Report for
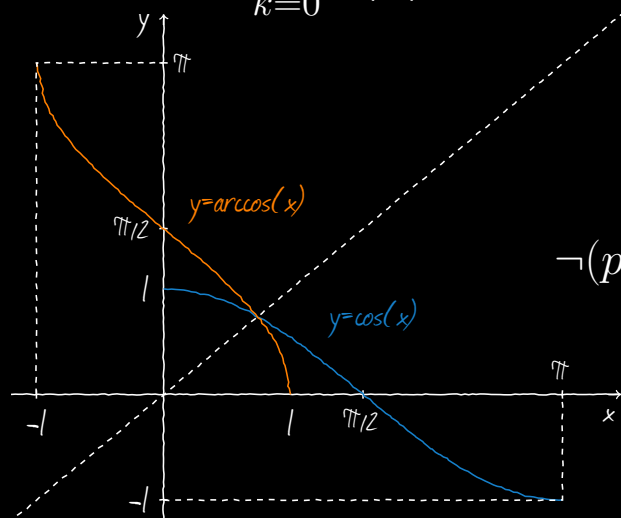# Digital Image Processing

Fanyong Xue

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

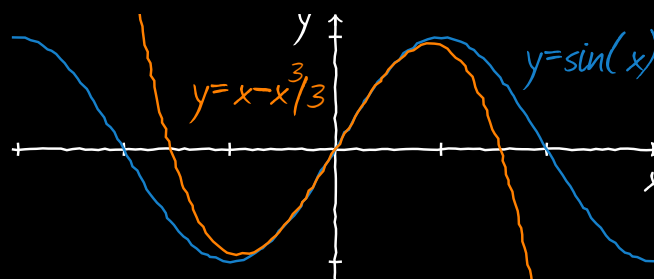$$\zeta_k = |a|^{1/n} e^{i(\arg(a) + 2k\pi)/n}$$

$$e^{i\pi} + 1 = 0$$

$$\neg(p \lor q) \equiv (\neg p) \land (\neg q)$$

$y = \arccos(x)$

$y = \cos(x)$

$y = x - x^3/3$

$y = \sin(x)$

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Contents

Contents

1 2 3 4 5 A

1.1 OverView

(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 transformation function, the enhanced image, and a plot of its histogram.

1.2 Generate the Histogram

1.2.1 Function

Generating the histogram of an image using following function:

$$H(i) = the\ number\ of\ pixel\ whose\ value\ euquals\ to\ i \qquad (1.1)$$

1.2.2 Histogram

The histogram pictures of Fig1.jpg and Fig2.jpg are listed as follows:
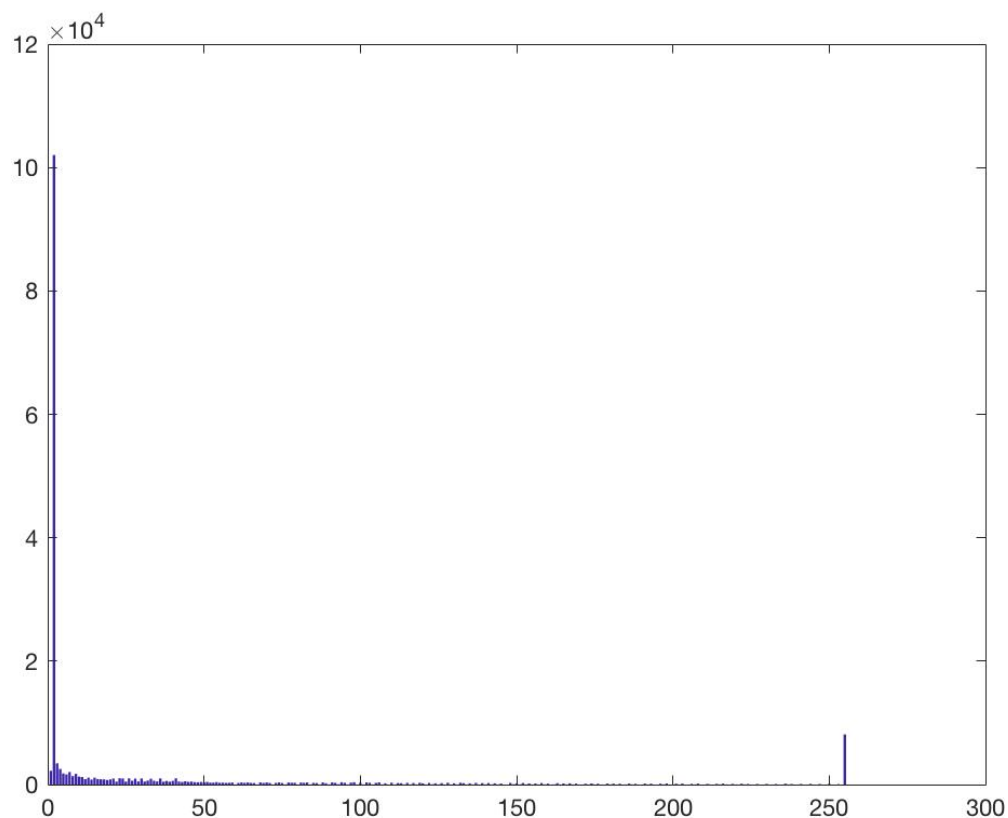
Figure 1.1: Histogram of fig1.jpg
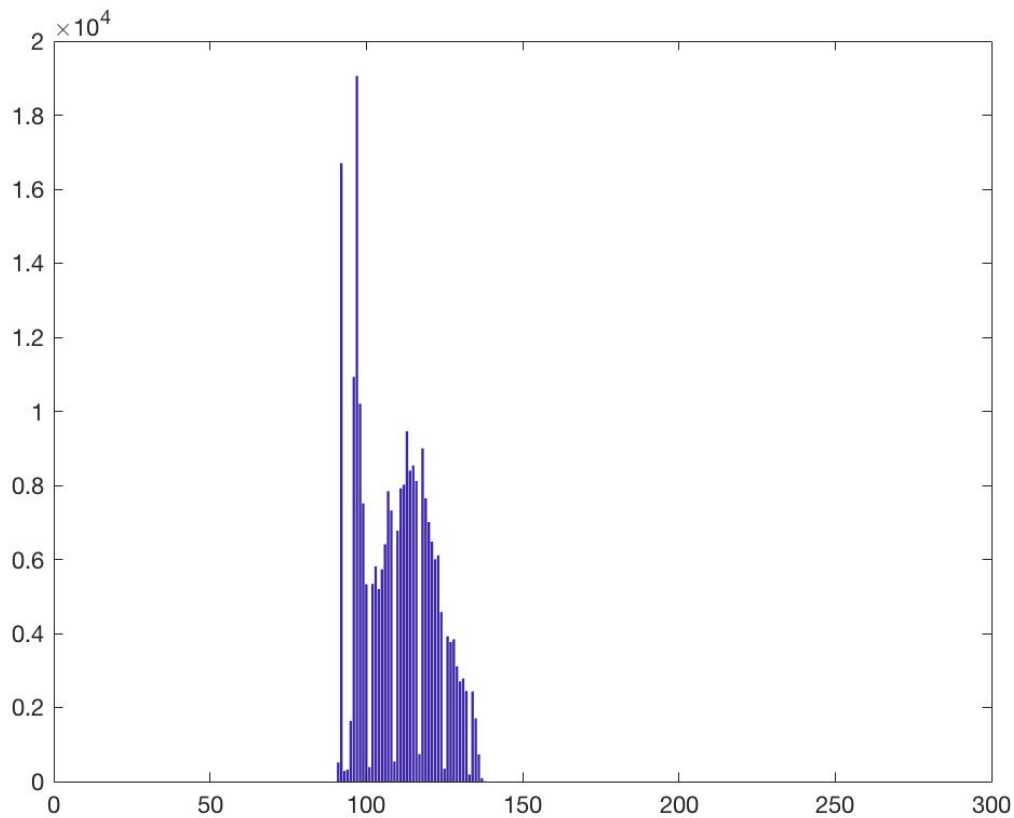
Figure 1.2: Histogram of fig2.jpg

1.3 Transfer Function

1.3.1 Implement the Histogram Equalization Technique

We use those functions to calculate the histogram equalization:

$$L = Max(image(r, c)) \ \forall r \in [1, rows] \ and \ \forall c \in [1, cols] \tag{1.2}$$

$$s(r_k) = L * T(r_k) = L * \sum_{j=0}^{k} P_r(r_j) = L * \sum_{j=0}^{k} \frac{n_j}{n} \tag{1.3}$$

1.3.2 Transfer Function

The transfer function of Fig1.jpg and Fig2.jpg are listed as follows:
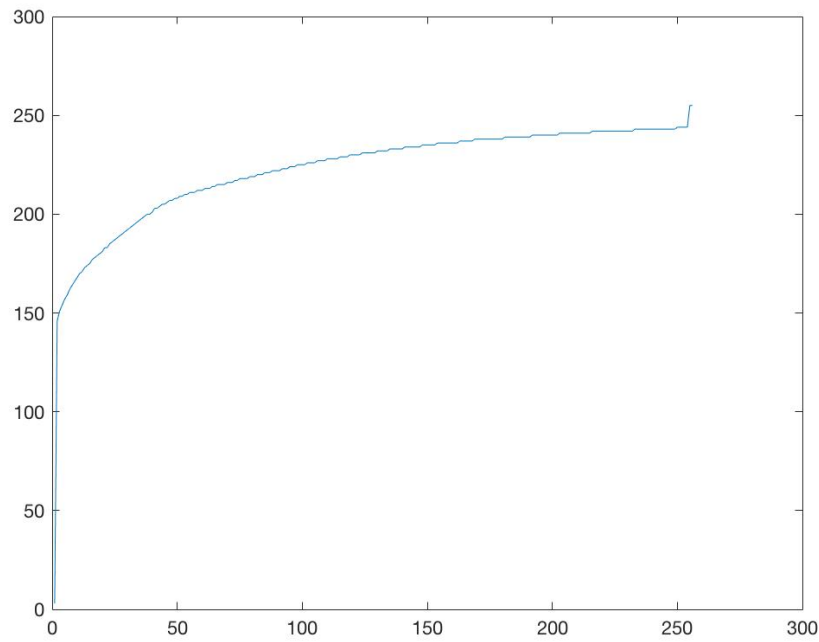
Figure 1.3: Transfer Function of fig1.jpg
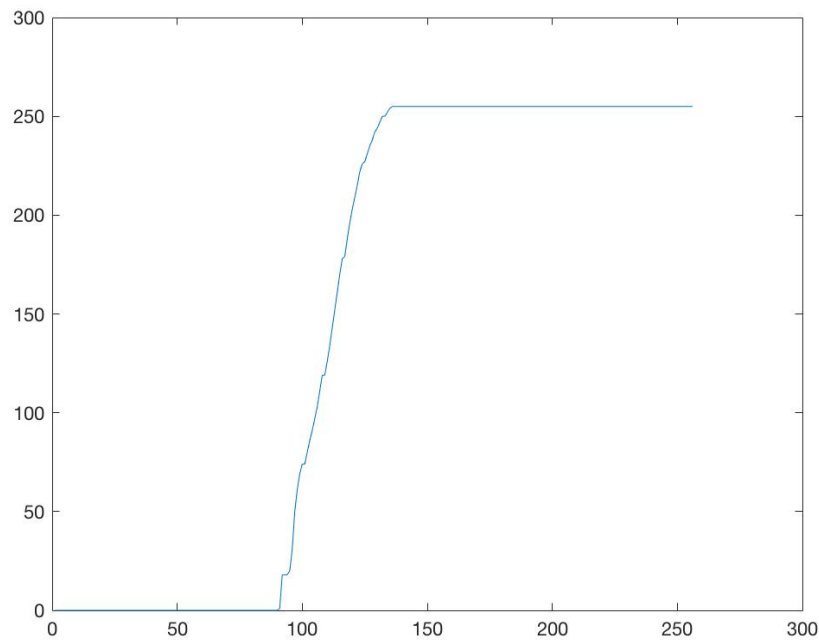
Figure 1.4: Transfer Function of fig2.jpg

1.4 Enhanced Images

1.4.1 Enhanced Function

We use the function

$$New\ Image(r, c) = Transfer\ Function(image(r, c))\ \forall r \in [1, rows]\ and\ \forall c \in [1, cols] \quad (1.4)$$

to enhance the original images.

1.4.2 Enhanced Images

The original images and enhanced images and histogram comparation are listed as follows.
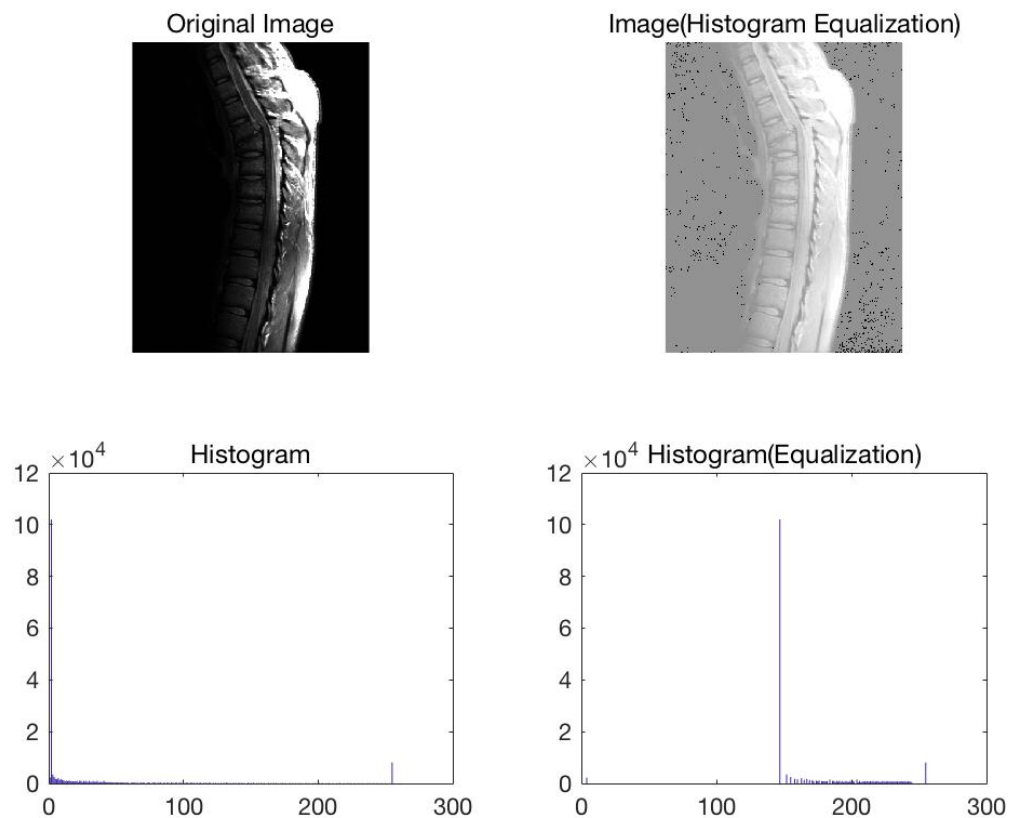
Figure 1.5: original image and enhanced image and histogram comparation of fig1.jpg
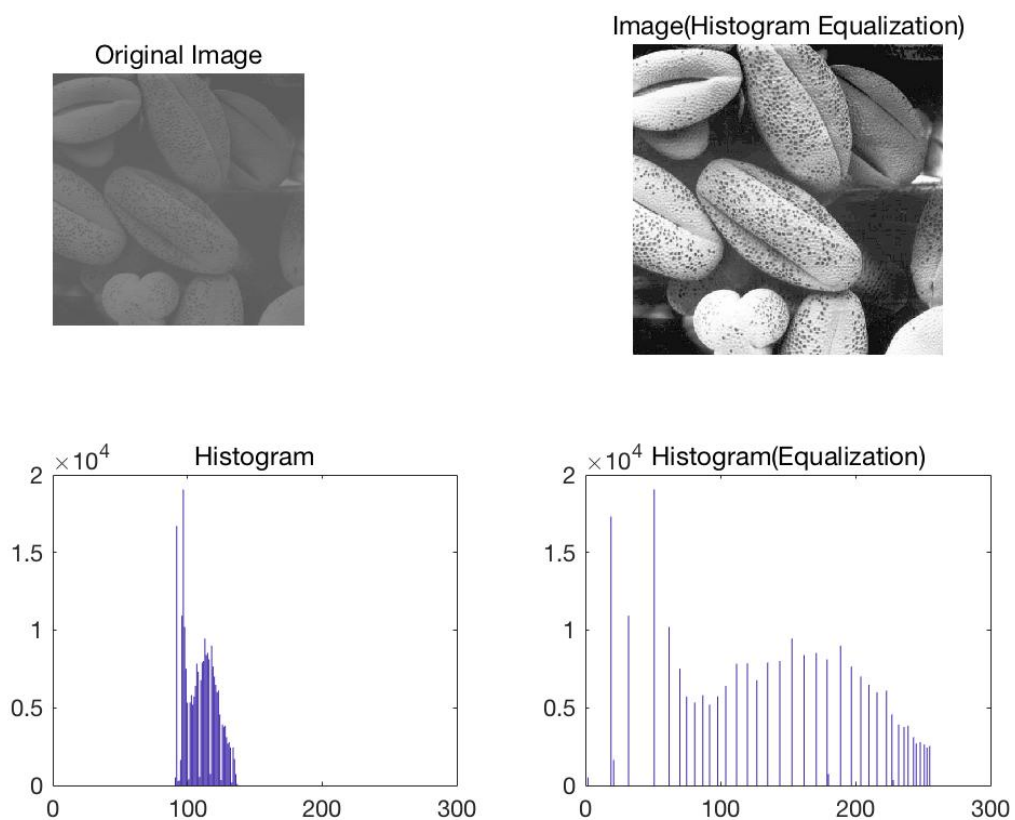
Figure 1.6: original image and enhanced image and histogram comparation of fig2.jpg

2.1 OverView

Implement the image enhancement task of Section 3.7 (Fig 3.43, page 171). 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, not to directly use Matlab functions such as imfilter or fspecial.)

2.2 Image b

2.2.1 Laplacian Transform Filter

We apply Laplacian Transform on the original image to get the image (b) using the following filter.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

2.2.2 Laplacian Transform

Image b:
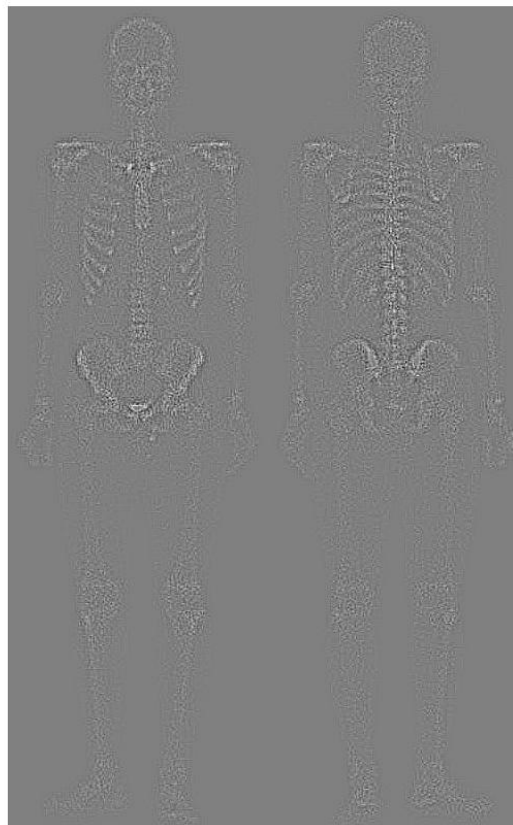
Figure 2.1: Laplacian Transform of skeleton_orig.tif

2.3   Image c

The we add the Laplacian of the original image to the original image, we will get the new image c. The new image c is a rather noisy sharpened image.
Image c:
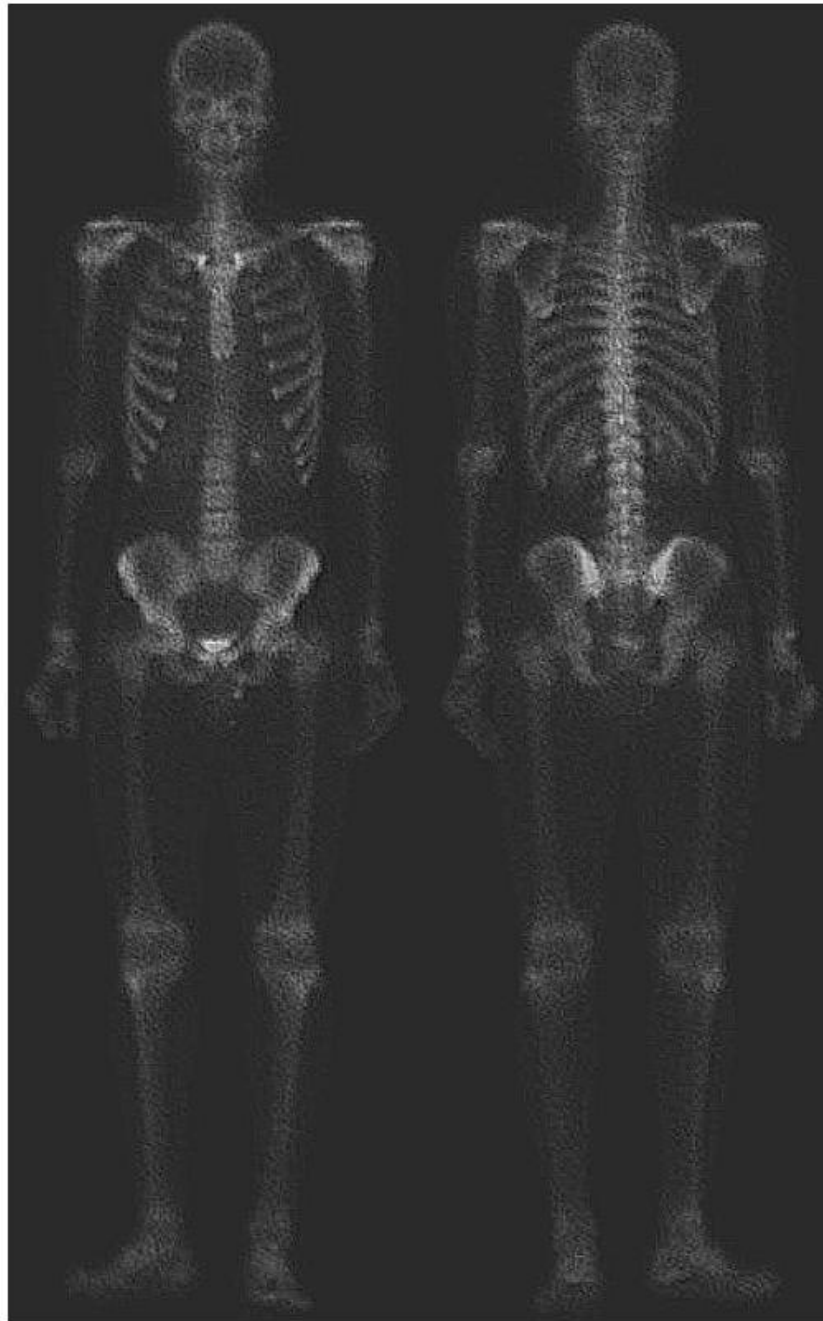
Figure 2.2: Laplacian Transform of skeleton_orig.tif

2.4   Image d

2.4.1   Sobel Gradient Masks

We will use two mask to separately get the components g_x and g_y. 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.

g_x:

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

g_y:

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

2.4.2   Image d

Image d:

Figure 2.3: Laplacian Transform of skeleton_orig.tif

2.5   Image e

Image e is formed by smoothing image d by 5*5 mean filter.
Image e:
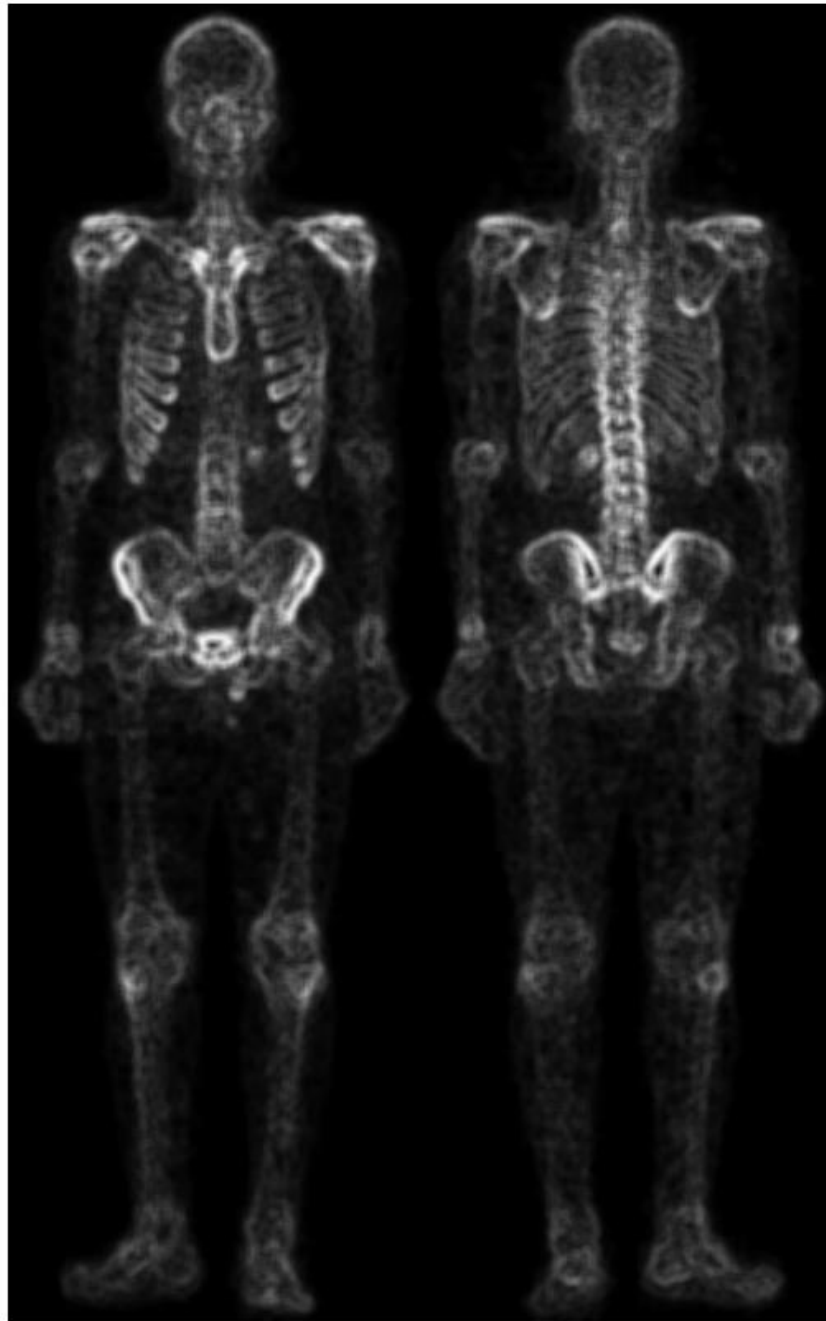
Figure 2.4: Laplacian Transform of skeleton_orig.tif

2.6   Image f

Image f is 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.
Image f:
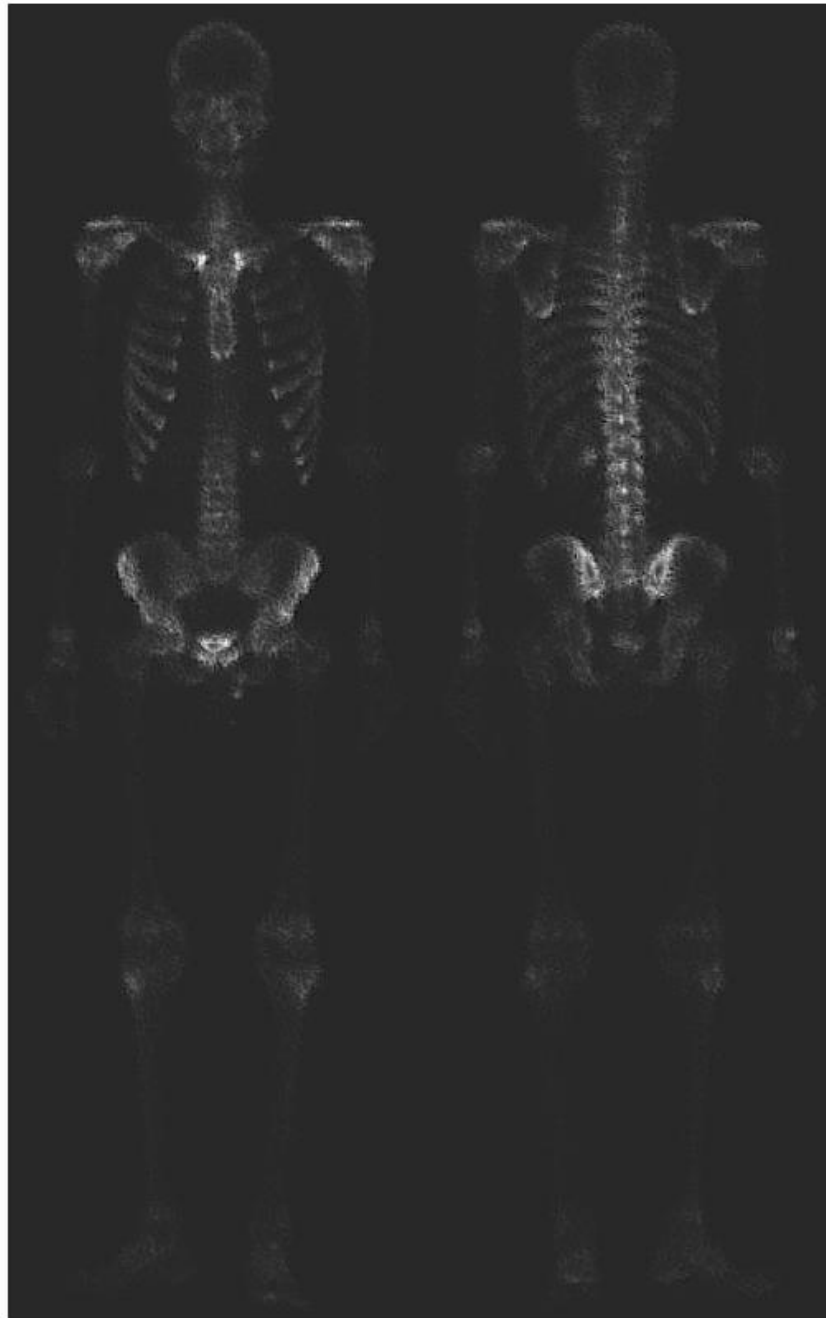
Figure 2.5: Laplacian Transform of skeleton_orig.tif

2.7   Image g

Adding the image f to the original image and then we get image g.
image g:
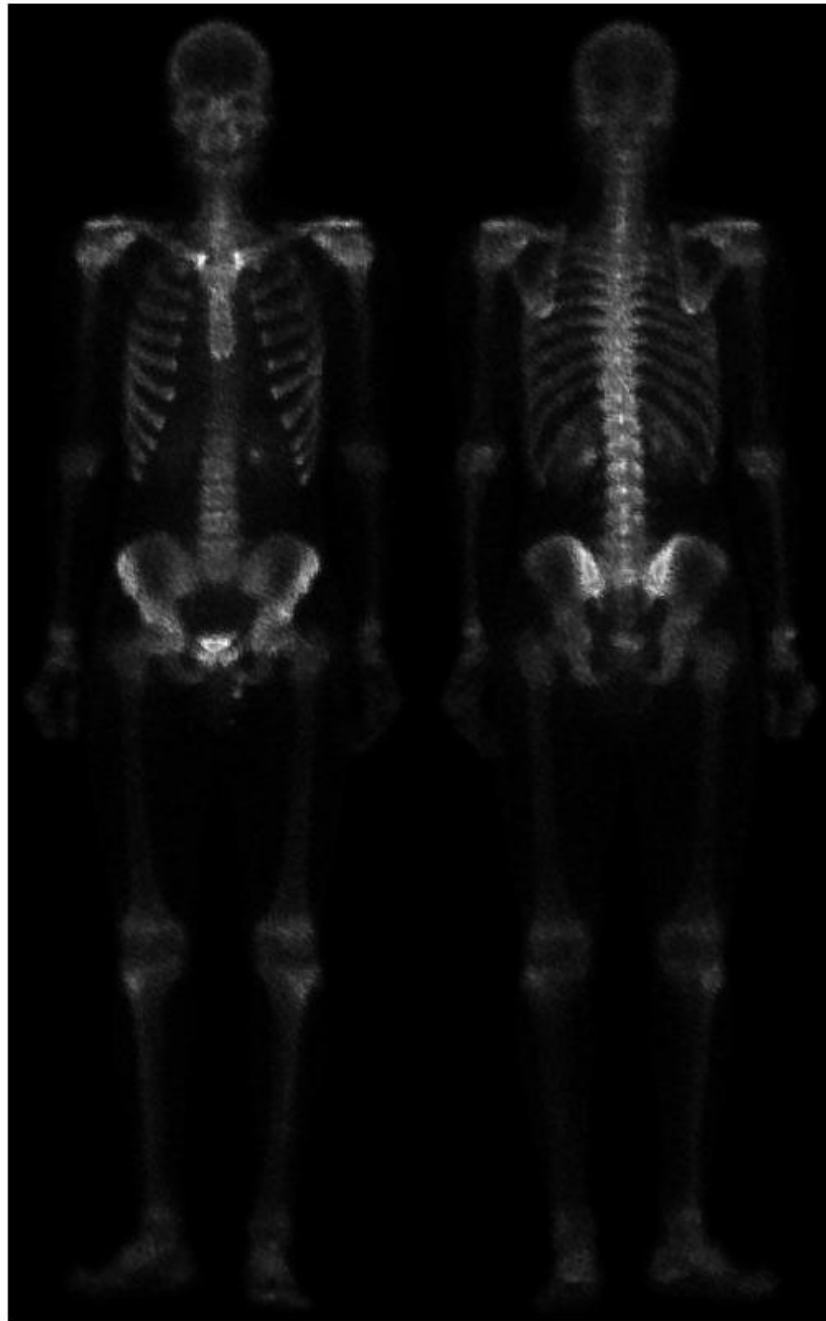
Figure 2.6: Laplacian Transform of skeleton_orig.tif

2.8   Image h

2.8.1   Power-Law Transformation

We use the following function to perform Power-Law Transformation on image g.

$$s = cr^\gamma \ (c \ = \ 1 \ and \ \gamma \ = \ 1) \tag{2.1}$$

2.8.2   Image h

Image h:
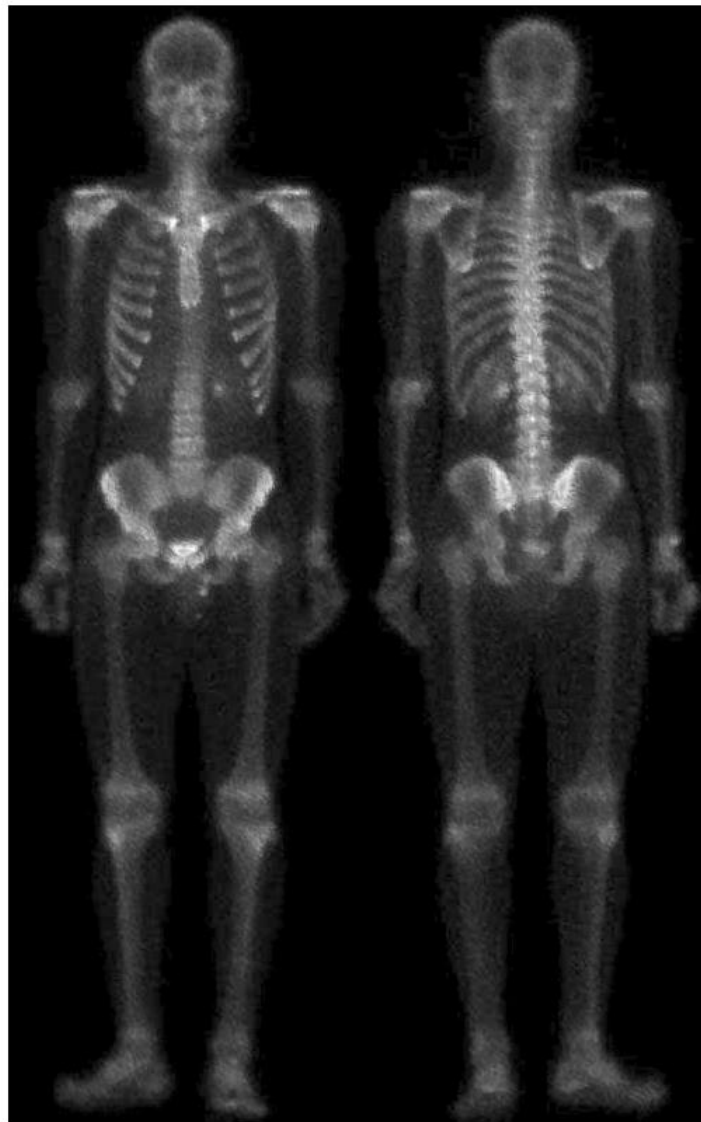
Figure 2.7: Laplacian Transform of skeleton_orig.tif

3.1 OverView

Implement the ideal, Butterworth and Gaussian lowpass and highpass filters and compare the results under different parameters using the image characters_test_pattern.tif (this image file can be found at the ftp server ftp://ftp.cs.sjtu.edu.cn:990/lu-ht/DIP/ images) as the test pattern.

3.2 Fourier Transform

3.2.1 Transform Function

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

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.1}$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.2}$$

3.2.2 Fast Fourier Transform

The inverse funtion of FFT is as folows:

$$F(u+\frac{M}{2},v+\frac{N}{2}) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux+x\frac{M}{2}}{M}+\frac{vy+v\frac{N}{2}}{N})} \tag{3.3}$$

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} * e^{-j\pi x} * e^{-j\pi y} \tag{3.4}$$

$$= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} -1^{x+y} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{3.5}$$

3.3 Filtering

- Expand image$(M * N)$ to image a$((2 * M) * (2 * N))$

- Perform the fourier transform on the new image and get image b

- Shift image b and get image c

- Perform the convolution on image c and filter and get image d

- Shift image d back and get image e

- Perform the inverse fourier transform on image e and get image f

- Cut image f to get the final result image$(M * N)$;

3.4   Ideal Filter

3.4.1   Ideal Low Pass

Ideal Low Pass Filter

The ideal low pass filter is defined as follows:

$$H(u,v) = \begin{cases} 1, & D(u,v) \leq 0 \\ 0, & D(u,v) > 0 \end{cases} \tag{3.6}$$

$$D(u,v) = \sqrt{[(u - \frac{P}{2})^2 + (v - \frac{Q}{2})^2]} \tag{3.7}$$

Result
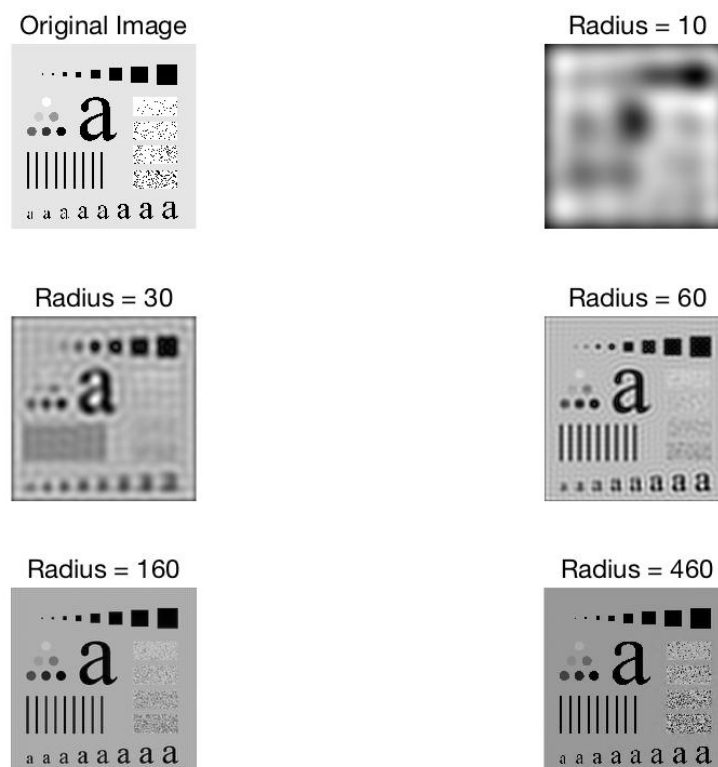
Figure 3.1: ILPF of characters_test_pattern.tif

3.4.2 Ideal High Pass

Ideal High Pass Filter

The ideal high pass filter is defined as follows:

$$H(u,v) = \begin{cases} 0, & D(u,v) \leq 0 \\ 1, & D(u,v) > 0 \end{cases} \tag{3.8}$$

$D(u,v)$ is defined in (3.7)
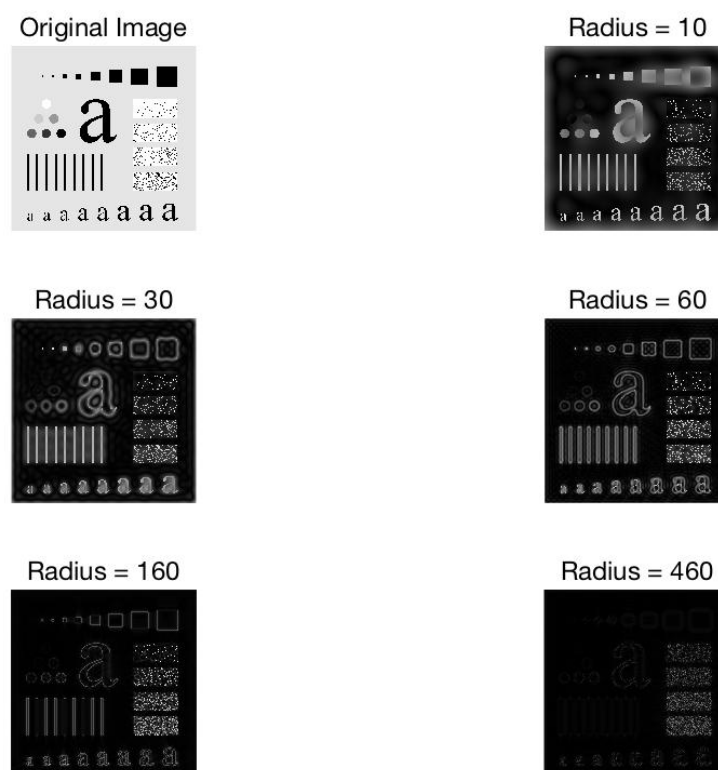
Result

Figure 3.2: ILPF of characters_test_pattern.tif

3.5 Butterworth Filter

3.5.1 Butterworth Low Pass

Butterworth Low Pass Filter

The butterworth low pass filter is defined as follows:

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

(3.9)

$D(u,v)$ is defined in (3.7)
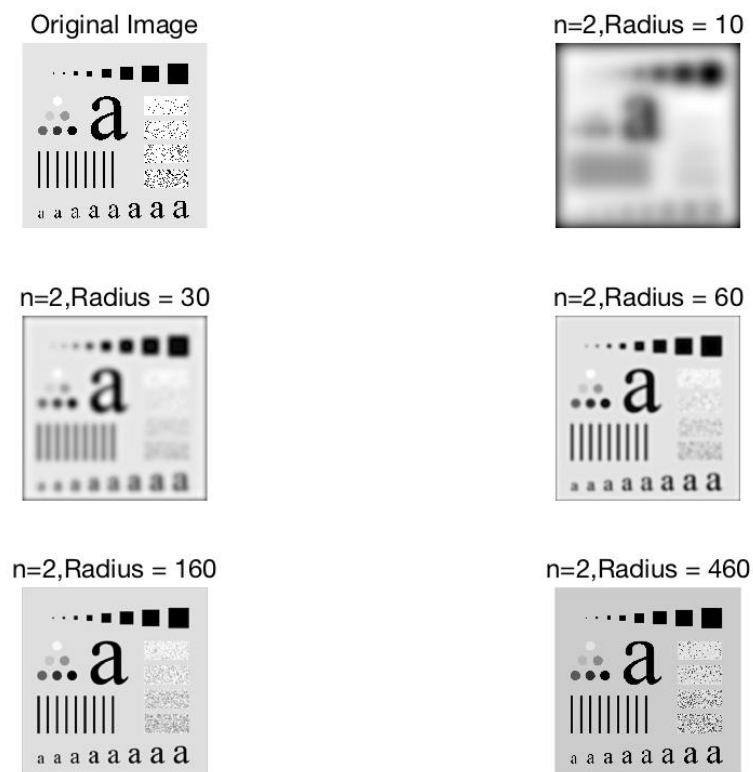
Result

Figure 3.3: BLPF of characters_test_pattern.tif

3.5.2 Butterworth High Pass

Butterworth High Pass Filter

The butterworth high pass filter is defined as follows:

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

(3.10)

19

$D(u,v)$ is defined in (3.7)
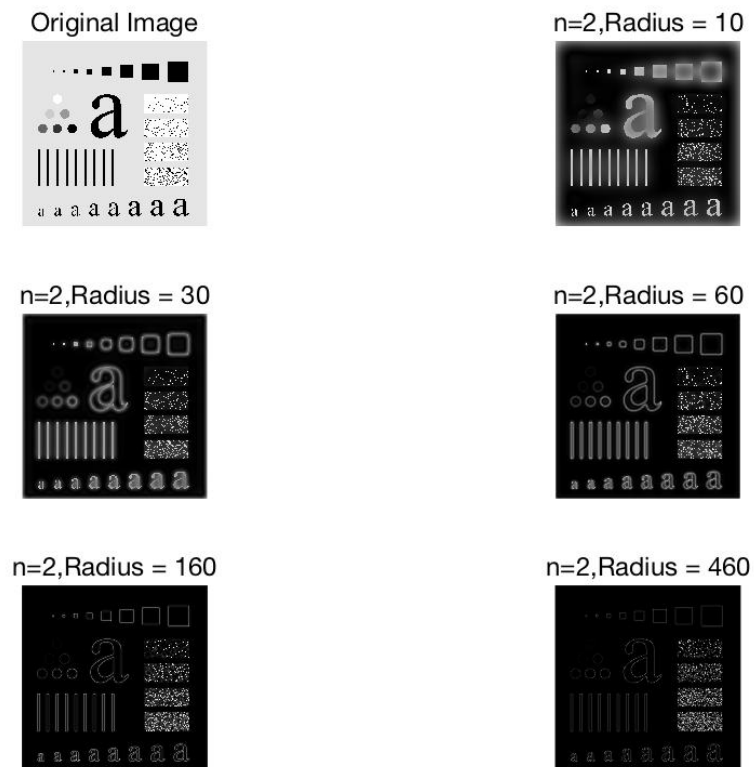
Result

Figure 3.4: BLPF of characters_test_pattern.tif

3.6 Gaussian Filter

3.6.1 Gaussian Low Pass

Gaussian Low Pass Filter

The gaussian low pass filter is defined as follows:

$$H(u,v) = e^{-\frac{D^2(u,v)}{2\sigma^2}} \tag{3.11}$$

$D(u,v)$ is defined in (3.7)
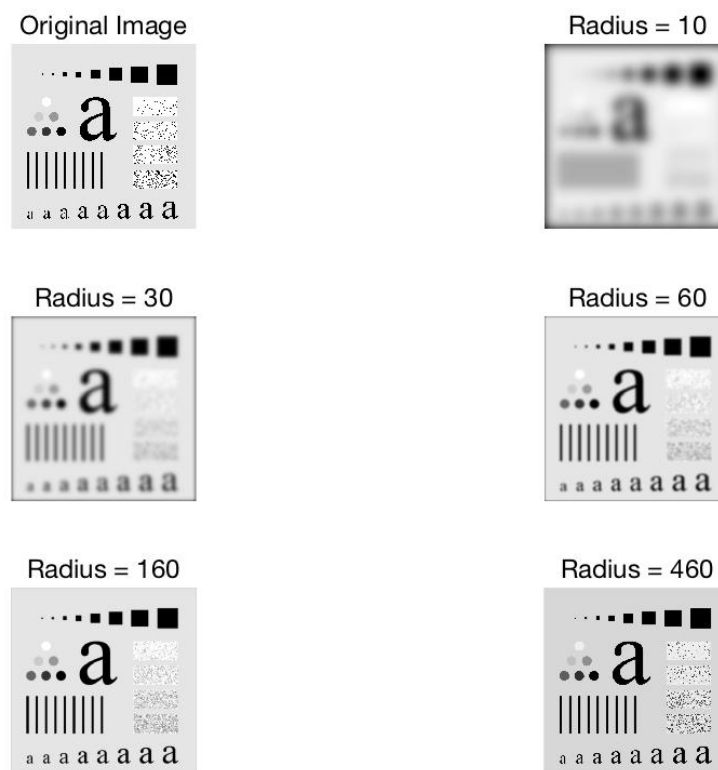
Result

Figure 3.5: GLPF of characters_test_pattern.tif

3.6.2 Gaussian High Pass

Gaussian High Pass Filter

The gaussian high pass filter is defined as follows:

$$H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2\sigma^2}} \tag{3.12}$$

$D(u,v)$ is defined in (3.7)

Result
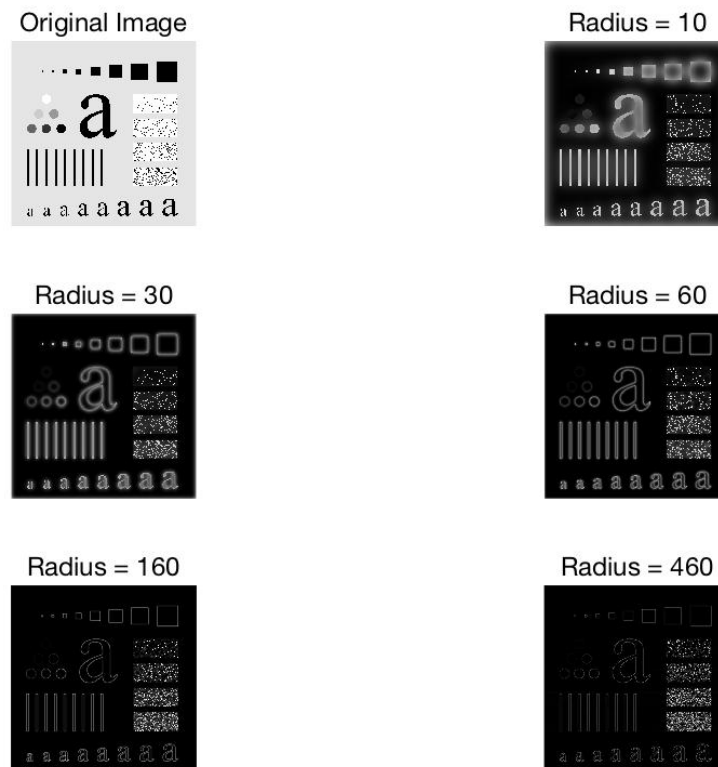
Figure 3.6: GLPF of characters_test_pattern.tif

4.1 OverView

In this problem, you are required to write a program to generate different types of random noise (Uniform, Gaussian, Rayleigh, Gamma, Exponential and Impulse, first started from the uniform noise and then use some functions to convert the uniform noise to Gaussian, Rayleigh, Gamma and Exponential; Impulse noise is generated in a different way, consulting the textbook and some other references) and then add these noises to the test patter image Fig0503(original_pattern).tif to compare the visual results of the noisy images.

Add some of these noises to the circuit image Circuit.tif (images can be found at ftp:// ftp.cs.sjtu.edu.cn:990/lu-ht/DIP/images) and investigate the noise reduction results using different mean filters and order statistics filters as the textbook did at pages 344-352 (Pages 322-329 in the electronic version of the textbook).

4.2 This is a section

4.2.1 This is a subsection

4.2.2 This is a subsection

5.1 OverView

5.2 This is a section

5.2.1 This is a subsection

5.2.2 This is a subsection

A.1 Problem1

```matlab
% Problem 1
% by Xue Fanyong
% Student ID:515030910443
% Histogram Equalizatio

%% Main Part
image1 = imread(' Image Path/Fig1.jpg' );
image2 = imread(' Image Path/Fig2.jpg' );

[histogram1,histogram_e1,transfer_f1,image_e1] =
histogram_equalization(image1);
[histogram2,histogram_e2,transfer_f2,image_e2] =
histogram_equalization(image2);

plot_data(image1,image_e1,histogram1,histogram_e1,transfer_f1);
plot_data(image2,image_e2,histogram2,histogram_e2,transfer_f2);

%% Functions Part

% get histogram of image
% image: get histogram of it
% histogram: the histogram of image
function histogram = get_histogram(image)
    histogram = zeros(256,1);
    [row,col]=size(image);
    for r = 1:row
        for c = 1:col
            gray = image(r,c);
            histogram(gray+1)=histogram(gray+1)+1;
        end
    end
end

% do the histogram_equalization for image
% image: do histogram_equalization for it
% histogram: original histogram; histogram_e:
% histogram after histogram
% equalizatio; transfer_f: transfer function;
% image_e: image after histogram
% equalizatio
function [histogram,histogram_e,transfer_f,image_e] =
histogram_equalization(image)
    [row,col]=size(image);
    transfer_f = zeros(256,1);
    histogram = get_histogram(image);
    transfer_f(1) = 256*histogram(1)/(row*col);

    for i = 2:256
        transfer_f(i) = transfer_f(i-1)+255*histogram(i)/(row*col);
```

```
50        end
51        transfer_f = round(transfer_f);
52
53        image_e = image;
54        for r = 1:row
55            for c = 1:col
56                image_e(r,c)=transfer_f(image(r,c)+1);
57            end
58        end
59        histogram_e = get_histogram(image_e);
60  end
61
62  % plot data
63  % image:original image; image_e:
64  % image after histogram equalizatio;
65  % histogram: original histogram;
66  % histogram_e: histogram after histogram equalizatio;
67  % transfer_f: transfer function
68  function plot_data(image,image_e,histogram,histogram_e,transfer_f)
69        figure();
70        subplot(2,3,1);
71        imshow(image);
72        title(" Original Image" );
73        subplot(2,3,2);
74        imshow(image_e);
75        title(" Image(Histogram Equalization)" );
76        subplot(2,3,3);
77        bar(histogram);
78        title(" Histogram" );
79        subplot(2,3,4);
80        bar(histogram_e);
81        title(" Histogram(Equalization)" );
82        subplot(2,3,5);
83        plot(transfer_f);
84        title(" Transfer Funciton" );
85  end
```

A.2   Problem 2

```
1  % Problem 2
2  % by Xue Fanyong
3  % Student ID:515030910443
4  % Combining spatial enhancement methods
5
6  %% Main Part
7  image = imread(' Image Path/skeleton_orig.tif' );
8  [row,col] = size(image);
9  mask = [-1 -1 -1;-1 8 -1;-1 -1 -1];
10 mask = double(mask);
```

```matlab
11  b_image = laplace_transformations(image,mask);
12  c_image = b_image+im2double(image);
13  d_image = sobel_gradient(image);
14  e_image = smooth(d_image);
15  f_image = im2double(e_image).*c_image;
16  g_image = abs(f_image)+im2double(image);
17  h_image = sqrt(g_image);
18  plot_data(image,b_image,c_image,d_image,
19          e_image,f_image,g_image,h_image);
20
21  %%% Function Part
22
23  % Laplace Transfromation for image using mask
24  % Input:
25  %    image:image you want to perform
26  %    mask:Laplace mask you want to use
27  % Output:
28  %    image_l:image after laplace transformation
29
30  function image_l = laplace_transformations(image,mask)
31      [row,col] = size(image);
32      mask = double(mask);
33      %append image
34      image_l = im2double(image);
35      image = [zeros(row,2) image zeros(row,2)];
36      image = [zeros(2,col+4);image;zeros(2,col+4)];
37      image_append = im2double(image);
38
39      for r = 1:row
40          for c = 1:col
41              image_l(r,c) = sum(sum(image_append(r:r+2,c:c+2).*mask));
42          end
43      end
44  end
45  %{
46      sobel gradient for image
47  %}
48  function image_s = sobel_gradient(image)
49      [row,col] = size(image);
50      x_mask = [-1 -2 -1;0 0 0;1 2 1];
51      y_mask = [-1 0 1;-2 0 2;-1 0 1];
52      image_s = image;
53      image = double(image);
54
55      for r = 2:row-1
56          for c = 2:col-1
57              image_s(r,c) =
58              abs(sum(sum(image(r-1:r+1,c-1:c+1).*x_mask)))+
59              abs(sum(sum(image(r-1:r+1,c-1:c+1).*y_mask)));
60          end
61      end
```

```matlab
62  end
63  %{
64      smooth image using 5*5 mean filter
65  %}
66  function image_s = smooth(image)
67      [row,col] = size(image);
68      image_s = image;
69      for r = 3:row-2
70          for c = 3:col-2
71              image_s(r,c) = mean(mean(image(r-2:r+2,c-2:c+2)));
72          end
73      end
74  end
75  %{
76      plot data
77  %}
78  function plot_data(a,b,c,d,e,f,g,h)
79      figure();
80
81      subplot(241);
82      imshow(a);
83      title(' (a) Oringinal Image' );
84
85      subplot(242);
86      imshow(b,[]);
87      title(' (b) Laplacian of (a)' );
88
89      subplot(245);
90      imshow(c,[]);
91      title(' (c) Sharpened image' );
92
93      subplot(246);
94      imshow(d);
95      title(' (d) Sobel gradient' );
96
97      subplot(243);
98      imshow(e);
99      title(' (e) Smoothed sobel image' );
100
101     subplot(244);
102     imshow(f,[]);
103     title(' (f) Product of (c) and (e)' );
104
105     subplot(247);
106     imshow(g,[]);
107     title(' (g) Sharpened image' );
108
109     subplot(248);
110     imshow(h,[]);
111     title(' (h) Final result' );
112 end
```

1 2 3 4 5 A

Codes

A.3 Problem 3

```matlab
%{
    Problem 3
    by Fanyong Xue
    Student ID:515030910443
    Filtering in frequency domain

%}

%% Main Part
image = imread('Image Path/characters_test_pattern.tif');

ideal_low_plot_data(image);
ideal_high_plot_data(image);
butterworth_low_plot_data(image);
butterworth_high_plot_data(image);
gaussian_low_plot_data(image);
gaussian_high_plot_data(image);


%% Function Part

% Ideal
function ideal_high_plot_data(image)

    figure('name','Ideal High Pass');

    subplot(321);
    imshow(image);
    title('Original Image');

    subplot(322);
    imshow(IHPF(image,10),[]);
    title('Radius = 10');

    subplot(323);
    imshow(IHPF(image,30),[]);
    title('Radius = 30');

    subplot(324);
    imshow(IHPF(image,60),[]);
    title('Radius = 60');

    subplot(325);
    imshow(IHPF(image,160),[]);
    title('Radius = 160');

    subplot(326);
    imshow(IHPF(image,460),[]);
    title('Radius = 460');
```

```matlab
50
51   end
52   function ideal_low_plot_data(image)
53
54       figure('name','Ideal Low Pass');
55
56       subplot(321);
57       imshow(image);
58       title('Original Image');
59
60       subplot(322);
61       imshow(ILPF(image,10),[]);
62       title('Radius = 10');
63
64       subplot(323);
65       imshow(ILPF(image,30),[]);
66       title('Radius = 30');
67
68       subplot(324);
69       imshow(ILPF(image,60),[]);
70       title('Radius = 60');
71
72       subplot(325);
73       imshow(ILPF(image,160),[]);
74       title('Radius = 160');
75
76       subplot(326);
77       imshow(ILPF(image,460),[]);
78       title('Radius = 460');
79
80   end
81   function image_i = ILPF(image,radius)
82
83       fliter = ILPF_fliter(image,radius);
84       image_i = transfer(image,fliter);
85       %{
86       image_f = fft2(image,2*row,2*col);
87       image_f = fftshift(image_f);
88       %image_f = log(1+abs(image_f));
89
90       %%%%%%%
91       image_i = image_f.*fliter;
92
93       image_i = ifftshift(image_i);
94       image_i = ifft2(image_i);
95
96       %image_i = abs(image_i);
97       image_i = image_i(1:row,1:col);
98       image_i = real(image_i);
99       image_f = log(1+abs(image_f));
100      %}
```

```matlab
101  end
102  function image_i = IHPF(image,radius)
103      fliter = 1-ILPF_fliter(image,radius);
104      image_i = transfer(image,fliter);
105  end
106  function fliter = ILPF_fliter(image,radius)
107      [row,col]=size(image);
108      fliter = zeros(2*row,2*col);
109
110      for r = 1:2*row
111          for c = 1:2*col
112              if sqrt((r-row)^2+(c-col)^2) <= radius
113                  fliter(r,c) = 1;
114              end
115          end
116      end
117  end
118
119
120  % Butterworth
121  function butterworth_low_plot_data(image)
122      figure('name','Butterworth Low Pass');
123
124      subplot(321);
125      imshow(image);
126      title('Original Image');
127
128      subplot(322);
129      imshow(BLFP(image,2,10),[]);
130      title('n=2,Radius = 10');
131
132      subplot(323);
133      imshow(BLFP(image,2,30),[]);
134      title('n=2,Radius = 30');
135
136      subplot(324);
137      imshow(BLFP(image,2,60),[]);
138      title('n=2,Radius = 60');
139
140      subplot(325);
141      imshow(BLFP(image,2,160),[]);
142      title('n=2,Radius = 160');
143
144      subplot(326);
145      imshow(BLFP(image,2,460),[]);
146      title('n=2,Radius = 460');
147
148  end
149  function butterworth_high_plot_data(image)
150      figure('name','Butterworth High Pass');
151
```

1 | 2 | 3 | 4 | 5 | A

```matlab
152        subplot(321);
153        imshow(image);
154        title(' Original Image' );
155
156        subplot(322);
157        imshow(BHFP(image,2,10),[]);
158        title(' n=2,Radius = 10' );
159
160        subplot(323);
161        imshow(BHFP(image,2,30),[]);
162        title(' n=2,Radius = 30' );
163
164        subplot(324);
165        imshow(BHFP(image,2,60),[]);
166        title(' n=2,Radius = 60' );
167
168        subplot(325);
169        imshow(BHFP(image,2,160),[]);
170        title(' n=2,Radius = 160' );
171
172        subplot(326);
173        imshow(BHFP(image,2,460),[]);
174        title(' n=2,Radius = 460' );
175
176 end
177 function image_b = BLFP(image, n,radius)
178
179        fliter = BLFP_fliter(image, n,radius);
180        image_b = transfer(image,fliter);
181
182 end
183 function image_b = BHFP(image, n,radius)
184
185        fliter = 1-BLFP_fliter(image, n,radius);
186        image_b = transfer(image,fliter);
187
188 end
189 function fliter = BLFP_fliter(image, n,radius)
190        [row,col]=size(image);
191        fliter = zeros(2*row,2*col);
192
193        for r = 1:2*row
194            for c = 1:2*col
195                fliter(r,c) = 1/(1+(sqrt((r-row)^2+
196                              (c-col)^2)/radius)^(2*n));
197            end
198        end
199 end
200
201 % Gaussian
202 function gaussian_low_plot_data(image)
```

```matlab
203      figure( 'name' ,' Gaussian Low Pass' );
204
205      subplot(321);
206      imshow(image);
207      title(' Original Image' );
208
209      subplot(322);
210      imshow(GLFP(image,10),[]);
211      title(' Radius = 10' );
212
213      subplot(323);
214      imshow(GLFP(image,30),[]);
215      title(' Radius = 30' );
216
217      subplot(324);
218      imshow(GLFP(image,60),[]);
219      title(' Radius = 60' );
220
221      subplot(325);
222      imshow(GLFP(image,160),[]);
223      title(' Radius = 160' );
224
225      subplot(326);
226      imshow(GLFP(image,460),[]);
227      title(' Radius = 460' );
228 end
229 function gaussian_high_plot_data(image)
230      figure( 'name' ,' Gaussian High Pass' );
231
232      subplot(321);
233      imshow(image);
234      title(' Original Image' );
235
236      subplot(322);
237      imshow(GHFP(image,10),[]);
238      title(' Radius = 10' );
239
240      subplot(323);
241      imshow(GHFP(image,30),[]);
242      title(' Radius = 30' );
243
244      subplot(324);
245      imshow(GHFP(image,60),[]);
246      title(' Radius = 60' );
247
248      subplot(325);
249      imshow(GHFP(image,160),[]);
250      title(' Radius = 160' );
251
252      subplot(326);
253      imshow(GHFP(image,460),[]);
```

33

1 2 3 4 5 A

```matlab
254        title(' Radius = 460' );
255 end
256 function image_g = GLFP(image,radius)
257        fliter = GLFP_fliter(image,radius);
258        image_g = transfer(image,fliter );
259 end
260 function image_g = GHFP(image,radius)
261        fliter = 1-GLFP_fliter(image,radius);
262        image_g = transfer(image,fliter );
263 end
264 function fliter = GLFP_fliter(image,radius)
265        [row,col]=size(image);
266        fliter = double(zeros(2*row,2*col));
267
268        for r = 1:2*row
269            for c = 2:2*col
270                fliter(r,c) = exp(-1*(((r-row)^2+
271                        (c-col)^2)/(2*radius^2)));
272            end
273        end
274 end
275
276 function image_t=transfer(image,fliter )
277        [row,col]=size(image);
278        image_f = fft2(image,2*row,2*col);
279        image_f = fftshift(image_f);
280        image_t = image_f.*fliter ;
281        image_t = ifftshift(image_t);
282        image_t = ifft2(image_t);
283        image_t = image_t(1:row,1:col);
284        image_t = abs(image_t);
285 end
286 %{
287
288 function [ mfft2 ] = JCGuoFFT2( data )
289        h = size(data, 1);
290        w = size(data, 2);
291        mfft2 = data;
292
293        if power(2, log2(h)) ~= h || power(2, log2(w)) ~= w
294            disp(' JCGuoFFT2 exit: h and w must be the power of 2!' )
295        else
296            for i = 1 : h
297                mfft2(i, :) = IterativeFFT(mfft2(i, :));
298            end
299
300            for j = 1 : w
301                mfft2(:, j) = IterativeFFT(mfft2(:, j));
302            end
303        end
304 end
```

1 2 3 4 5 A

```
305
306  function image_s = shift_image(image)
307
308      [row,col]=size(image);
309      image_s = image;
310
311      for r = 1:row
312          for c = 1:col
313              image_s(r,c) = image(r,c)*(-1)^(r+c);
314          end
315      end
316  end
317
318
319  function image_f = DFT(image,rows,cols)
320      [row,col]=size(image);
321
322      %pad image to rows*cols
323      image = [image zeros(row,cols-col)];
324      image = [image;zeros(rows-row,cols)];
325      image = double(image);
326      for i = 1:rows
327          k = cols/2;
328          M = round(log2(cols));
329          for j = 1:cols-2
330              if j<k
331                  t = image(i,k);
332                  image(i,k) = image(i,j);
333                  image(i,j) = t;
334              end
335              l = cols/2;
336              while l<=k
337                  k = k-1;
338                  l = l/2;
339              end
340              k = k+1;
341          end
342          for m = 1:M
343              la = 2^m;
344              lb = la/2;
345              for l = 1:lb
346                  r = (l-1)*2^(M-m);
347                  n = l-1;
348                  while n<rows-1
349                      lc = n+lb;
350                      t = image(lc,j)*exp(2*pi*r/rows);
351                      image(i,lc) = image(i,n) - t;
352                      image(i,n) = image(i,n) + t;
353                      n = n+la;
354                  end
355              end
```

1 2 3 4 5 A

```matlab
356            end
357        end
358        image_f = image;
359  end
360  function image_f=DFT(image,rows,cols)
361
362  end
363
364  function v = DFT_1(V)
365      n = length(V);
366      fft_m = BitReverseCopy(V);
367
368      for r = 1:log2(n)
369          m = power(2,r);
370          wm = exp(- 2 * pi * i / m);
371
372          for k = 0 : m : n - 1
373              w = 1;
374              for j = 0 : m / 2 - 1
375                  t = w * fft_m(k + j + m / 2 + 1);
376                  u = fft_m(k + j + 1);
377                  fft_m(k + j + 1) = u + t;
378                  fft_m(k + j + m / 2 + 1) = u - t;
379                  w = w * wm;
380              end
381          end
382      end
383  end
384  %}
```