb)}$$

(a) Implement a blurring filter using Eq. (1).
(b) Blur the test image book_cover.jpg using parameters a=b=0.1 and T = 1.
(c) Add Gaussian noise of 0 mean and variance of 650 to the blurred image.
(d) Restore the blurred image and the blurred noisy image using the inverse filter, Wiener deconvolution filter, respectively.
(e) Add Gaussian noise of 0 and different variances to the blurred image and repeat (d), investigate the performance of the Wiener deconvolution filter.

Function

H(u,v)

Assume that we know $x_0(t), y_0(t)$, and the image is moving in x-axis and y-axis moving according to the $x_0(t) = at/T$, $y_0(t) = bt/T$. The we know:

$$H(u,v) = \frac{T}{\pi(ua+vb)} sin[\pi(ua+vb)]e^{-j\pi(ua+vb)}$$

Inverse filter

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

Wiener filter

$$\hat{F}(u,v) = [\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + k}]G(u,v)$$

Experiment image results

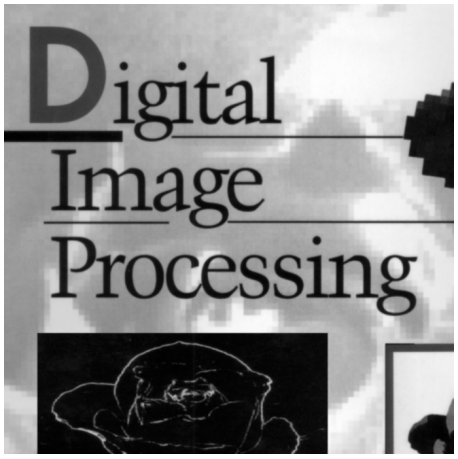


Figure 59: Original image

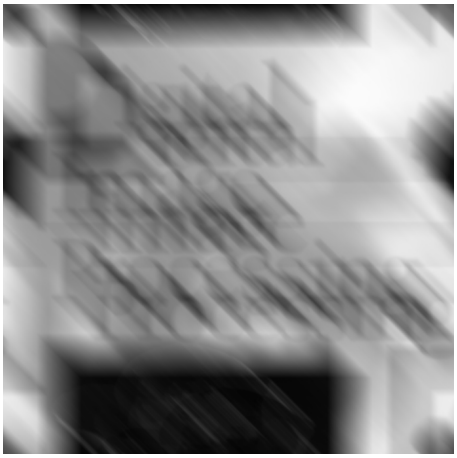


Figure 60: Blurred image using H(u,v) with a=b=0.1 and T = 1

Figure 61: Blurred image using inverse filter

Figure 62: Blurred image using wiener filter

Figure 63: Image corrputed by motion blur and additive nosie

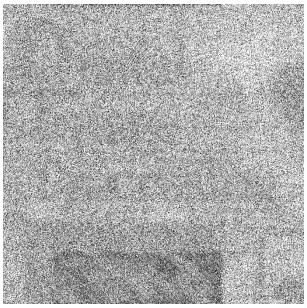


Figure 64: Result of inverse filter

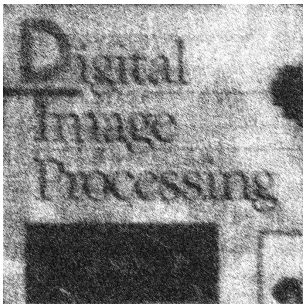


Figure 65: Result of wiener filter

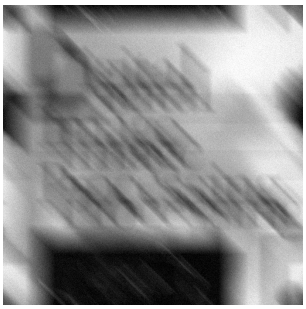


Figure 66: Image corrputed by motion blur and additive nosie but noisy variance one order of the magnitude less

Figure 67: Result of inverse filter

Figure 68: Result of wiener filter

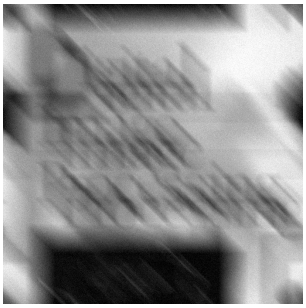


Figure 69: Image corrputed by motion blur and additive nosie but noisy variance five order of the magnitude less

Figure 70: Result of inverse filter

Figure 71: Result of wiener filter

Problem 6: Geometric transform

Geometric transform (test image: ray_trace_bottle.tif) Develop a geometric transform program that will rotate, translate, and scale an image by specified amounts, using the nearest neighbor and bilinear interpolation methods, respectively.

Algorithm

In terms of digital image processing, a geometric transformation consists of two basic operations: (1) a spatial transformation, which defines the rearrangement of pixels on the image plane; and (2) gray-level interpolation, which deals with the assignment of gray levels to pixels in the spatially transformed image.
Suppose that an image f with pixel coordinates (x, y) undergoes geometric distortion to produce an image g with coordinates $(x^{'}, y^{'})$. And this transformation may be expressed as

$$x^{'} = r(x, y)$$
$$y^{'} = s(x, y)$$

One of the most commonly used forms of spatial transformations is the affine transform. The affine transform can be written in a matrix form as

$$[x \;\; y \;\; 1] = [x^{'} \;\; y^{'} \;\; 1]T = [x^{'} \;\; y^{'} \;\; 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \tag{1}$$

And in my implementation I use different value of elements in transformation matrix.

Exoeriment results

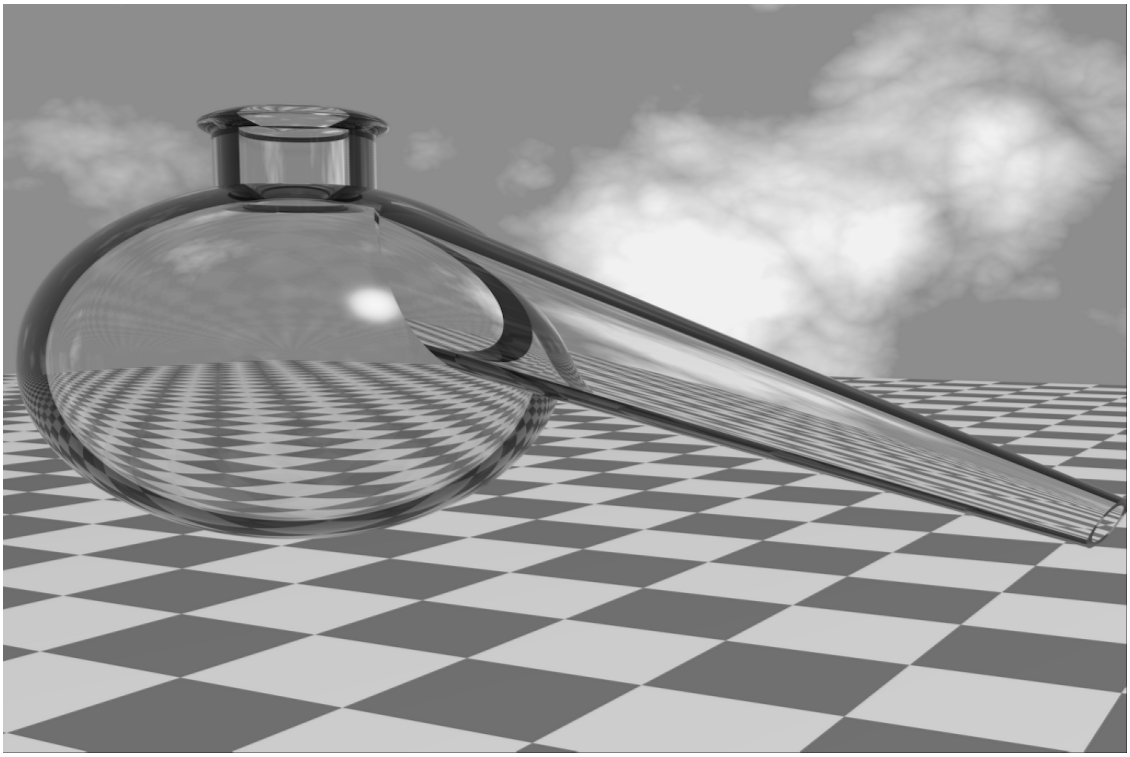


Figure 72: Scale using bilinear

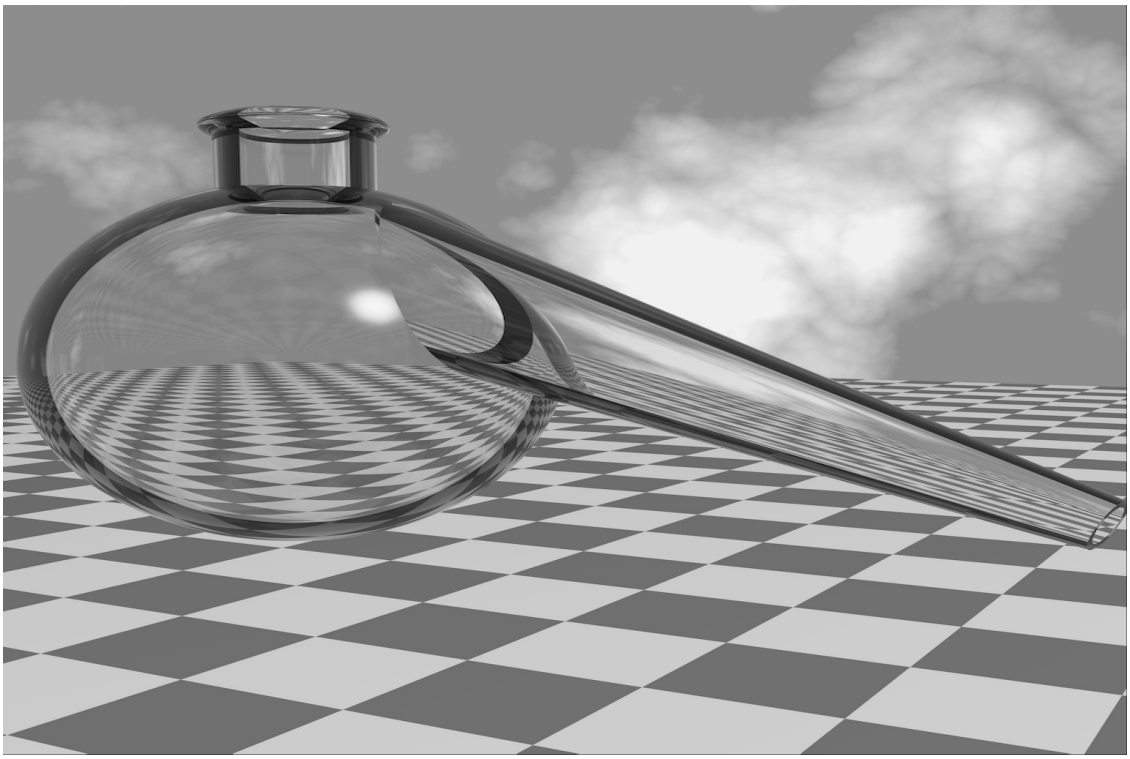


Figure 73: Scale using nearest neighbour

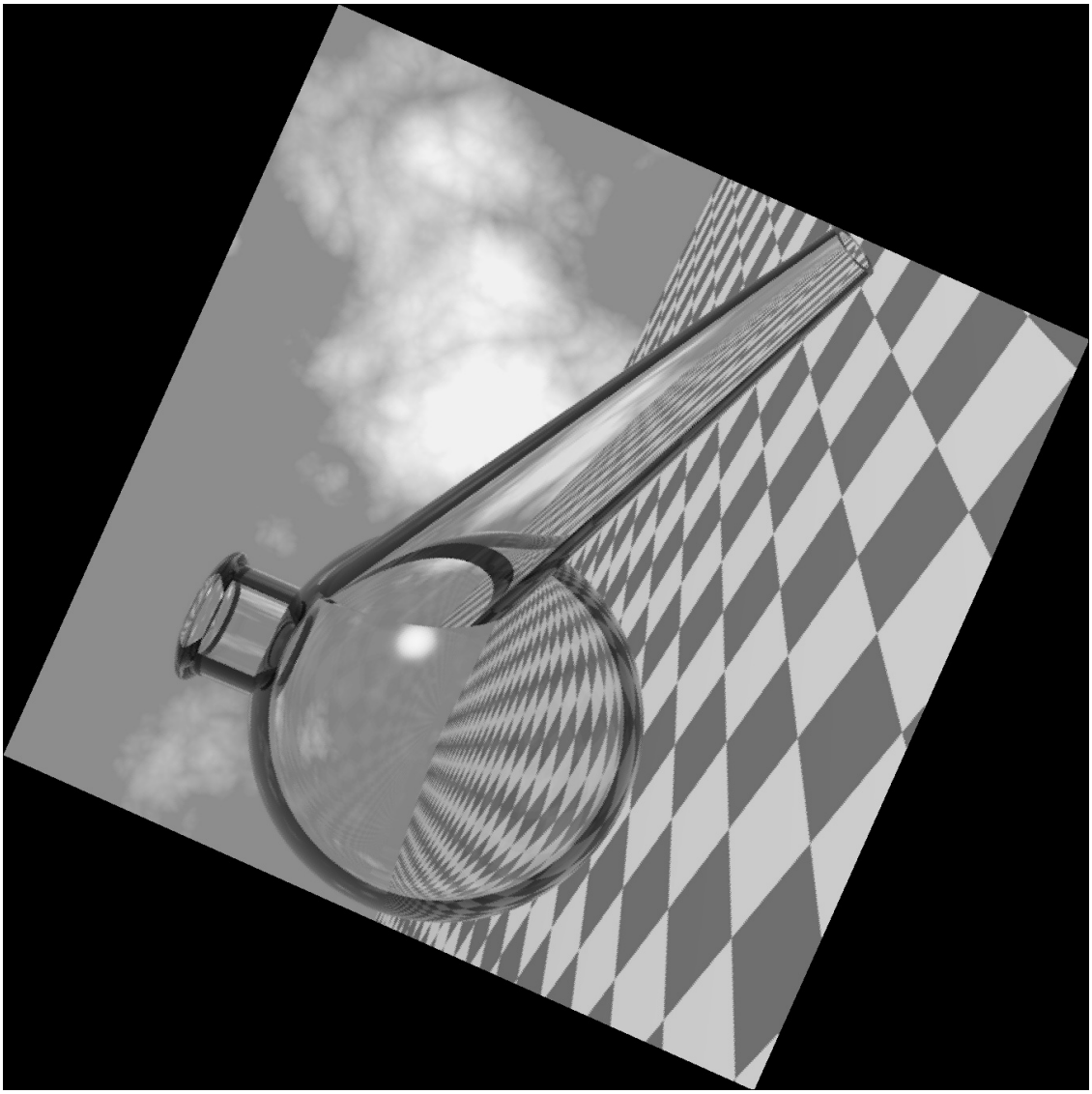


Figure 74: Rotate using bilinear

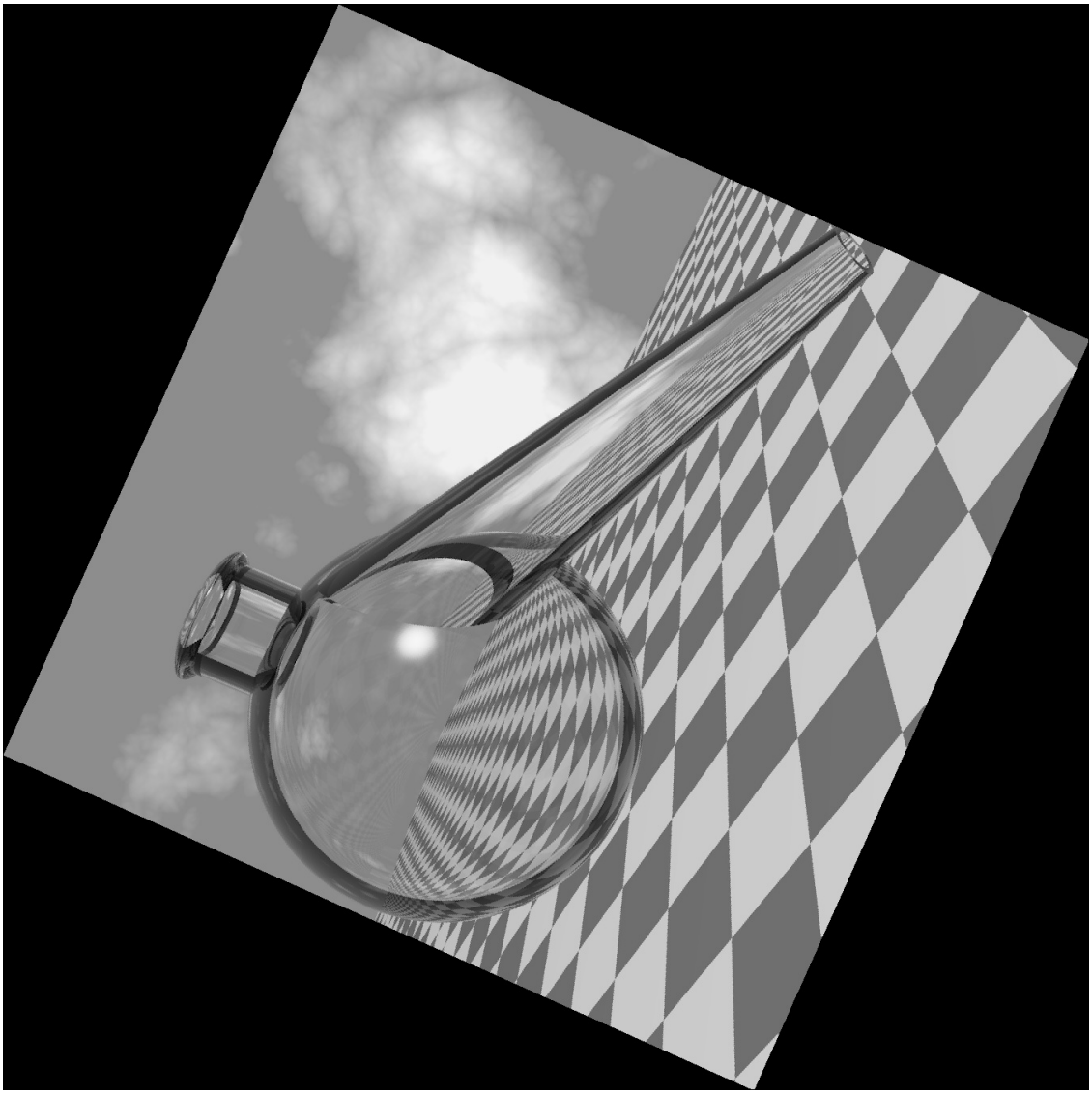


Figure 75: Rotate using nearest neighbour

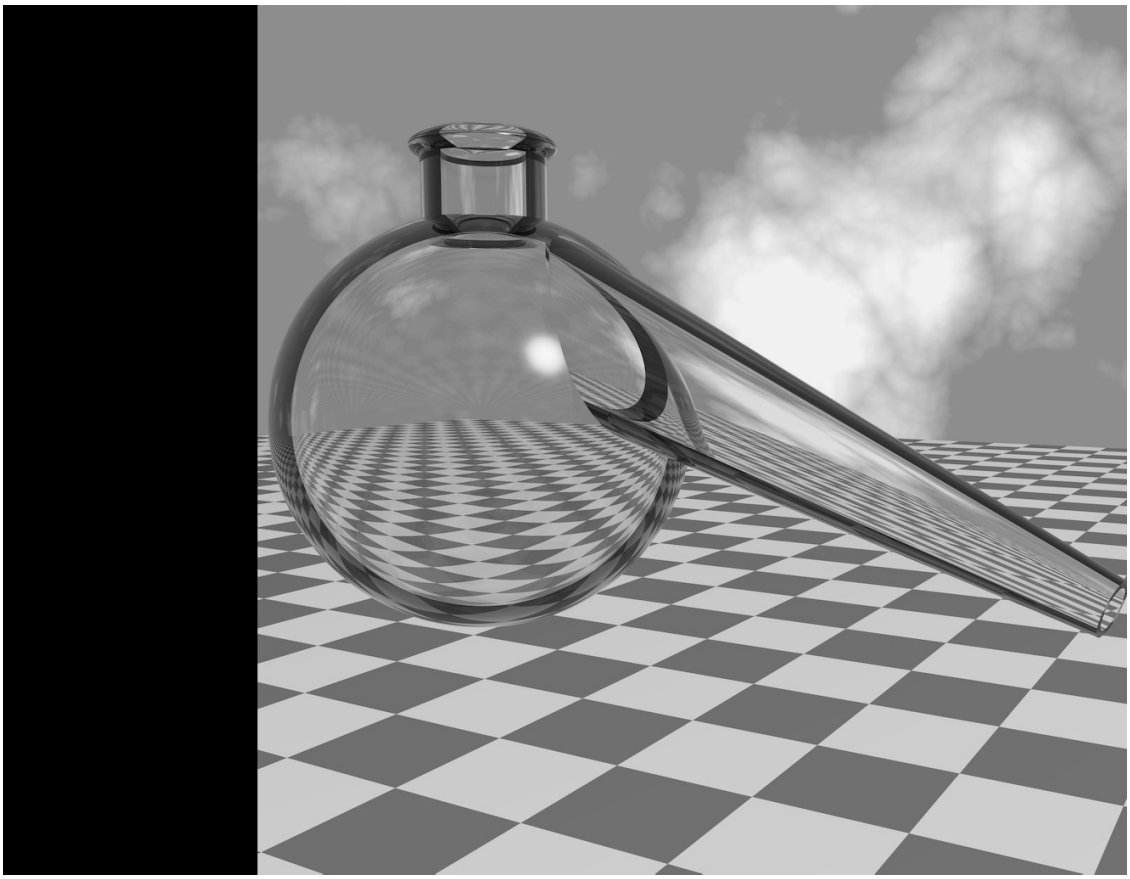


Figure 76: Right translate

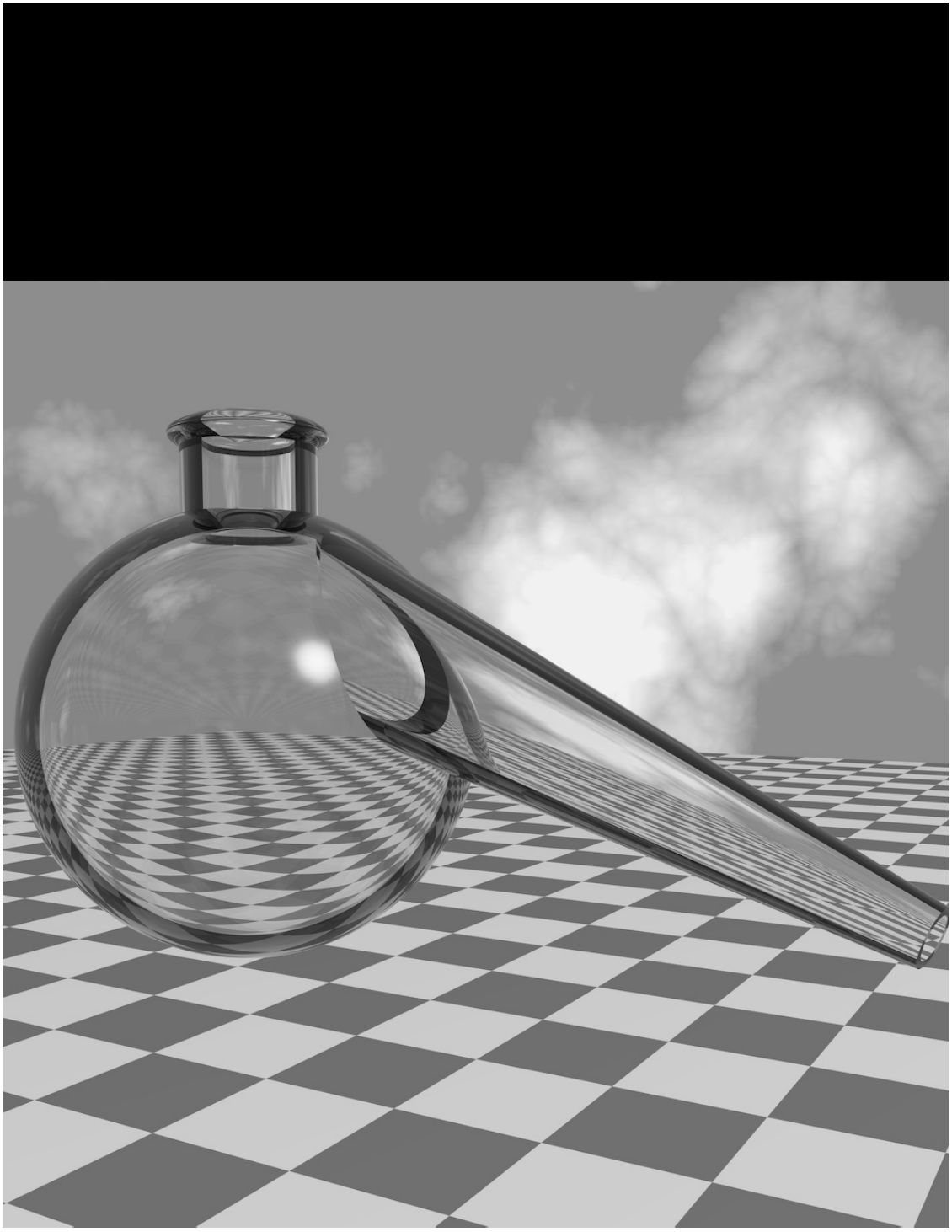


Figure 77: Down translate

Problem 8: Morpholigical Processing

Morpholigical Processing
(a) Implement the morphological operations: erosion, dilation, opening and closing, and use the noisy_fingerprint.tif to check your implementation.
(b) Implement boundary extraction, hole filling, connected component extraction. Using licoln_from_penny.tif, region_filling_refletion.tif and chickenfilet_with_bones.tif to verify the results, respectively.

Erosion and Dilation

Dilation

$$A \oplus B = \{z|(\hat{B}) \cap A \neq \emptyset\} = \{z|(\hat{B}_z \cap A \subseteq A)\}$$

- Set B: A structural elements
- Realtion to Convolution mask:
  - Flipping
  - Overlapping
- Other names: Grow, Expand

Figure 78: Noisy image

Figure 79: Dilated image

Erosion

$$A \odot B = \{z|(B)_z \subseteq A\}$$

- Set B: A structural elements
- Other banes: Shrink, Reduce

Opening and closing

Opening

A erosion followed by a dilation using the same structuring element for both operations.

$$A \circ B = A \ominus B \oplus = \cup(B_z)|(B_z) \subset A$$

Figure 80: Noisy image

Figure 81: Eroded image

Figure 82: Noisy image

Figure 83: Open of noisy image

Closing

A dilation followed by a erosion using the same structuring element for both operations.

$$A \circ B = (A \oplus B) \ominus B$$

Figure 84: Noisy image

Figure 85: Close of noisy image

Boundary extraction

$$\beta(A) = A - (A \ominus B)$$

Figure 86: A simple image, with 1's represented in white

Figure 87: Result of boundary extraction

Hole filling

Start from p inside boundary.

$$X_0 = p$$

$$X_k = (X_k \oplus B) \cap A^c$$

$$Until : X_(k+1) = X_k$$

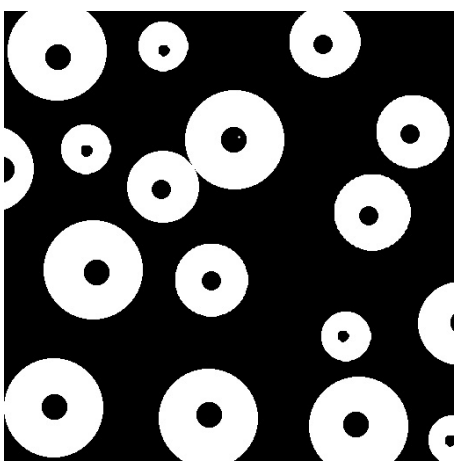


Figure 88: Binary image

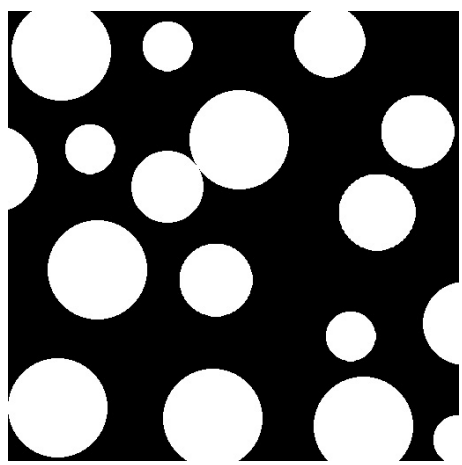


Figure 89: Result of filling that region

Connected component extraction

Srart from p belong to desired region.

$$X_0 = p$$

$$X_K = (X_k \oplus B) \cap A$$

$$Until : X_{k+1} = X_k$$

Number of pixes in each connected components is [1131, 784, 274, 53, 48, 35, 68, 52, 1, 112, 54].

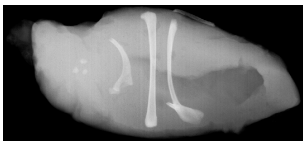


Figure 90: X-ray image

Figure 91: Thresholded image

Figure 92: Image eroded with 5×5 structuring elements

Problem 9: Image segmentation

Image segmentation

(a). Develop a program to implement the Roberts, Prewitt, Sobel, the Marr-Hildreth and the Canny edge detectors. Use the image building.tif to test your detectors. (For technique details of Marr-Hildreth and Canny, please refer to pp.736-747 (3rd edition, Gonzalez DIP) or MH-Canny.pdf at the same address of the slides.)

(b). Develop a program to implement the Otsus method of thresholding segmentation, and compare the results with the global thresholding method using test image polymersomes.tif. (For technique details, please refer to pp.763-770 (3rd edition, Gonzalez DIP), or Otsu.pdf at the same ftp address of slides.)

Dectors

Roberts

$$g_x = z_9 - z_5$$

$$g_y = z_8 - z_6$$

Figure 93: Roberts g_x

Prewitt

$$g_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

Figure 94: Roberts g_y

Figure 95: Roberts $|g_x| + |g_y|$

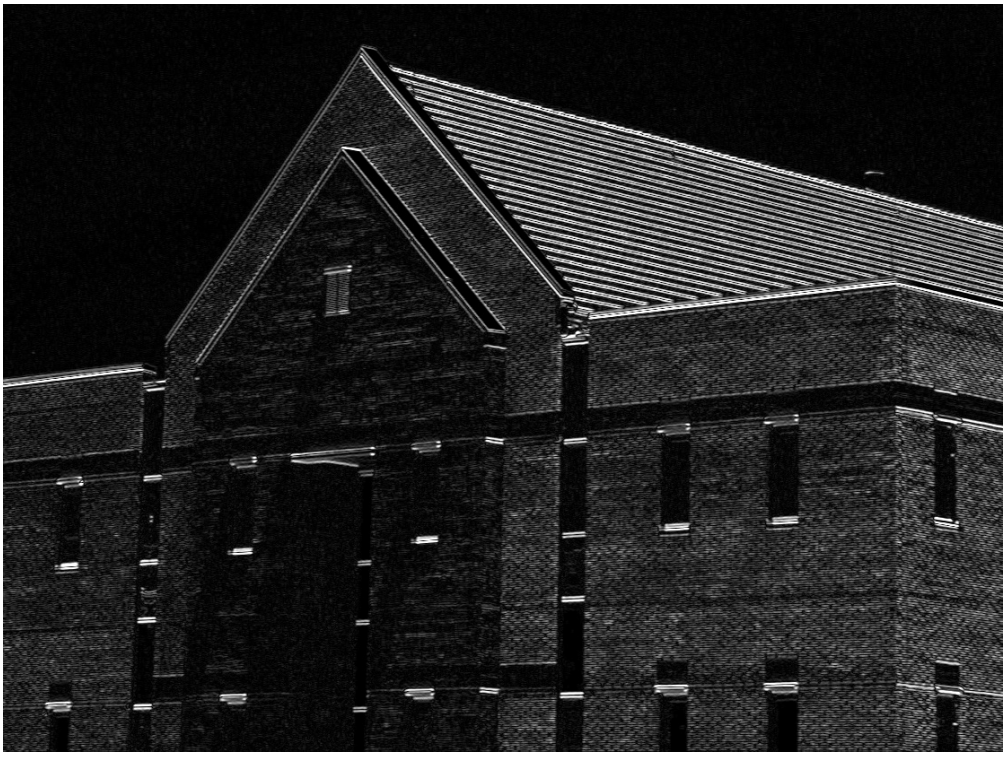


Figure 96: Prewitt g_x

Figure 97: Prewitt g_y

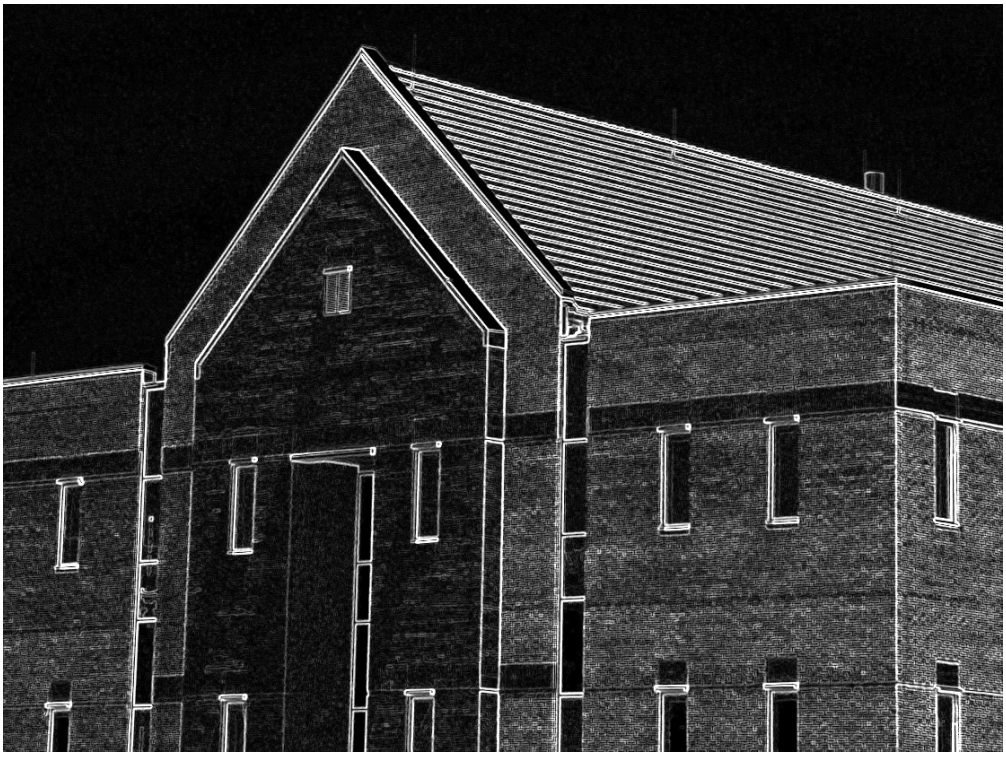


Figure 98: Prewitt $|g_x| + |g_y|$

Sobel

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

marr-Hildreth

For Marr-Hildreth, I firstly calculate the Laplacian of Gaussian (LoG)

$$LoG(x, y) = [\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Then we use GLPF to process the image. And we can judge that if the point has both positive and negative values in its neighbors, it is sure that there exists zero point. So here we find the zero crossings of the image and set them as the max value.

Canny

For Canny algorithm, I smoothed the input image with a Gaussian filter. Then I used the sobel edge detection with the row edge and col edge. At last I made the nonmaxima suppression.

Thresholding segmentation

Global threshold

In global threshold method, I firstly set a global threshold T. Then I generated two pixel vectors separated by the T. Third, I calculated their means and a new threshold $T^{'}$ using the mean of the two value; Finally,

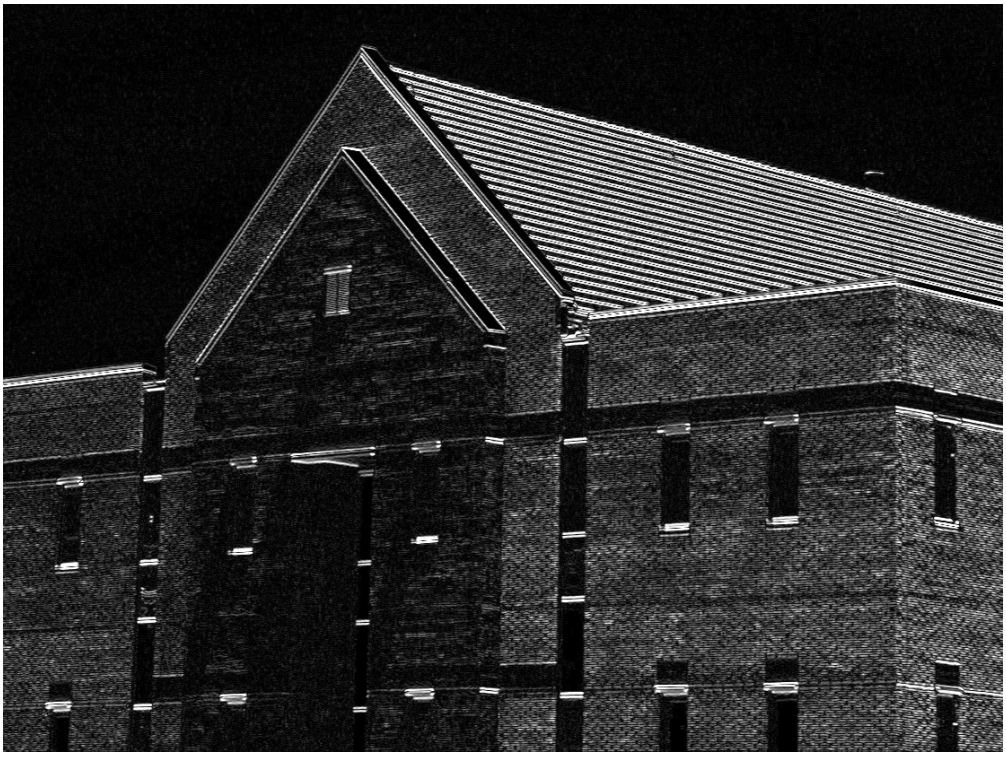


Figure 99: Sobel g_x

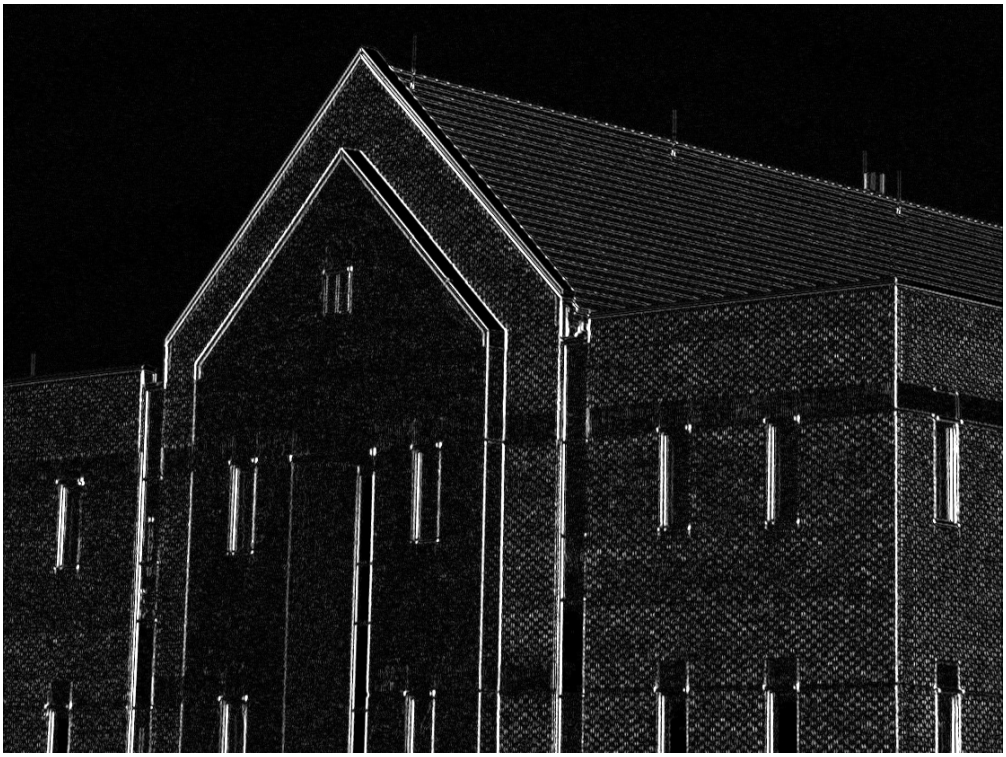


Figure 100: Sobel g_y

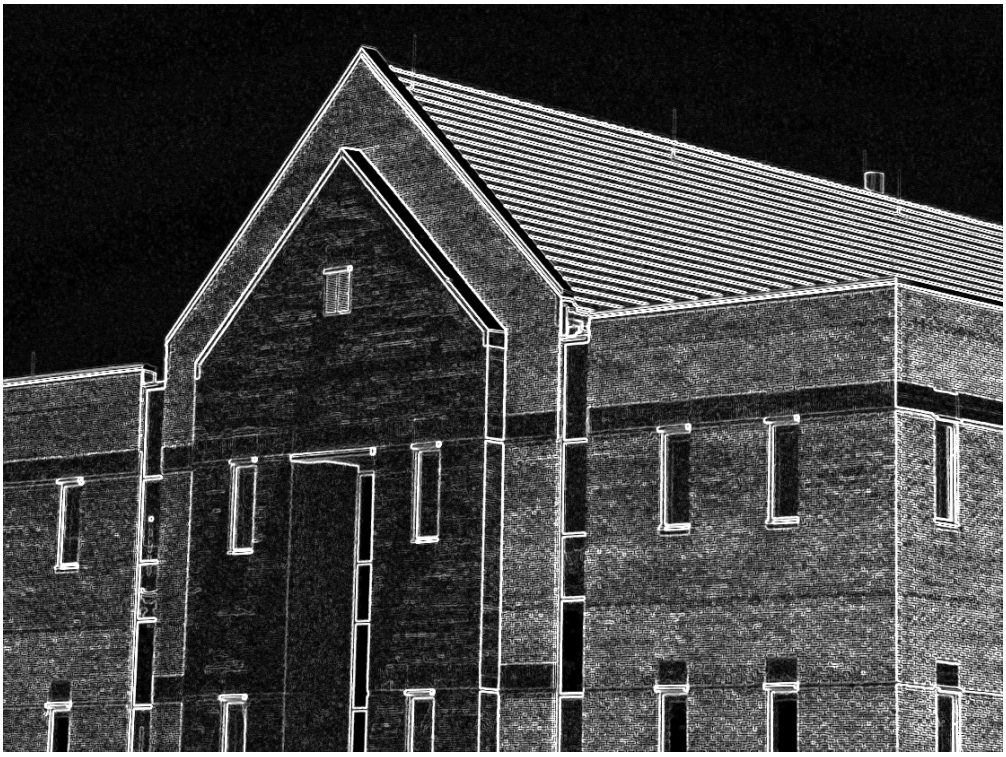


Figure 101: Sobel $|g_x| + |g_y|$

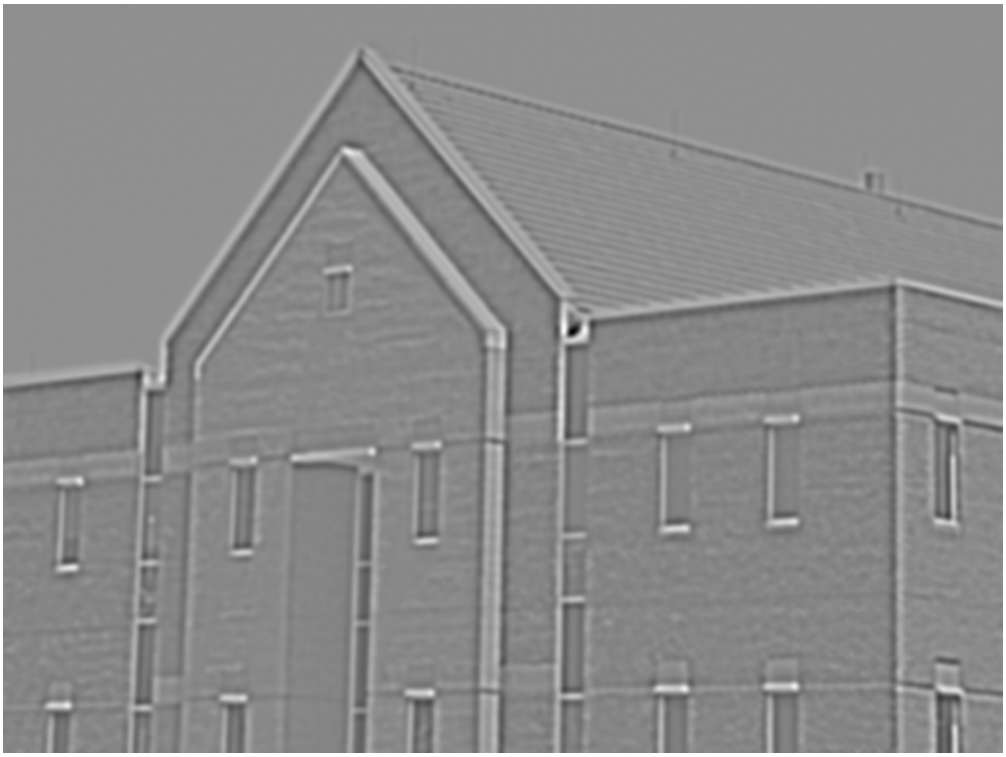


Figure 102: Gaussian image

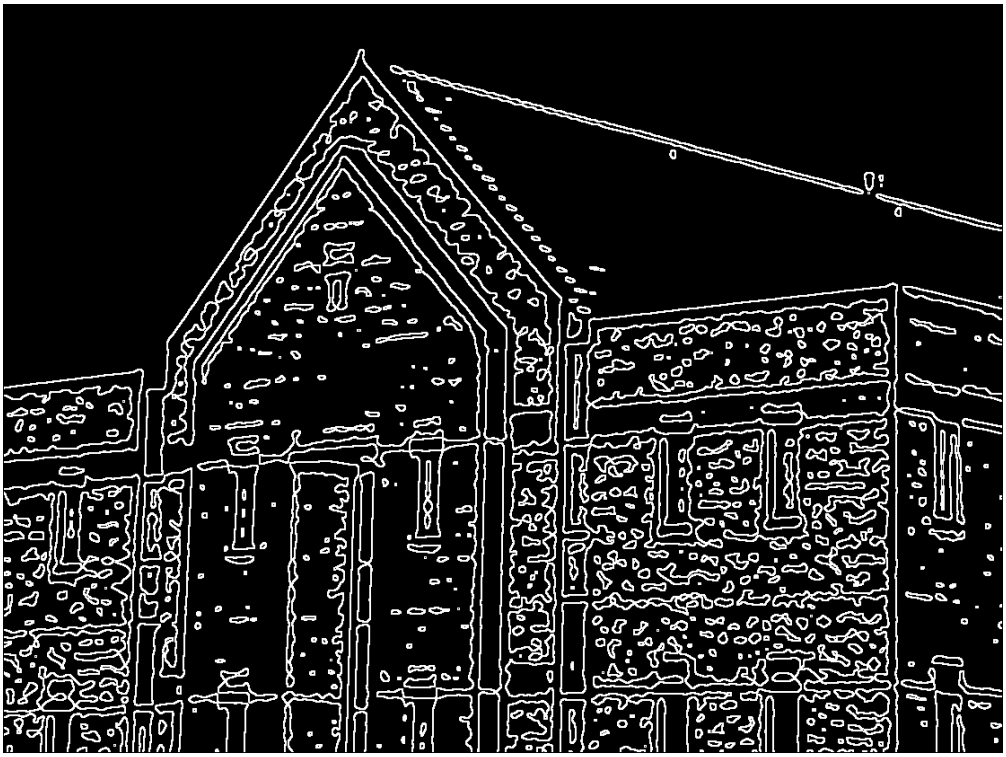


Figure 103: MarrHildreth image

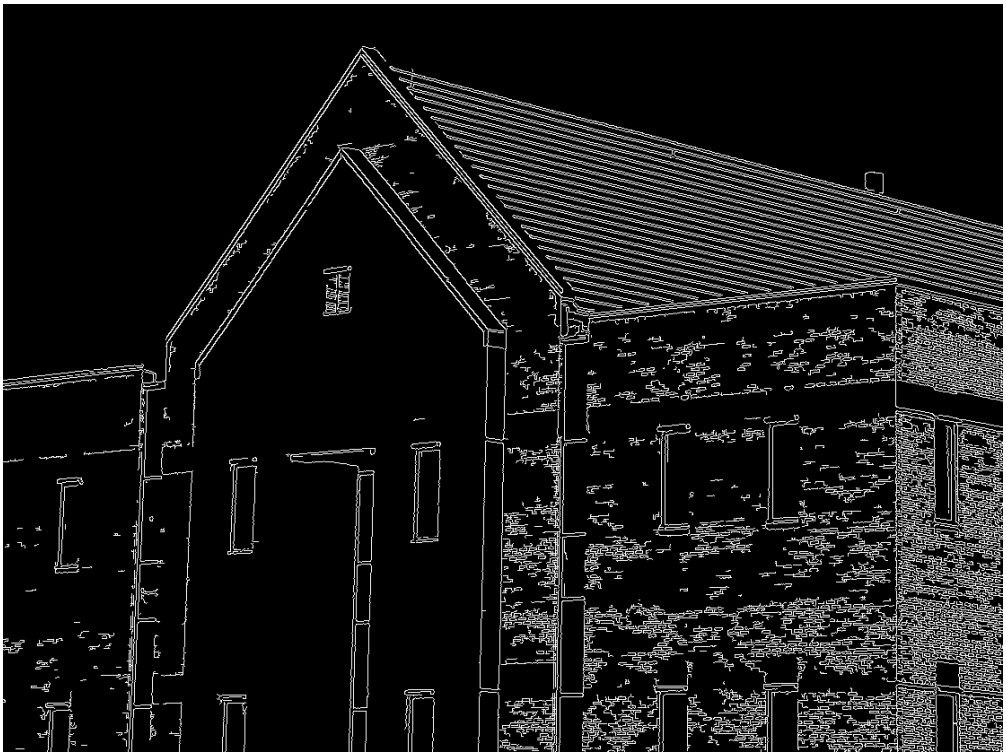


Figure 104: Canny image

repeat these operations until $|TT^{'}|$ is less than a value set previously.

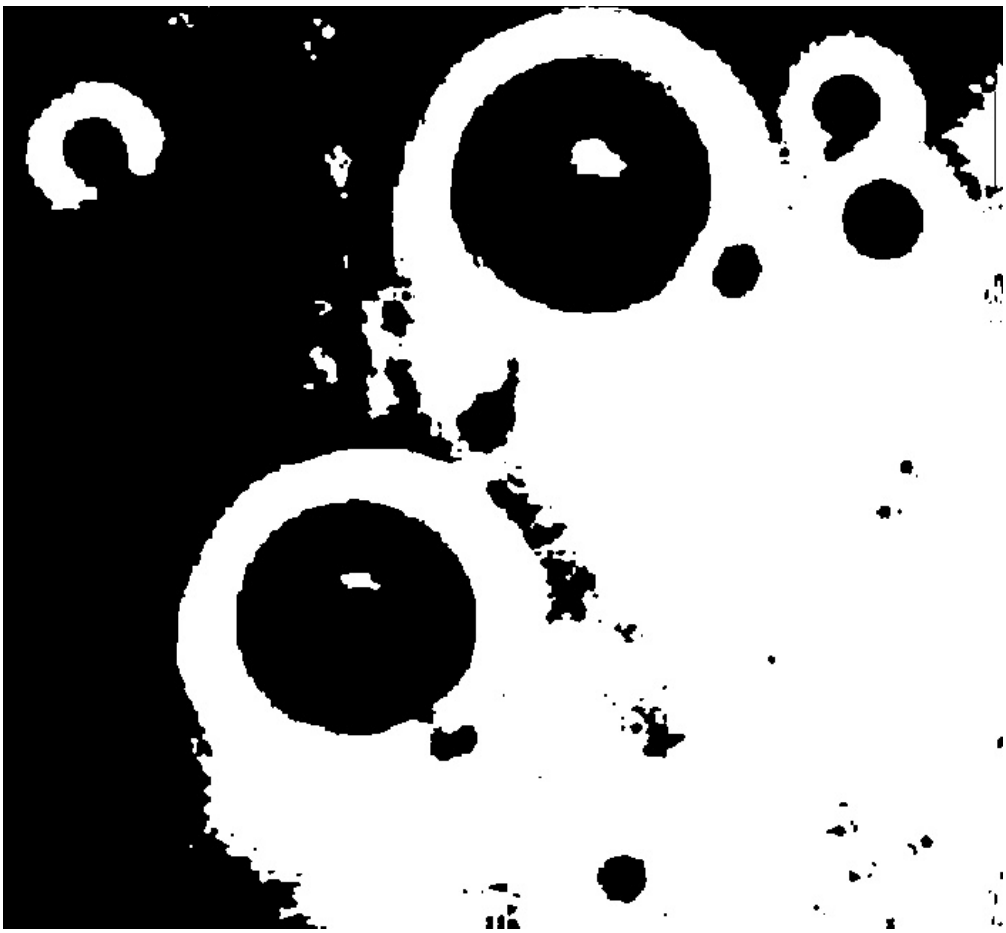


Figure 105: Global thresholding segmentation image

Ostu

In Ostu method, our goal is to find a threshold k to maximize the $\sigma_B^2(k)$, where

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

Firstly, I calculated the histogram normalization to get p_i for each pixel. Then I calculated $P_1(k) = \sum_{i=1}^{k} p_i$ and $m(k) = \sum_{i=0}^{k} ip_i$. Then we can get $m_G = m(L)$. At last, we calculate the $\sigma^2(k)$ and find the maximum.

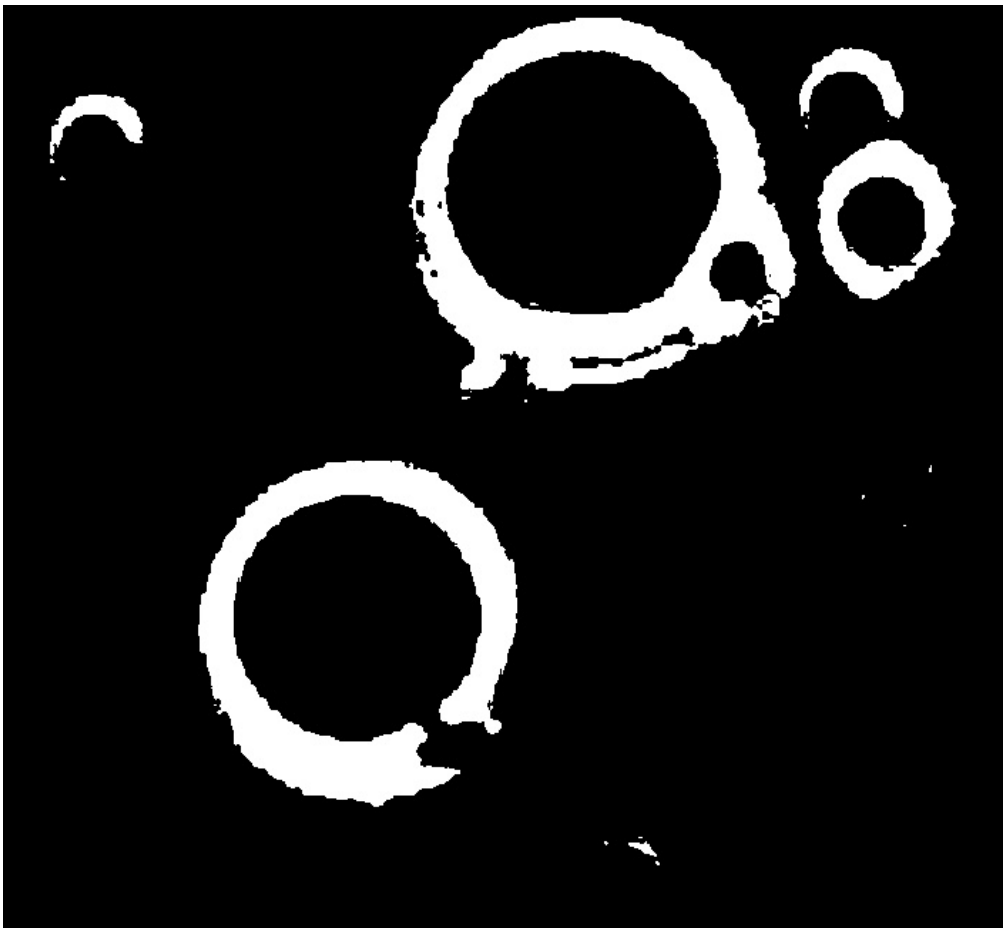


Figure 106: Ostu thresholding segmentation image