Thierry Cantenot

Report

Digital Image Processing

« Assignments »

2014-2015

Summary

A. Histogram Equalization

A.1 Problem statement

1. Write a computer program for computing the histogram of an image.
2. Implement the histogram equalization technique.
3. Your program must be general to allow any gray-level image as its input.

A.2 Python implementation

Usage : **python problem1.py [-h] image_path**

A.3 Figure 1

A.3.1 Histogram

Original image : A.1 | Original image's histogram : A.2

A.3.2 Histogram equalization

Enhanced image : A.3 | Enhanced image's histogram : A.4

A.4 Figure 2

A.4.1 Histogram

Original image : A.5 | Original image's histogram : A.6

A.4.2 Histogram equalization

Enhanced image : A.7 | Enhanced image's histogram : A.8

FIGURE A.1 – Original *Fig1.jpg*

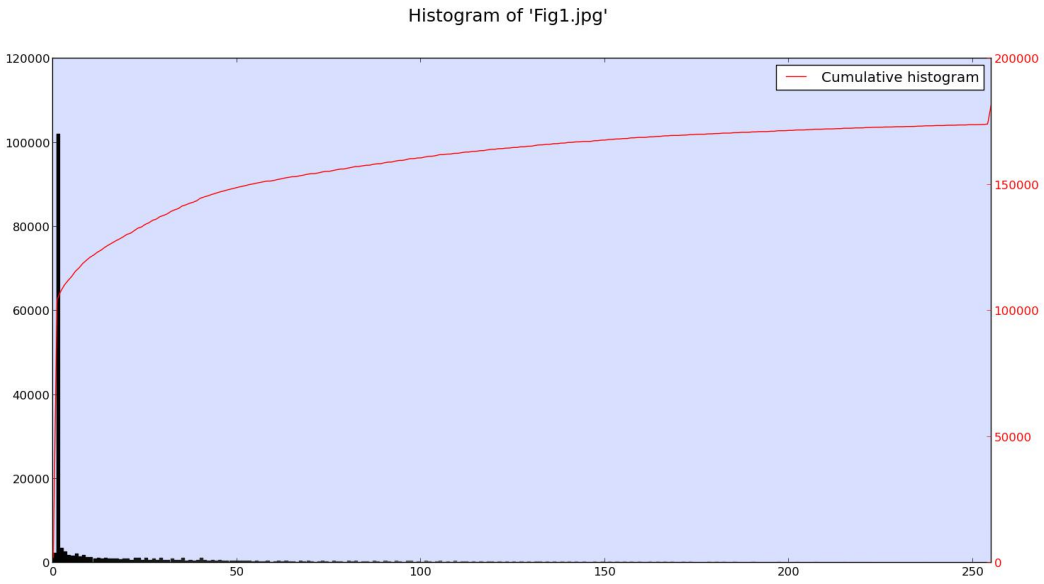

FIGURE A.2 – Histogram of *Fig1.jpg*

FIGURE A.3 – Enhanced *Fig1.jpg*

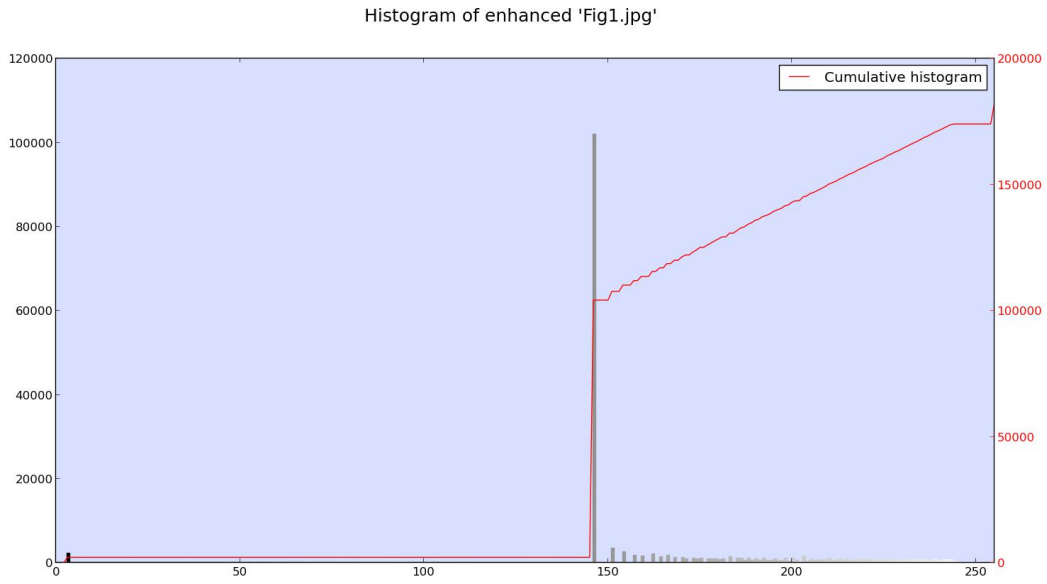

FIGURE A.4 – Equalized histogram of *Fig1.jpg*

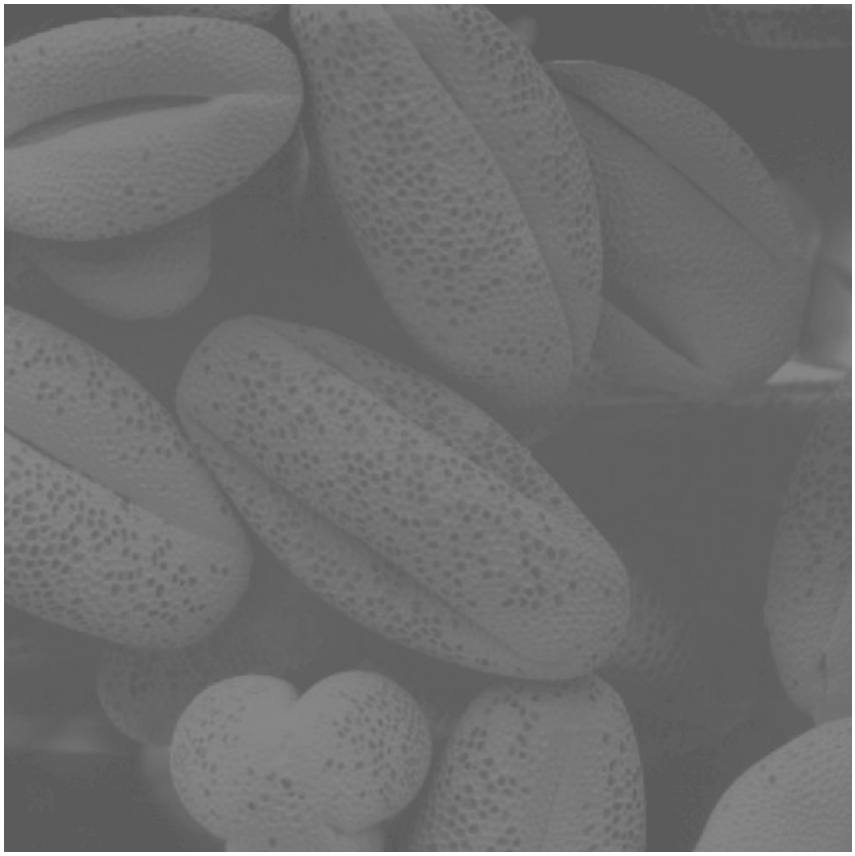

FIGURE A.5 – Original *Fig2.jpg*

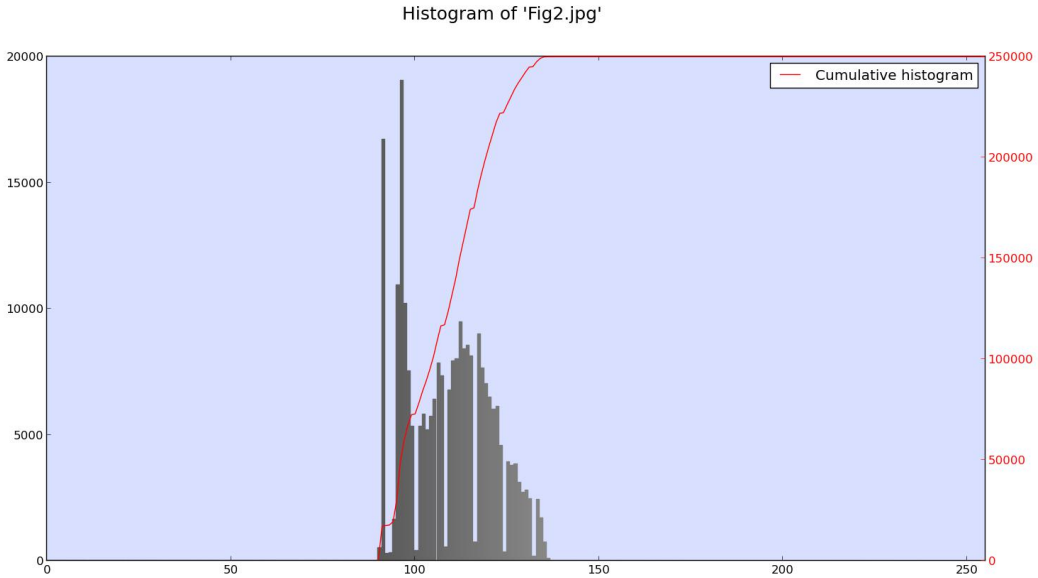

FIGURE A.6 – Histogram of *Fig2.jpg*

FIGURE A.7 – Enhanced *Fig2.jpg*

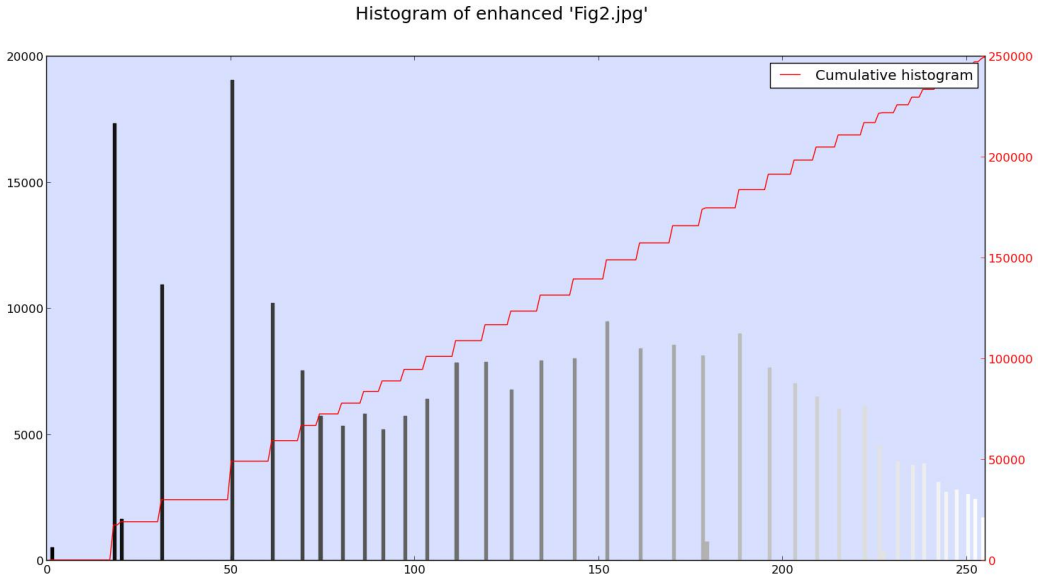

FIGURE A.8 – Equalized histogram of *Fig2.jpg*

B. Spatial enhancement methods

B.1 Problem statement

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 cannot directly use functions of Matlab such as imfilter or fspecial, implement all functions by yourself).

B.2 Python implementation

Usage : **python problem2.py [-h] [–laplacian] [–sobel] [-a A] [-g G] [-c C] image_path**

For example, to run the full image enhancement described in the assignment, using a 3×3 Laplacian filter with A = 1.7, then a Sobel, a smoothing filter and a Power-Law transformation with c = 1 and gamma = 0.5 type :

python problem2.py –laplacian -a 1.7 –sobel -g 0.5 -c 1 skeleton_orig.tif

B.3 Results

B.3.1 Original image

FIGURE B.1 – Original *skeleton_ orig.tif*

B.3.2 3x3 Laplacian (A = 0)

FIGURE B.2 – Laplacian (A=0)

FIGURE B.3 – Sharpened image

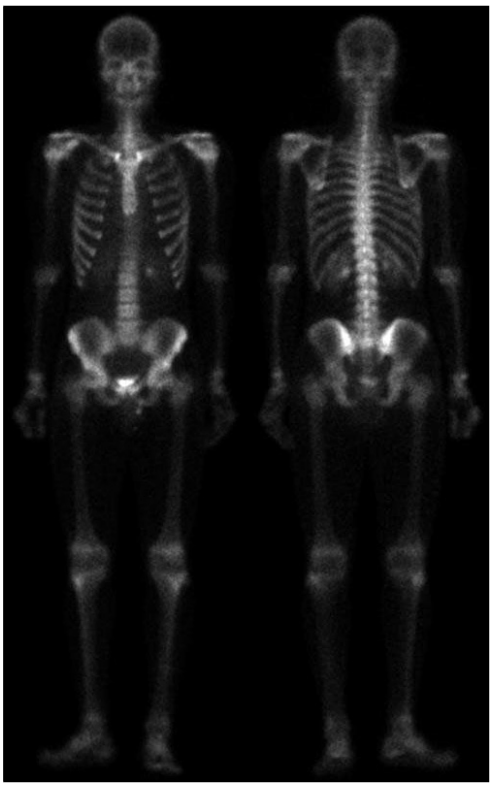

FIGURE B.4 – Original image

B.3.3 3x3 Laplacian (A = 1)

FIGURE B.5 – Laplacian (A=1)

FIGURE B.6 – Sharpened image

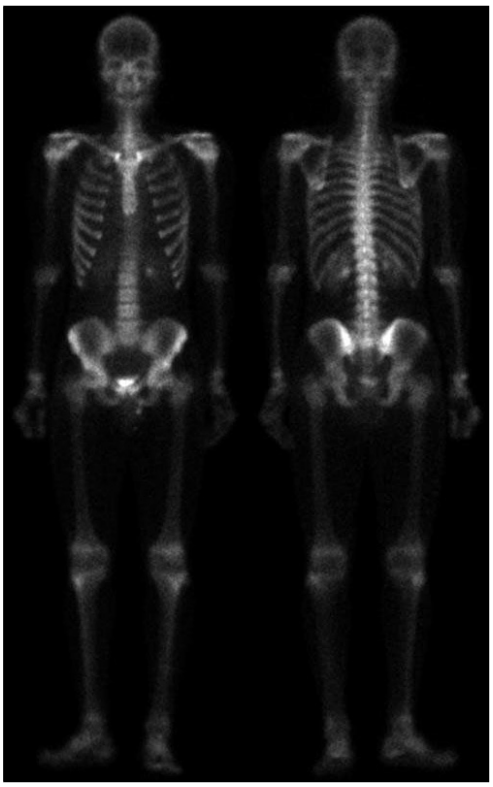

FIGURE B.7 – Original image

B.3.4 3x3 Laplacian (A = 1.7)

FIGURE B.8 – Laplacian (A=1.7)

FIGURE B.9 – Sharpened image

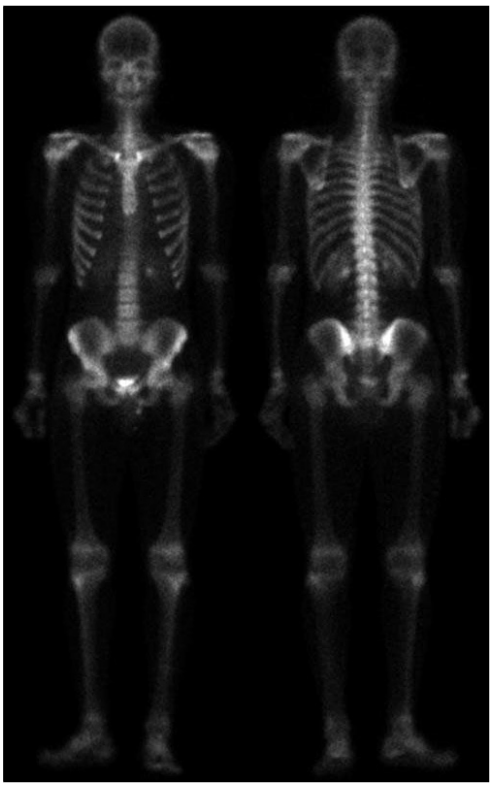

FIGURE B.10 – Original image

B.3.5 Sobel

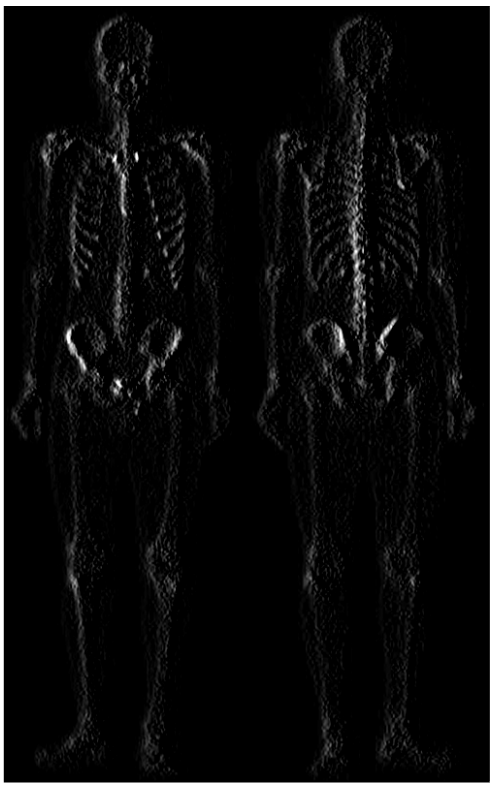

FIGURE B.11 – Sobel x-gradient

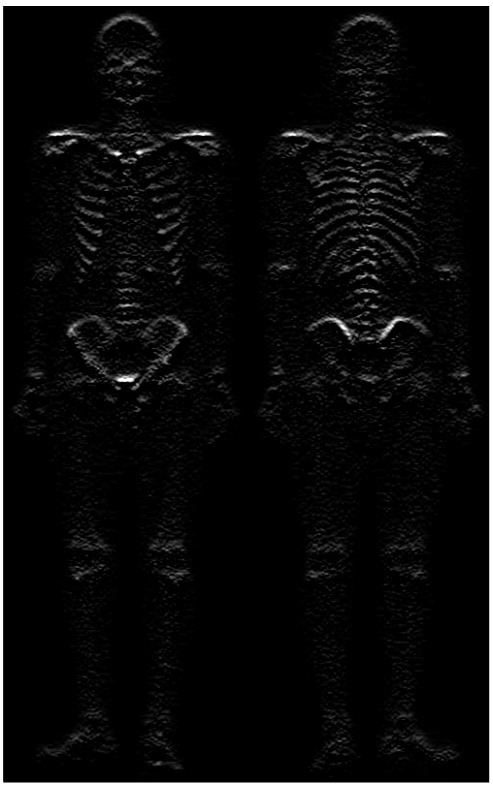

FIGURE B.12 – Sobel y-gradient

FIGURE B.13 – Sobel image

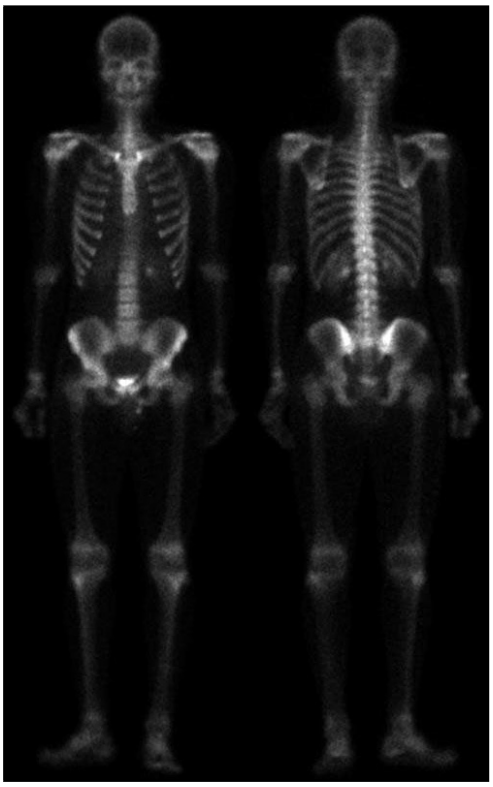

FIGURE B.14 – Original image

B.3.6 Smoothing, Sharpening and Power-Law transformation

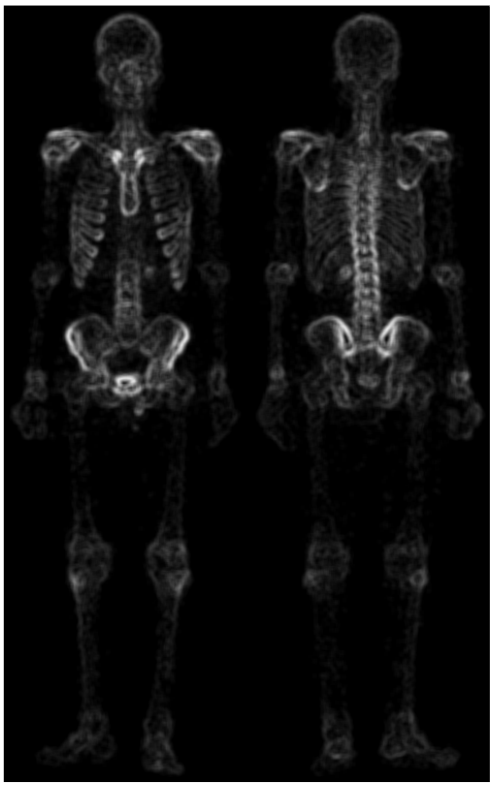

FIGURE B.15 – Smooth Sobel

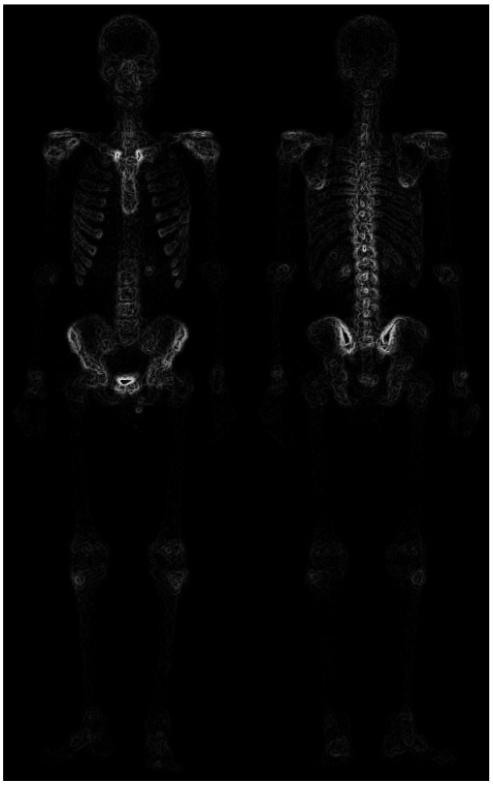

FIGURE B.16 – Laplacian x Sobel

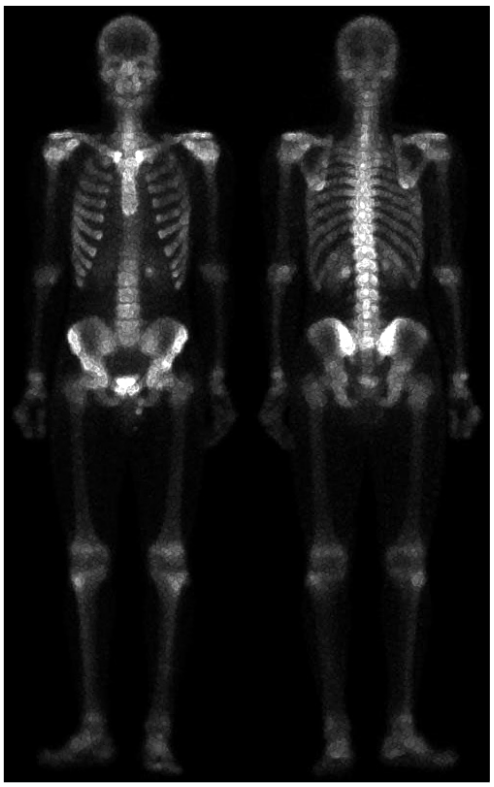

FIGURE B.17 – Original + Laplacian x Smooth Sobel

FIGURE B.18 – Final image (after Power-Law (c=1, g=0.5))

B.3.7 Comparison

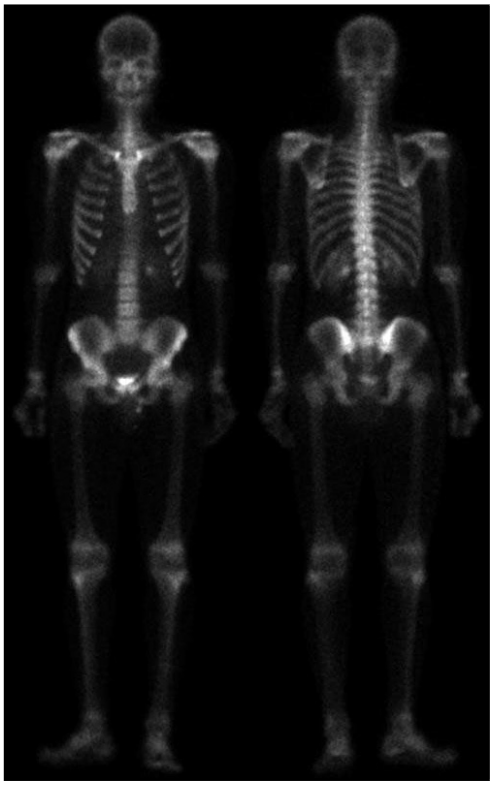

FIGURE B.19 – Original

FIGURE B.20 – Final image

C. Filtering in frequency domain

C.1 Problem statement

Implement the ideal, Butterworth and Gaussian lowpass and highpass filters and test them under different parameters using *characters_ test_ pattern.tif*.

C.2 Python implementation

Usage : **python problem3.py [-h] [–ideal] [–butterworth] [–gaussian] (–low | –high) [–npp] [-d D] [-n N] image_path**

Use **python problem3.py -h** to see the help.

C.3 Results

C.3.1 Original image

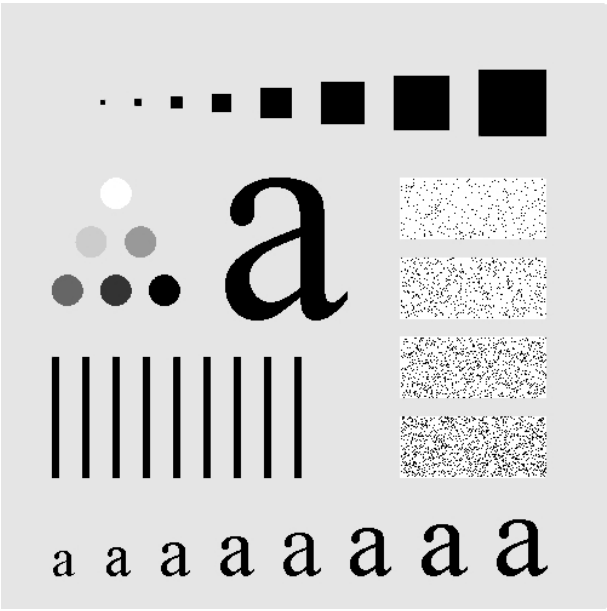

FIGURE C.1 – Original *characters_ test_ pattern.tif*

C.3.2 Ideal filter

Low pass

python problem3.py –ideal -d 5 –low characters_test_pattern.tif

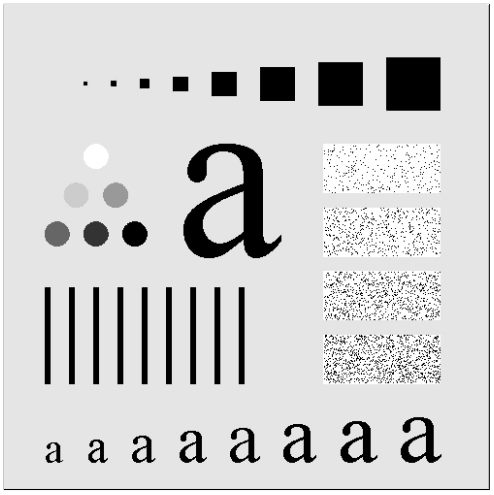

FIGURE C.2 – Original image

FIGURE C.3 – Ideal low 5

FIGURE C.4 – Ideal low 15

FIGURE C.5 – Ideal low 30

High pass

python problem3.py –ideal -d 5 –high characters_test_pattern.tif

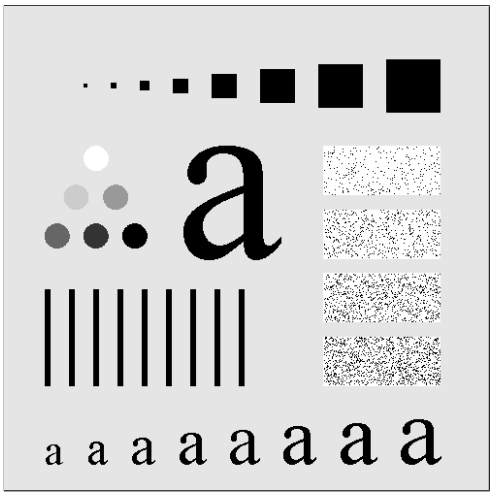

FIGURE C.6 – Original image

FIGURE C.7 – Ideal high 5

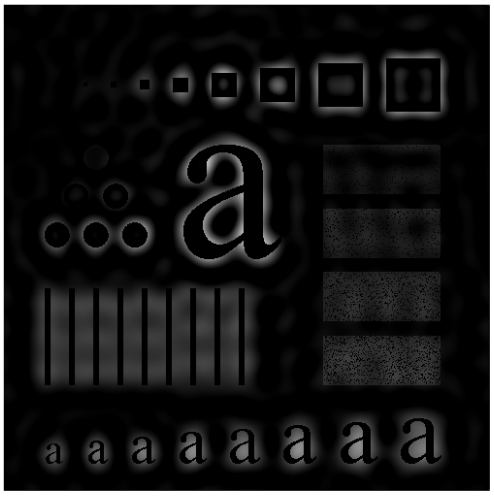

FIGURE C.8 – Ideal high 15

FIGURE C.9 – Ideal high 30

C.3.3 Butterworth order 2 filter

Low pass

python problem3.py –butterworth -d 5 -n 2 –low characters_test_pattern.tif

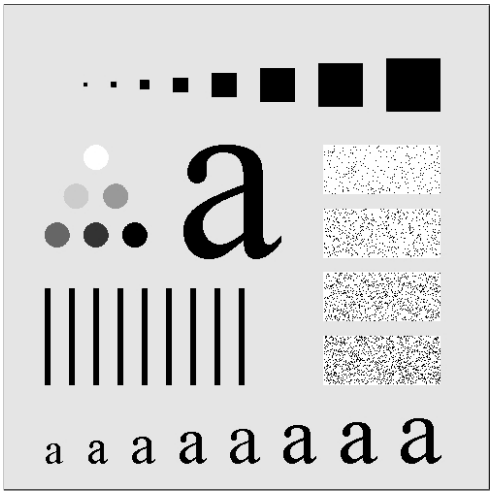

FIGURE C.10 – Original image

FIGURE C.11 – Butterworth low 5

FIGURE C.12 – Butterworth low 15

FIGURE C.13 – Butterworth low 30

High pass

python problem3.py –butterworth -d 5 -n 2 –high characters_test_pattern.tif

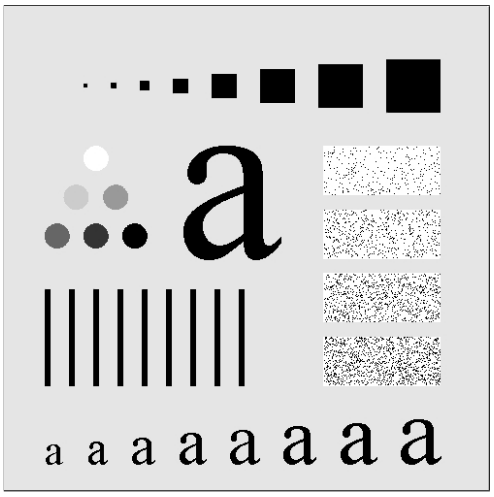

FIGURE C.14 – Original image

FIGURE C.15 – Butterworth high 5

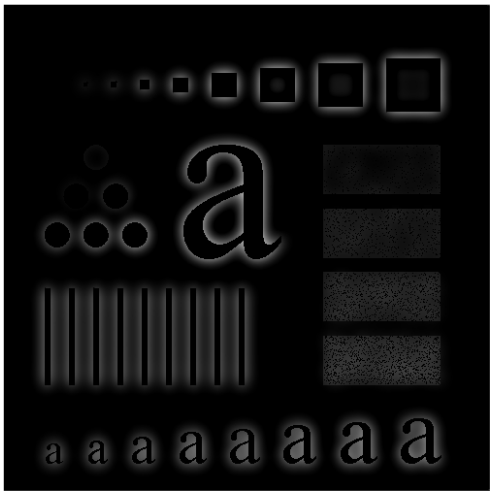

FIGURE C.16 – Butterworth high 15

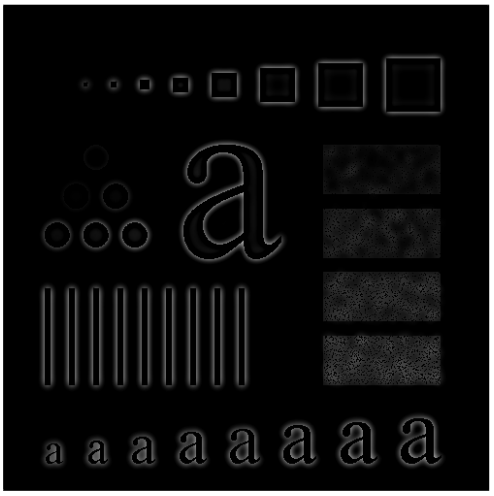

FIGURE C.17 – Butterworth high 30

C.3.4 Butterworth order 5 filter

Low pass

python problem3.py –butterworth -d 5 -n 5 –low characters_test_pattern.tif

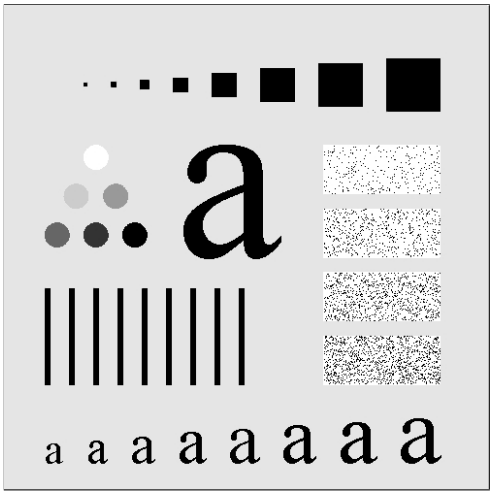

FIGURE C.18 – Original image

FIGURE C.19 – Butterworth low 5

FIGURE C.20 – Butterworth low 15

FIGURE C.21 – Butterworth low 30

High pass

python problem3.py –butterworth -d 5 -n 5 –high characters_test_pattern.tif

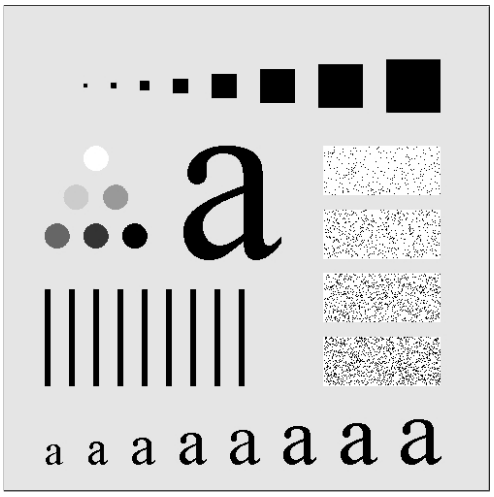

FIGURE C.22 – Original image

FIGURE C.23 – Butterworth high 5

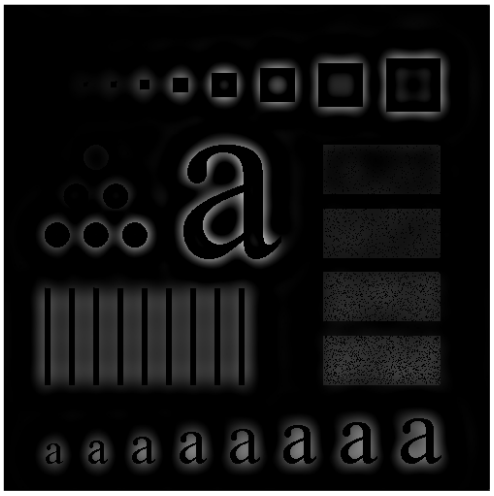

FIGURE C.24 – Butterworth high 15

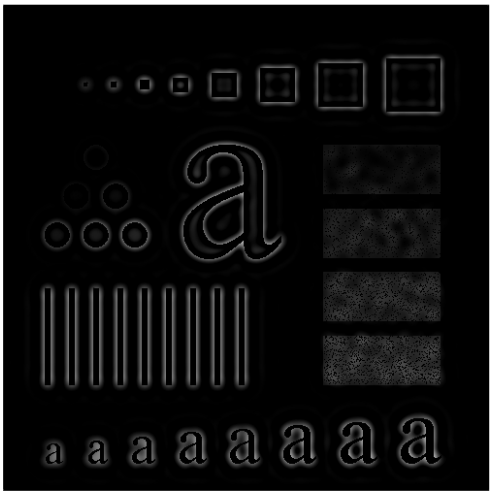

FIGURE C.25 – Butterworth high 30

C.3.5 Gaussian filter

Low pass

python problem3.py –gaussian -d 5 –low characters_test_pattern.tif

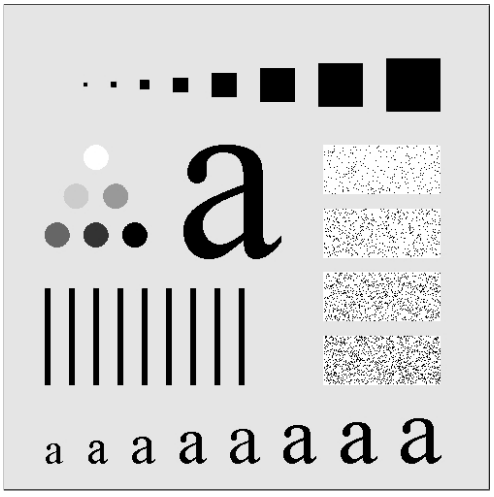

FIGURE C.26 – Original image

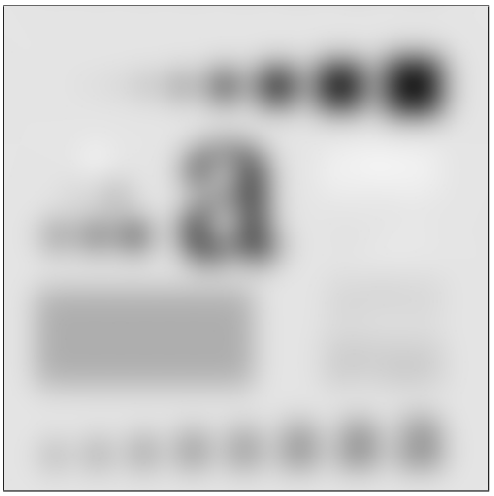

FIGURE C.27 – Gaussian low 5

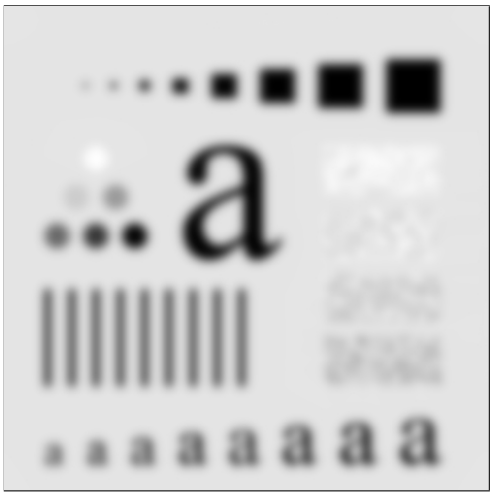

FIGURE C.28 – Gaussian low 15

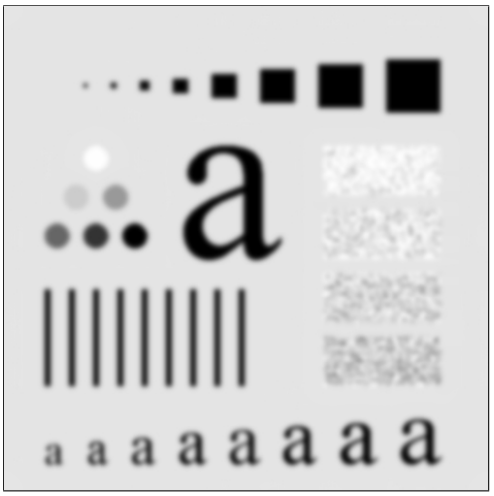

FIGURE C.29 – Gaussian low 30

High pass

python problem3.py –gaussian -d 5 –high characters_test_pattern.tif

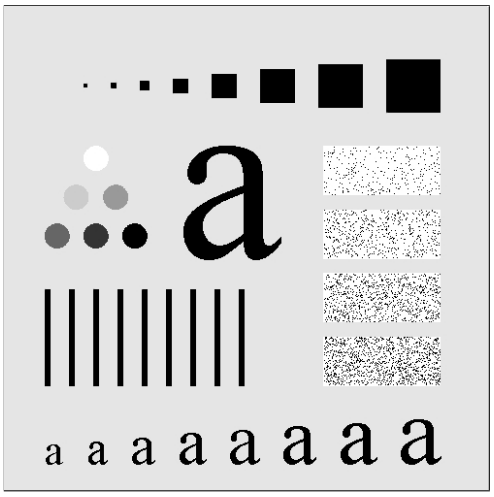

FIGURE C.30 – Original image

FIGURE C.31 – Gaussian high 5

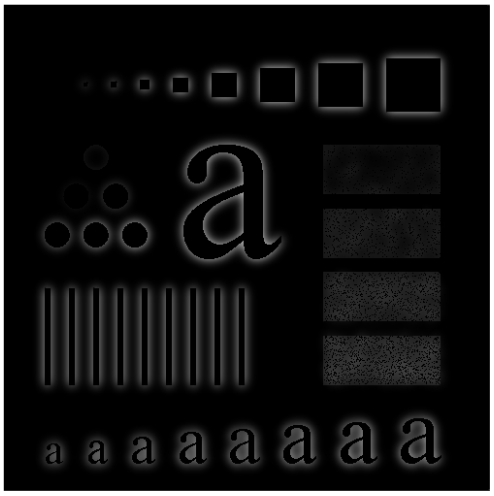

FIGURE C.32 – Gaussian high 15

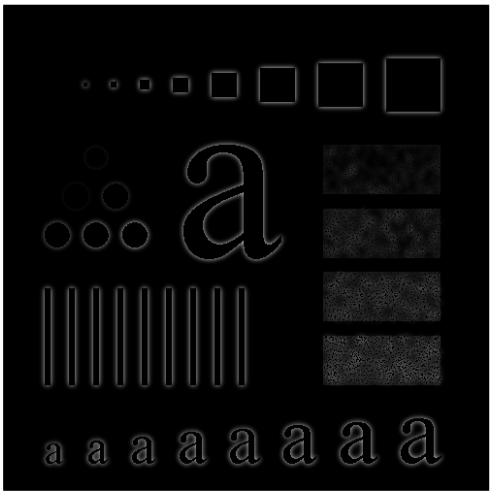

FIGURE C.33 – Gaussian high 30

D. Noise generation and noise reduction

D.1 Problem statement

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.
And then add some of these noises to the circuit image and investigate the different mean filters and order statistics as the textbook did at pages 344-352.