Digital Image Processing 2017fall
Project Report

Name: Wang Yiqing
Student Number: 515 030 910 456
Email: WangYiqing_2015@sjtu.edu.cn

2017-12-10

Contents

1 Project 1 - Histogram Equalization

1.1 Project Proposal

In this project we implement histogram equalization. First, plot the histogram of an image, then implement histogram equalization and display the equalized image and histogram. Test images are *Fig1.jpg, Fig2.jpg.*

1.2 Preliminaries

Let r denote the intensity of a pixel in the original image and $r \in [0, L-1]$. We consider a transformation

$$s = T(r) \quad 0 \leq r \leq L-1 \tag{1.1}$$

that produce an output intensity level s for every pixel in the input image having intensity r. We assume that:
(a) $T(r)$ is a monotonically increasing function in the interval $0 \leq r \leq L-1$; and
(b) $0 \leq T(r) \leq L-1$ for $r \in [0, L-1]$.
Condition **(a)** and **(b)** guarantee the existing of inverse function $r = T^{-1}(s)$ which is monotonically increasing. The central idea is that *intensity levels of an image can be viewed as random variables in the interval* $[0, L-1]$ *and can be described by probability density function (PDF).* Let $p_r(r)$ and $p_s(s)$ to be the PDF of r and s respectively. Thus we can apply the basic result of probability theory that *if* $T(r)$ *is differentiable over the range of interest, we have this below relationship between* $p_r(r)$ *and* $p_s(s)$*:*

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| \tag{1.2}$$

Then consider the cumulative distribution function(CDF) which is exactly used here as the transformation function $T(r)$

$$s = T(r) = (L-1) \int_0^r p_r(w) dw \tag{1.3}$$

Now we are ready to prove the transformation really does the histogram equalization. Firstly, compute the derivatives. The last = is from the assumption **(a)** stating the monotonically increasing.

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)p_r(r) = \left| \frac{ds}{dr} \right| \tag{1.4}$$

Thus, substituting the result for Eq.1.2, we get the desired result

$$p_s(s) = p_r(r) \frac{1}{(L-1)p_r(r)} = \frac{1}{L-1} \tag{1.5}$$

which show $p_s(s)$ follows uniform distribution.

1.3 Histogram equalization

Histogram equalization on an image of size $M \times N$ is like a discrete version of the process in the preliminaries section.

$$p_r(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, ..., L-1 \tag{1.6}$$

where n_k is the number of pixels that have intensity r_k. The discrete form of Eq.1.3 is

$$s_k = T(r_k) = (L-1) \sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^{k} n_k \qquad k = 0, 1, 2, ..., L-1 \tag{1.7}$$

Based on these discrete form equation, I implement the matlab function and test them on *Fig1.jpg, Fig2.jpg* and get the results listed in 1.1 and 1.2 respectively.

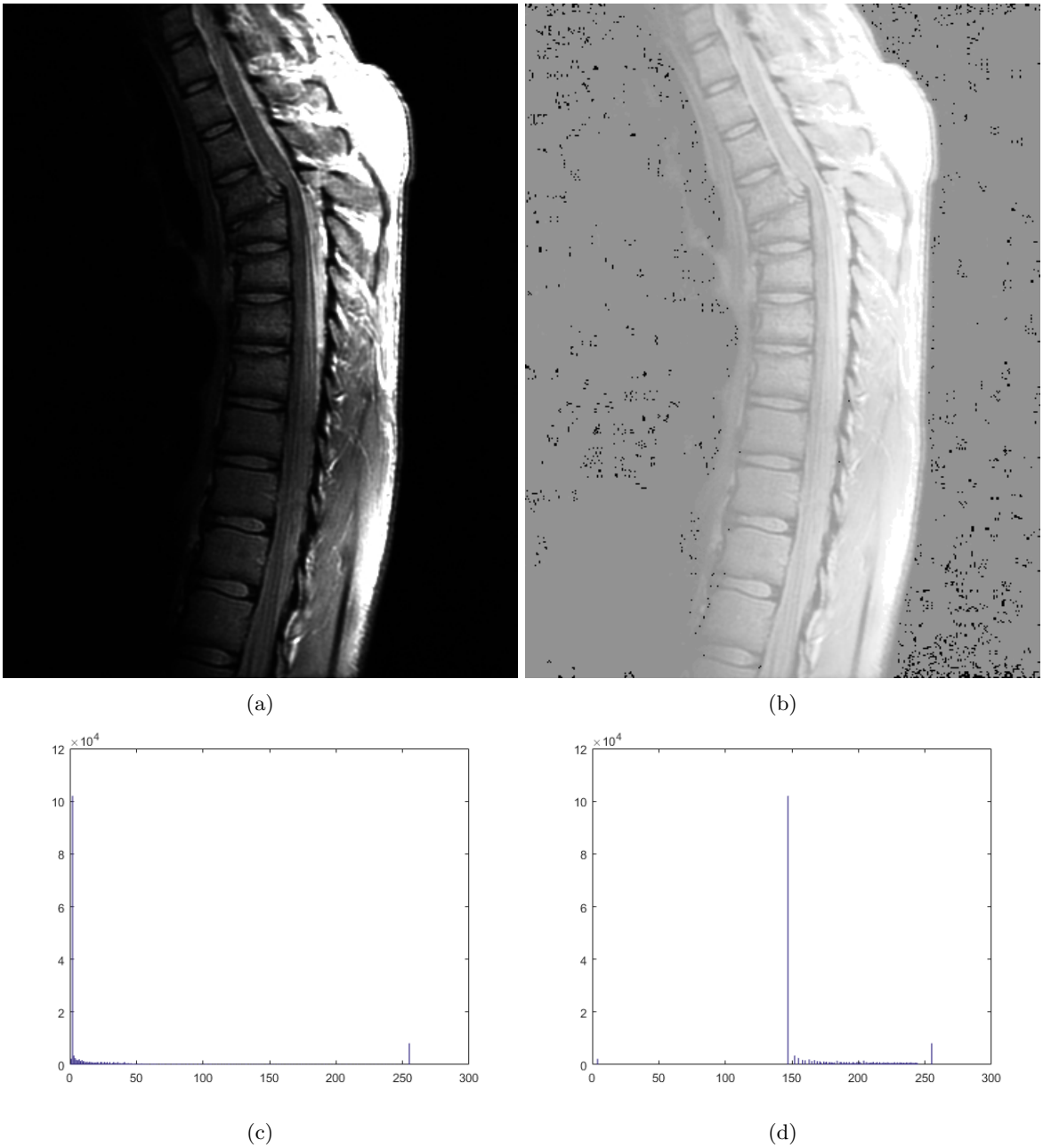


Figure 1.1: Results of Fig1.jpg. (a)Original image. (b)Processed image after applying histogram equalization. (c)Histogram of original image. (d)Histogram of processed image.

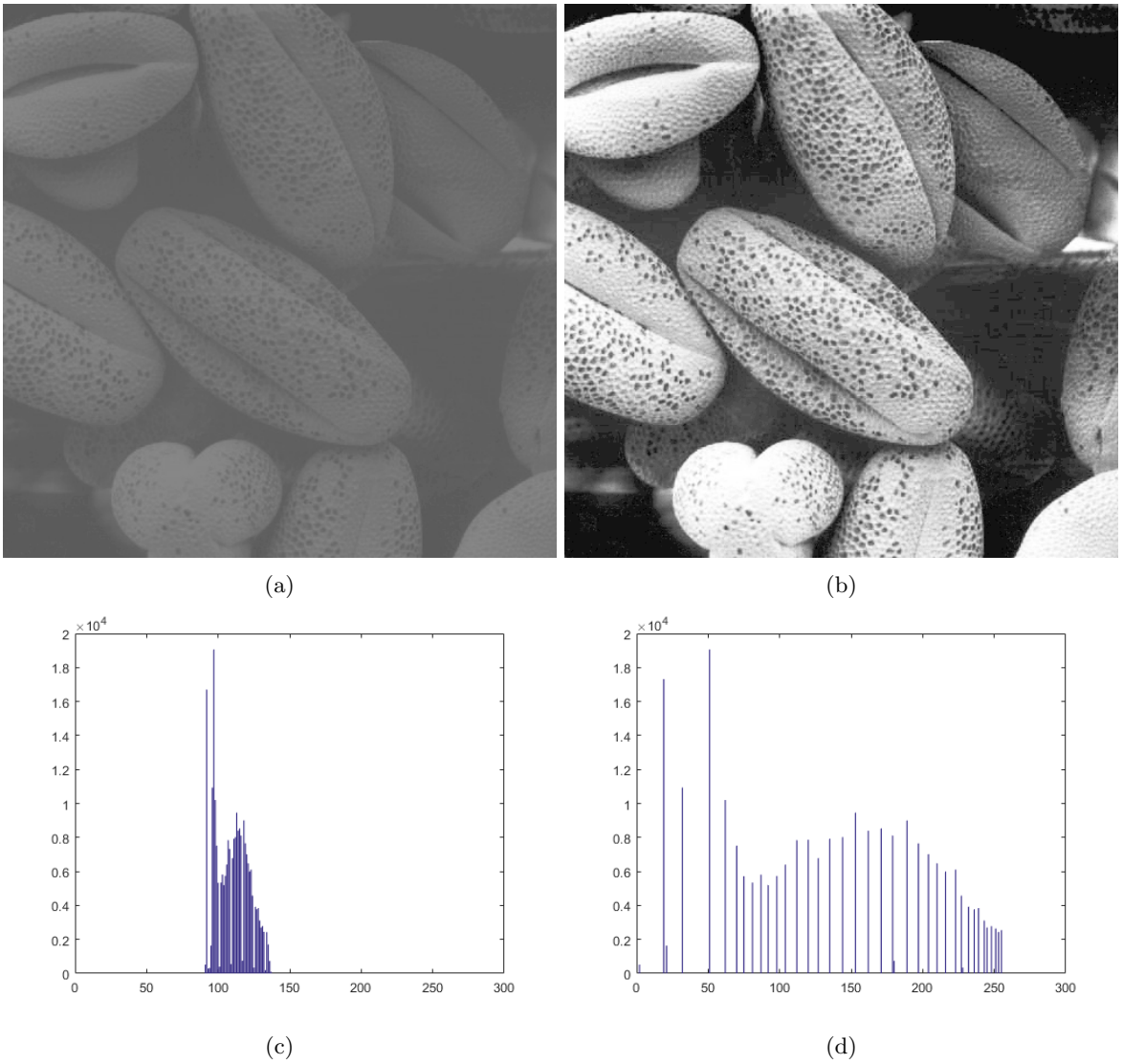


Figure 1.2: Results of Fig2.jpg. (a)Original image. (b)Processed image after applying histogram equalization. (c)Histogram of original image. (d)Histogram of processed image.

1.4 Discussion

We can see that histograms are quite different from continuous functions because there are gaps between each horizontal value. So the histogram of the equalized image is like a sparse version of the original image. The result of Fig2(low contrast ratio) shows an increasing on contrast ratio and it's more useful than the original result. However, Fig1(high contrast ratio) is not a good example of histogram equalization. The affect is just like a intensity transition. We can simply draw a not so serious conclusion that histogram equalization is useful for images with low contrast ratio but not for images with high contrast ratio.

1.5 Implementation

There is some key part of my implementation.

```
function [x, y] = histShow( imgf )
%HISTSHOW display the histogram graph of the imgf
% x - the horizontal axis of histogram,
% y - the vertical axis of histogram
g = imgf(:) + 1;
n = length(g);
x = (1 : 256);
y = zeros(1, 256);
for i = (1 : n)
    y(g(i)) = y(g(i)) + 1;
end

end

function [ imgg ] = histEqual( imgf )
%HISTEQUAL
%
[x, y] = histShow(imgf);
T = zeros(1, 256);
a = 0;
g = imgf(:);
n = length(g);
for i = (1 : 256)
    T(i) = a + y(i);
    a = T(i);
end
T = round(255 * T / n);
for i = (1 : n)
    g(i) = T(g(i)+1);
end
imgg = reshape(g, size(imgf));

end
```

2 Project 3 - Filtering in Frequency Domain

2.1 Project Proposal

Implement the ideal, Butterworth and Gaussian lowpass and highpass filters and the results under different parameters using the image character_test_pattern.tif

2.2 Preliminaries

2.2.1 Summary of Fourier transform properties

Table 2.1: Summary of useful formulas.

Name	Expression(s)
1D FT	$$F(u) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ut}dt \quad (2.1)$$
1D IFT	$$f(t) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ut}du \quad (2.2)$$
1D DFT	$$F(u) = \sum_{t=0}^{M-1} f(t)e^{-j2\pi ut/M} \quad (2.3)$$
1D IDFT	$$f(t) = \frac{1}{M}\sum_{u=0}^{M-1} F(u)e^{j2\pi ut/M} \quad (2.4)$$
2D FT	$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy \quad (2.5)$$
2D DFT	$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)} \quad (2.6)$$
2D IDFT	$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1]} F(u,v)e^{j2\pi(ux/M+vy/N)} \quad (2.7)$$
Power spectrum	$$P(u,v) = \lvert F(u,v)\rvert^2 \quad (2.8)$$

2.2.2 Steps for filtering in the frequency domain

1. Given an input image $f(x, y)$ of size $M \times N$, obtain the padding parameters $P = 2M$ and $Q = 2N$.Form a padded image, $f_p(x, y)$, of size $P \times Q$ by appending zeros.
2. Multiply $f_p(x, y)$ by $(-1)^{x+y}$ to center its transform.
3. Compute the DFT, $F(u, v)$ of the image from centered padded image.
4. Generate a real, symmetric filter function $H(u, v)$ of size $P \times Q$ with center at coordinates $(P/2, Q/2)$. Form the product $G(u, v) = H(u, v)F(u, v)$.
5. Obtain the precessed image: $g_p(x, y) = \left\{\text{real}\left[\ \mathcal{F}^{-1}[G(u, v)]\right]\right\}(-1)^{x+y}$ where the real part is selected in order to ignore parasitic complex components resulting from computational inaccuracies.
6. Obtain the final processed result, $g(x, y)$, by extracting the top left $M \times N$ quadrant of $g_p(x, y)$

2.3 Image Smoothing Using Frequency Domain Filters

Edges and other sharp intensity transitions such as noise in an image contribute significantly to the high-frequency content of its Fourier transform. Hence, smoothing is achieved in the frequency domain by high-frequency attenuation.

2.3.1 Ideal lowpass filters

Ideal lowpass filters (ILPF) is very sharp as it is specified by the function

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where D_0 is a positive constant and $D(u,v)$ is the distance between (u,v) in frequency domain and the center of the frequency rectangle; that is

$$D(u,v) = \left[(u - P/2)^2 + (v - Q/2)^2\right]^{1/2} \tag{2.10}$$

One way to establish a set of standard cutoff frequency loci is to compute circles that enclose specified amounts of total image power P_T.

$$P_T = \sum_{u=0}^{P-1} \sum_{v=0}^{Q-1} P(u,v) \tag{2.11}$$

The percentage of power enclosed by the circle of radius D_0 with origin at the center of the frequency rectangle is

$$\alpha = 100 \left[\sum_u \sum_v P(u,v)/P_T \right] \quad \text{(u,v) is inside the circle} \tag{2.12}$$

3 Project 8 - Morphological Processing

3.1 Project Proposal

Implement the *"Opening by reconstruction", "Filling holes" and "Border clearing"* operations on textbook *chapter 9.5*. The task is to reproduce the results in *Figure 9.29, 9.31 and 9.32*.

3.2 Preliminaries

3.2.1 Basic morphological operations

Morphology offers a unified and powerful approach to numerous image processing problem. These operations defined based on set theory. In this project we consider that morphological operations are conducted on binary images (preprocessing is required for gray-level images). The below table describes the basic widely used morphological processing.

Table 3.1: Summary of basic morphological operations.

Operation	Equation	Comments
Translation	$(B)_z = \{w \mid w = b + z,\ b \in B\}$	Translation the origin of B to point z.
Reflection	$\hat{B} = \{w \mid w = -b,\ b \in B\}$	Reflects all elements of B about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in A.
Difference	$A - B = \{w \mid w \in A, w \notin B\}$	Set of points that belong to A but not to B.
Dilation	$A \oplus B = \left\{z \mid (\hat{B}_z) \cap A \neq \emptyset\right\}$	Expands the boundary of A.
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	Contracts the boundary of A.
Opening	$A \circ B = (A \ominus B) \oplus B$	Smoothes contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks.
Closing	$A \bullet B = (A \oplus B) \ominus B$	Smoothes contours, fuses narrow breaks and long thin gulfs and eliminates small holes.
Hit-or-miss transform	$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2)$	The set of pints at which, simultaneously B_1 found a hit in A and B_2 found a match in A^c.

3.2.2 Morphological restoration

With the basic operations, we can discuss a powerful morphological transformation *morphological restoration* that involves two images and a structuring element. One image, the *marker*, contains the starting points for the transformation. The other image, the *mask*, constrains the transformation.

Geodesic dilation and erosion

Geodesic dilation and erosion are the central concepts to morphologic reconstruction. Let F denote the maker and G denote the mask and $F \subseteq G$. The *geodesic dilation* of size 1 of the marker with respect to the mask, denoted by $D_G^(1)(F)$, is defined as

$$D_G^{(1)}(F) = (F \oplus B) \cap G \tag{3.1}$$

The geodesic dilation of size n of F with resplect to G is defined as

$$D_G^{(n)}(F) = D_G^{(1)}\left[D_G^{(n-1)}(F)\right] \tag{3.2}$$

with $D_G^{(0)}(F) = F$. Similarly, the *geodesic erosion* of size 1 of marker F with respect to mask G is defined as

$$E_G^{(1)}(F) = (F \ominus B) \cup G \tag{3.3}$$

The geodesic erosion of size n of F with respect to G is defined as

$$E_G^{(n)}(F) = E_G^{(1)}\left[E_G^{(n-1)}(F)\right] \tag{3.4}$$

with $E_G^{(0)}(F) = F$. Geodesic dilation and erosion are duals with respect to set complementation.

Morphological reconstruction by dilation and by erosion

Morphological reconstruction by dilation of a mask G from a marker F, denoted $R_G^D F$ is defined

$$R_G^D(F) = D_G^{(k)}(F) \tag{3.5}$$

with k such that

$$D_G^{(k)}(F) = D_G^{(k+1)}(F) \tag{3.6}$$

Morphological reconstruction by erosion of mask G from a marker F, denoted $R_G^E(F)$ is defined

$$R_G^E(F) = E_G^{(k)}(F) \tag{3.7}$$

with k such that

$$E_G^{(k)}(F) = E_G^{(k+1)}(F) \tag{3.8}$$

3.3 Task-1 Opening by reconstruction

The opening by reconstruction of size n of an image F is defined as the reconstruction by dilation of F from the erosion of size n of F; that is

$$O_R^{(n)}(F) = R_F^D\left[(F \ominus nB)\right] \tag{3.9}$$

where $(F \ominus nB)$ indicates n erosions of F by B.
Figure 3.1a is the original image for this task. We are interested in extracting the characters that contain long, vertical strokes. The origin image is of size 918 × 2018 pixels. The approximate average height of the tall characters is 50 pixels. Thus we use a structuring element of size 51 × 1 pixels to erode the original image. The erosion image is shown in Figure 3.1b. For comparison, Figure 3.1c shows the opening of Figure 3.1a with the same structuring element. Using erosion image (b) as the marker and the original image (a) as mask, we restored the characters containing long vertical strokes accurately via opening by reconstruction. This result is displayed in Figure 3.1d.

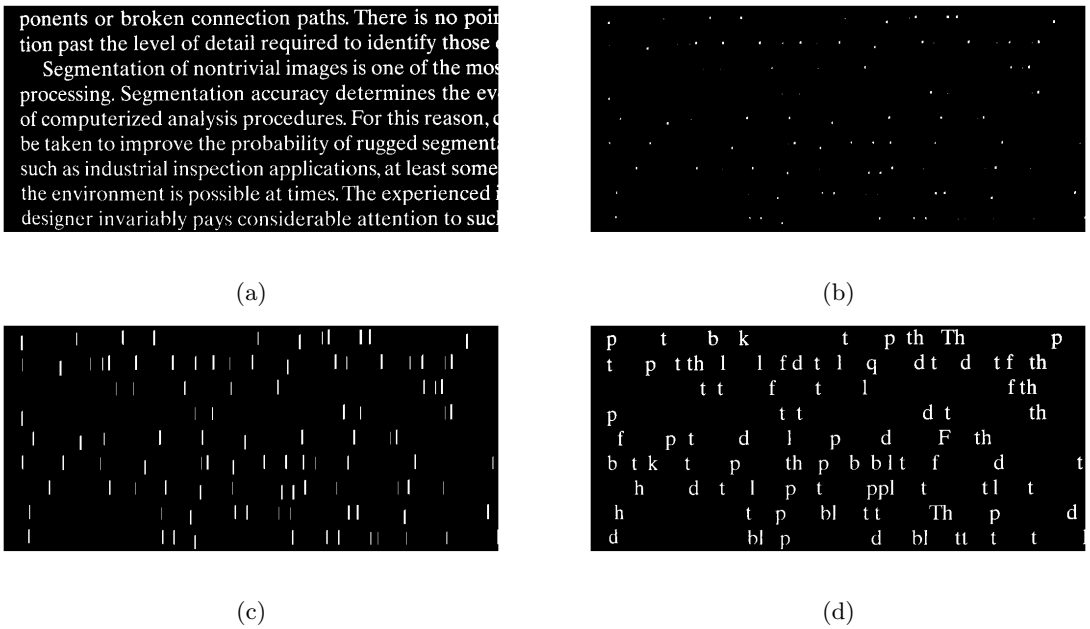
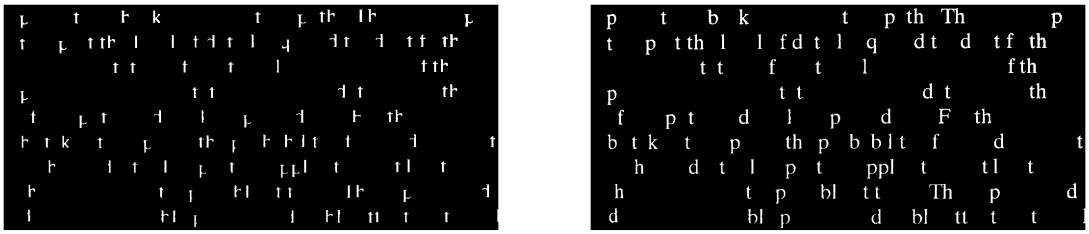


Figure 3.1: Task1-opening by reconstruction. (a)Original binary image 918 × 2018. (b)Erosion of (a) with a structuring element of size 51 × 1 pixels. (c)Opening of(a) with the structuring element 51 × 1, shown for comparison. (d)Result of opening by reconstruction.

The size of geodesic dilation reconstruction is 76. This process cost about 7 minutes. I output some of the intermediate results in Figure 3.2 that help us understand the reconstruction better.



(a) (b)

Figure 3.2: See task-1's process in details. (a)Dilation of size 20. (b)Dilation of size 76, the final results.

3.4 Task-2 Hole filling

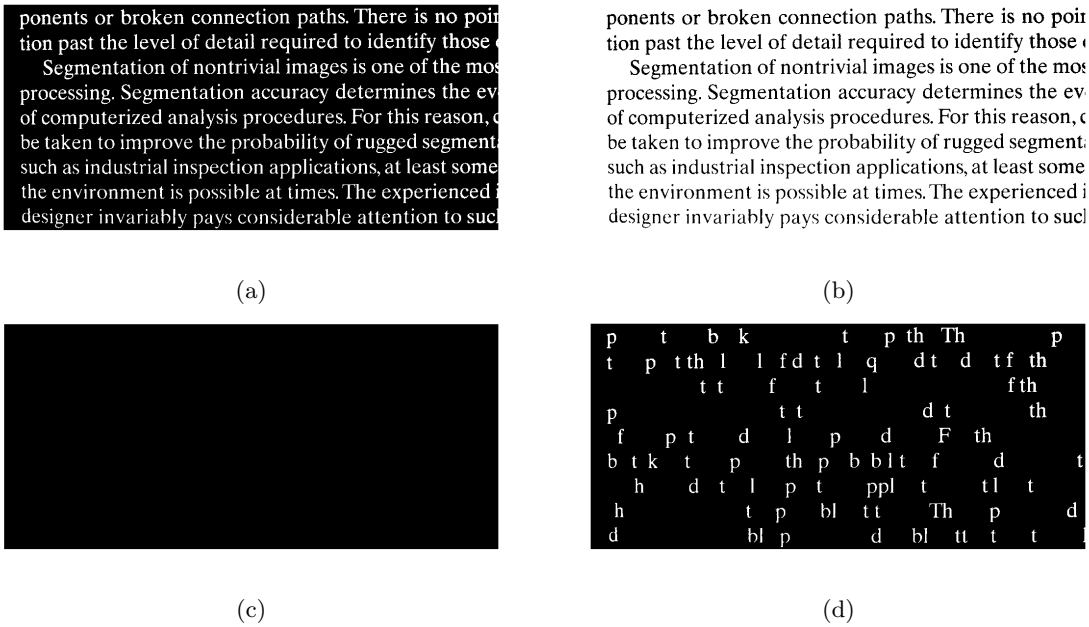
Here we develop a fully automated procedure based on morphological reconstruction. Let $I(x, y)$ denote a binary image and we form a marker F that is 0 everywhere, except at the image border; that is,

$$F(x,y) = \begin{cases} 1 - I(x,y) & \text{if } (x,y) \text{ is on the border of } I \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

Then

$$H = \left[R_{I^c}^D(F)\right]^c \tag{3.11}$$

is a binary image equal to I with all holes filled.
I use 3 structuring element. The hole filling images are shown in Figure 3.3. A detailed process of dilation reconstruction with intermediate results are shown in Figure 3.4. The reconstruction takes 479 steps of dilation, which costs a very long period of about 40 minutes.



(a) (b)

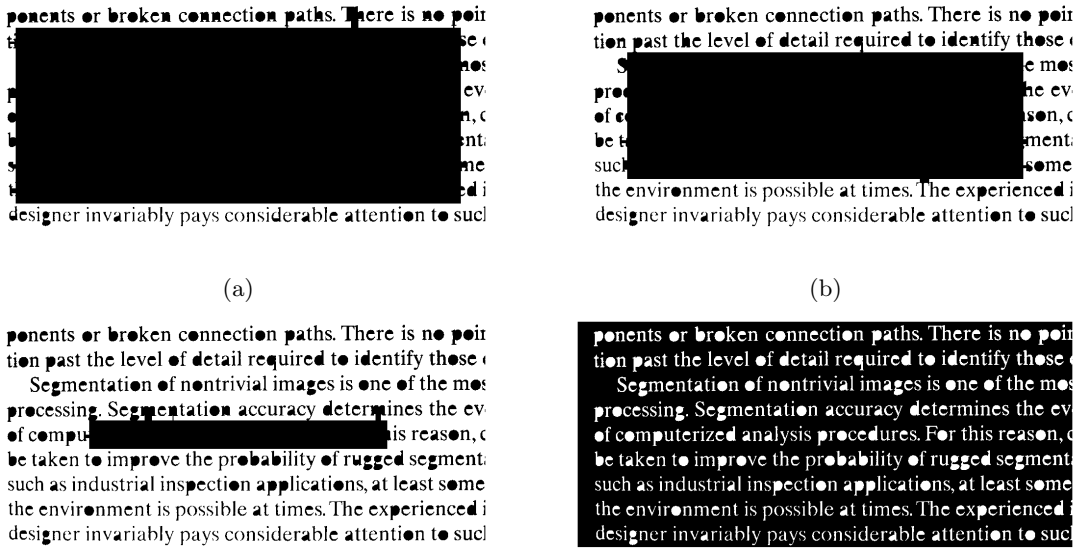
(c) (d)

Figure 3.3: Task2-hole filling. (a)Original binary image 918 × 2018. (b)Complement of (a). Used as a mask. (c)Marker image. Seems like a whole black one, but in fact with some white on border. (d)Result of opening by reconstruction.

3.5 Task-3 Border clearing

Removing objects that touch the border is a useful work because it can screen images so that only complete objects remain for further processing. The marker $F(x, y)$ is defined as:

$$F(x,y) = \begin{cases} I(x,y) & \text{if } (x,y) \text{ is on the border of } I \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$



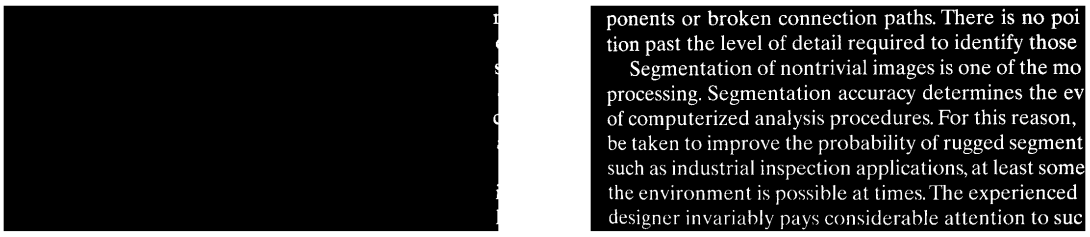
(a) (b) (c) (d)

Figure 3.4: The process of hole filling in details. (a)After 100 steps of dilation. (b)After 200 steps of dilation. (c)After 400 steps of dilation. Much closer to success! (d)The final result with completion.

First we computes morphological reconstruction $R_I^D(F)$ and then computes the desired image X

$$X = I - R_I^D(F) \tag{3.13}$$

I use structuring element of size 3×3. This task is much easier than task 2 and takes just 21 steps of dilation in about 2 minutes. Results are shown in Figure 3.5.



(a) (b)

Figure 3.5: Border clearing. (a)After use border marker for 21-dilation reconstruction. The border letter. (b)The final result after subtraction.

3.6 Implementation

Matlab functions *erosion, dilation, geodesic_dilation, dilation_reconstruction and opening_reconstruction* are implemented for this project. I made a **mistake** on opening_reconstruction at first that I still use 51×1 as the maker while calling dilation_reconstruction. The wrong output image is the same as (c) as all the shorter horizontal adjacent relationship in our wanted characters was damaged. So for correction, we must use 3×3 as structuring element for dilation_reconstruction called in opening_reconstruction.
In the below code frame, I list the key part of these functions. Other process of calling these functions in main script are trivial so omitted here.

```
% key part of function erosion
for i = (1:M-m+1)
    for j = (1:N-n+1)
        x = imgf(i:i+m-1, j:j+n-1);
        if sum(sum(x.*B)) == m*n
```

```
            g(i+downshift, j+rightshift) = 1;
        end
    end
end

% key part of dilation_reconstruction
k_times = 0;
while( ~isequal(f0, f1) )
    k_times = k_times + 1;
    f0 = f1;
    f1 = geodesic_dilation(f1, G, B);
end

% key part of geodesic_dilation
imgg = dilation(imgf, B) & G;

% key part of opening_reconstruction
for i=(1:n_size)
    f_erosion = erosion(f_erosion, B);
end
[imgg, k_times] = dilation_reconstruction(f_erosion, imgf, ones(3,3)); % here
    can not use ones(51, 1)
```

3.7 Discussion

These three tasks show the basic idea of morphological reconstruction used in feature extraction. We start from a marker which can be obtained easily. Then we use the mask as the constrain to conduct set operations. More advanced topics are morphological operations on gray intensity images and image segmentation. This is a very interesting subproject. The main obstacle here is **time**! I tried to do much vectorization but still have a big problem that 20-step-dilation cost about 2 minutes on the 918×2018 image but the complexity is only about $918 \times 2018 \times 20 \times 9 \approx 4 \times 10^8$. I think the computation of this complexity need no more than 2 sec on language like C++ or python. However, I tried the matlab toolbox *IPT* and it's just as fast as the expectation (within 2 sec). This is a strange but interesting problem.

4 Project 9 - Image Segmentation

4.1 Project Proposal

There tow parts of project 9. One task is for edge detection: implement the Roberts, Prewitt, Sobel, the Marr-Hildreth and the Canny edge detectors. The test image is *building.tif*. The other task is to implement the Otsus method of thresholding segmentation, and compare the results with the global thresholding method using test image *polymersomes.tif*.

4.2 Preliminaries

4.2.1 Edge detection

The central idea of edge detection is that local changes in intensity can be detected using derivatives. We have the following conclusions which show that first- and second-order derivatives are particularly well suited for this purpose: (1) First-order derivatives generally produce thicker edges in an image. (2) Second-order derivatives have a stronger response to fine details such as thin lines, isolated points, and noise. (3) Second-order derivatives produce a double-edge response at ramp and step transitions in intensity. (4) The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.

4.2.2 Basic edge detection

The tool of choice for finding edge strength and direction at location (x, y) of an image f is the gradient.

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{4.1}$$

The magnitude of vector ∇f defined as

$$M(x, y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y| \tag{4.2}$$

is the value of the rate of change in the direction of the gradient vector. The second $=$ is a frequently used approximate to avoid square roots. The direction of gradient vector is given by the angle

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right] \tag{4.3}$$

The direction of an edge at an arbitrary point (x, y) is orthogonal to the direction $\alpha(x, y)$. Here we mention three masks(Figure 4.1) Roberts, Prewitt and Sobel which can be used to compute the gradient of the center point with convolution operation.

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel

Figure 4.1: 3 masks for gradient computation: Roberts, Prewitt, Sobel

4.2.3 More advanced techniques for edge detection