Lab 8

1. Define two structural elements, perform binary dilation, erosion, opening, and closing operations on noisy_fingerprint.pgm and noise_rectangle.pgm
2. Extract the boundaries of licoln.pgm and U.pgm
3. Count the number of pixels in each white connected component from connected.pgm and put them into a .txt file
4. Problem 9.36, write a program to separate the three required sets from bubbles_on_black_background.pgm

Algorithm:

1. Dilation|

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

Input: array of image f
for(i=0; i<size of image;i++){
if there is one pixel value =255 within the structural element
image[i]=255
return image

2. Erosion

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

Input: array of image f
for(i=0; i<size of image;i++){
if there is one pixel value =0 within the structural element
image[i]=0
return image

3. Opening
First conduct erosion operation and then dilation operation
Input: array of image f
outimage=Erosion(image)
outimage=Dilation(outimage)
return outimage

4. Closing
First conduct dilation operation and then erosion operation
Input: array of image f
outimage= Dilation(image)
outimage= Erosion(outimage)
return outimage

5. Boundary Extraction
Input: array of image f
erosionimage = Erosion(outimage)
outimage=image-erosionimage
return outimage

6. Count the number of pixels in each white connected component
Denote the input image is A0, the iteration matrix is Xk, and the connected component storage matrix is B

A=A0

```
    while(there is still white dot in A){
    find the point in A , the grey value of which is 1
    select the first point whose grey value is 1 as the initial point
    find the intersection after dilation
    if(Xk==Xk-1) ends iteration
    B=B+Xk
    A=A-B
}
```

7. Extract Boundary-Overlapping-Only Particles

   Input: the array of image

```
temp=0 array with length of image size
temp[0]=255
while (true) {
    Xk = temp
    memcpy(Xk, temp, size);
    temp = Dilation(temp);
    for (i = 0; i < size of image; i++) {
        if (temp[i] > image[i]) {
            temp[i] = image [i];
        }
    }
    if (Xk is the same as temp) ends the iteration
}
outimage =temp
return outimage;
```

8. Extract Particle-Overlapping-Only Particles

   Input: the array of image

   A=the area of one particle.

   1. First depends on the result of Boundary-Overlapping-Only method to remove those particles on the boundary
   2. Then depends on the result of the connected component algorithm, count the number of pixels in each component. If the number of pixels in current component is less than A+ε, where ε is a small error that can be tolerated, than remove it.

9. Extract None-Overlapping Particles

   Input: the array of image

```
bondraryoverlapping=boundary-Overlapping-Only(image)
particleoverlapping= Particle-Overlapping-Only(image)
outimage=image- bondraryoverlapping- particleoverlapping
```

Results (compare the results with the original image):

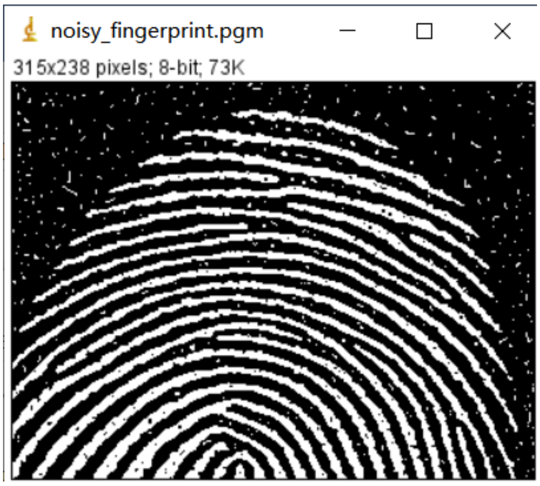
1. Here use 8-neighbour and 4-connected structural elements in task 1.
   a) Noisy_rectangle.pgm

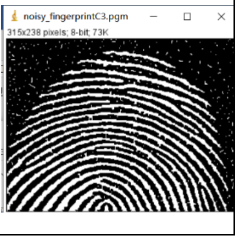
	Dilation	Erosion
8-neighbour		
4-connected		

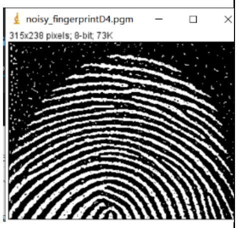
	Opening	Closing

8-neighbour		
4-connected		

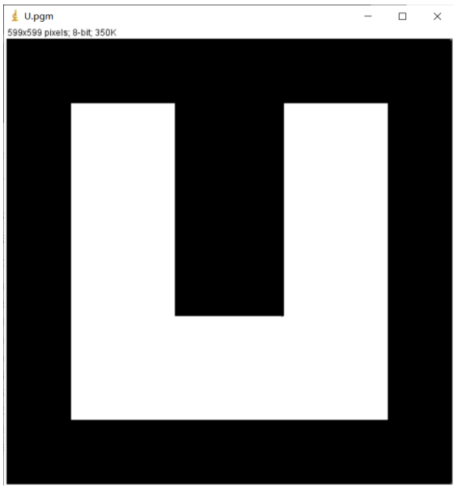
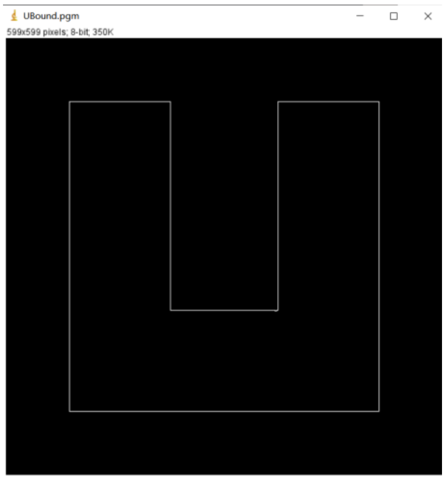
b) Noise_fingerprint.pgm



	Dilation	Erosion	Opening	Closing
8-neigh bour				

4-connected				

2. Task2

	Original	Boundary Extraction
Licoln.pgm		
U.pgm		

3. Task 3

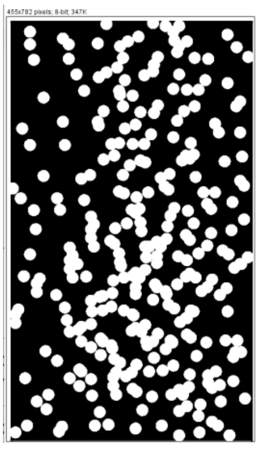
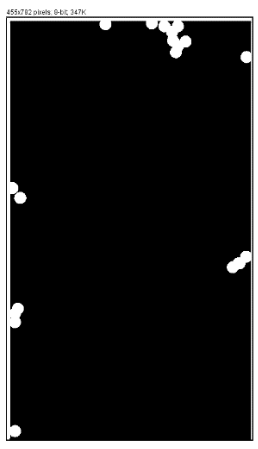
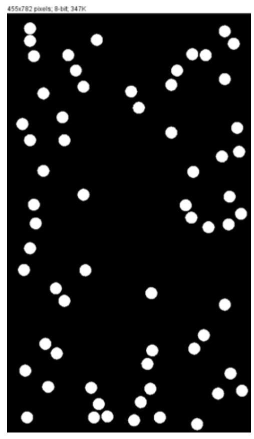
```
connectedres.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
The label of the 0th pixel is 16
The label of the 1th pixel is 2
The label of the 2th pixel is 16
The label of the 3th pixel is 1
The label of the 4th pixel is 6
The label of the 5th pixel is 16
The label of the 6th pixel is 81
The label of the 7th pixel is 2
The label of the 8th pixel is 100
The label of the 9th pixel is 16
The label of the 10th pixel is 90
The label of the 11th pixel is 20
The label of the 12th pixel is 1
The label of the 13th pixel is 2
The label of the 14th pixel is 16
The label of the 15th pixel is 16
The label of the 16th pixel is 144
The label of the 17th pixel is 380
The label of the 18th pixel is 2
The label of the 19th pixel is 2
The label of the 20th pixel is 16
The label of the 21th pixel is 2
The label of the 22th pixel is 144
The label of the 23th pixel is 20
The label of the 24th pixel is 49
The label of the 25th pixel is 16
The label of the 26th pixel is 1
The label of the 27th pixel is 16
The label of the 28th pixel is 81
The label of the 29th pixel is 42
The label of the 30th pixel is 42
The label of the 31th pixel is 23
The label of the 32th pixel is 2
The label of the 33th pixel is 12
The label of the 34th pixel is 1
The label of the 35th pixel is 16
```

```
connectedres.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
The label of the 36th pixel is 2
The label of the 37th pixel is 2
The label of the 38th pixel is 16
The label of the 39th pixel is 100
The label of the 40th pixel is 2
The label of the 41th pixel is 2
The label of the 42th pixel is 24
The label of the 43th pixel is 1
The label of the 44th pixel is 156
The label of the 45th pixel is 90
The label of the 46th pixel is 16
The label of the 47th pixel is 90
The label of the 48th pixel is 36
The label of the 49th pixel is 42
The label of the 50th pixel is 12
The label of the 51th pixel is 2
The label of the 52th pixel is 1
The label of the 53th pixel is 1
The label of the 54th pixel is 13
The label of the 55th pixel is 16
The label of the 56th pixel is 400
The label of the 57th pixel is 144
The label of the 58th pixel is 49
The label of the 59th pixel is 144
The label of the 60th pixel is 16
The label of the 61th pixel is 144
The label of the 62th pixel is 2
The label of the 63th pixel is 42
The label of the 64th pixel is 2
The label of the 65th pixel is 1
The label of the 66th pixel is 2
The label of the 67th pixel is 16
The label of the 68th pixel is 81
The label of the 69th pixel is 42
The label of the 70th pixel is 81
The label of the 71th pixel is 16
```

```
connectedres.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
The label of the 72th pixel is 50
The label of the 73th pixel is 100
The label of the 74th pixel is 2
The label of the 75th pixel is 42
The label of the 76th pixel is 2
The label of the 77th pixel is 20
The label of the 78th pixel is 81
The label of the 79th pixel is 1
The label of the 80th pixel is 16
The label of the 81th pixel is 49
The label of the 82th pixel is 2
The label of the 83th pixel is 7
The label of the 84th pixel is 90
The label of the 85th pixel is 16
The label of the 86th pixel is 90
The label of the 87th pixel is 16
The label of the 88th pixel is 16
The label of the 89th pixel is 2
The label of the 90th pixel is 24
The label of the 91th pixel is 1
The label of the 92th pixel is 42
The label of the 93th pixel is 16
The label of the 94th pixel is 49
The label of the 95th pixel is 1
The label of the 96th pixel is 1
The label of the 97th pixel is 144
The label of the 98th pixel is 1
The label of the 99th pixel is 1
The label of the 100th pixel is 400
The label of the 101th pixel is 90
The label of the 102th pixel is 16
The label of the 103th pixel is 1
The label of the 104th pixel is 24
The label of the 105th pixel is 1
The label of the 106th pixel is 16
The label of the 107th pixel is 16
```

```
connectedres.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
The label of the 86th pixel is 90
The label of the 87th pixel is 16
The label of the 88th pixel is 16
The label of the 89th pixel is 2
The label of the 90th pixel is 24
The label of the 91th pixel is 1
The label of the 92th pixel is 42
The label of the 93th pixel is 16
The label of the 94th pixel is 49
The label of the 95th pixel is 1
The label of the 96th pixel is 1
The label of the 97th pixel is 144
The label of the 98th pixel is 1
The label of the 99th pixel is 1
The label of the 100th pixel is 400
The label of the 101th pixel is 90
The label of the 102th pixel is 16
The label of the 103th pixel is 1
The label of the 104th pixel is 24
The label of the 105th pixel is 1
The label of the 106th pixel is 16
The label of the 107th pixel is 16
The label of the 108th pixel is 1
The label of the 109th pixel is 2
The label of the 110th pixel is 144
The label of the 111th pixel is 2
The label of the 112th pixel is 24
The label of the 113th pixel is 16
The label of the 114th pixel is 81
The label of the 115th pixel is 1
The label of the 116th pixel is 2
The label of the 117th pixel is 2
The label of the 118th pixel is 2
The label of the 119th pixel is 16
```

4. Task4

			
Origin	Boundray overlap only	particle overlap only	None overlapping

Discussion:

1. In task 1, the performance of 4-connected structural element is worse than the 8-neighbour structural element.
2. From the result of task 1, we can see that the erosion operation helps to remove noise. erosion operations on low-size structural elements easily remove scattered salt and pepper noise points. And the dilation operation helps to connect adjacent elements in the image.
3. Opening operation: erosion first, then dilation, can remove some bright pixels, magnify local low brightness areas, separate objects at thin points, and smooth the boundary of larger objects while not significantly changing its area.
4. Closed operation: dilation, then erosion, can remove some dark pixels, i.e. exclude small black

holes (black areas)

5. During the process of boundary extraction, when a 3x3 1-value structural element is used, the extracted boundary value is 1 pixel. Just as the result of previously shown.
6. The connected component analysis is useful to the following image separation and segmentation. The algorithm that I implemented in this lab has a low efficiency because there are too many iterations.
7. In Task 5, the operation of separating the image is the combination of dilation, connected component analysis operation.

Codes:

1. Dilation

```
for (i = 1; i < image->Height-1; i++) {
    for (j = 1; j < image->Width-1; j++) {
        if (flag == 1) {
            for (p = -1; p < 2; p++) {
                for (q = -1; q < 2; q++) {
                    if (tempin[(i + p) * image->Width + (j + q)] == 255) {
                        temp[i * image->Width + j] = 255;
                        break;
                    }
                }
            }
        }
        else {
            if (tempin[(i - 1) * image->Width + j ] == 255) {
                temp[i * image->Width + j] = 255;
                break;
            }
            else if(tempin[(i + 1) * image->Width + j] == 255){
                temp[i * image->Width + j] = 255;
                break;
            }
            else if (tempin[i * image->Width + (j - 1)] == 255) {
                temp[i * image->Width + j] = 255;
                break;
            }
            else if (tempin[i * image->Width + (j + 1)] == 255) {
                temp[i * image->Width + j] = 255;
                break;
            }
        }
    }
}
```

2. Erosion

```
for (i = 1; i < image->Height - 1; i++) {
    for (j = 1; j < image->Width - 1; j++) {
        if (flag == 1) {
            for (p = -1; p < 2; p++) {
                for (q = -1; q < 2; q++) {
                    if (tempin[(i + p) * image->Width + (j + q)] == 0) {
                        temp[i * image->Width + j] = 0;
                        break;
                    }
                }
            }
        }
        else {
            if (tempin[(i - 1) * image->Width + j] == 0) {
                temp[i * image->Width + j] = 0;
                break;
            }
            else if (tempin[(i + 1) * image->Width + j] == 0) {
                temp[i * image->Width + j] = 0;
                break;
            }
            else if (tempin[i * image->Width + (j - 1)] == 0) {
                temp[i * image->Width + j] = 0;
                break;
            }
            else if (tempin[i * image->Width + (j + 1)] == 0) {
                temp[i * image->Width + j] = 0;
                break;
            }
        }
    }
}
```

3. Opening

```
Image* Opening(Image* image, int flag) {
    Image* outimage;
    outimage = Erosion(image, flag);
    outimage = Dilation(outimage, flag);
    return outimage;
}
```

Closing

4. Closing

```
Image* Closing(Image* image, int flag) {
    Image* outimage;
    outimage = Dilation(image, flag);
    outimage = Erosion(outimage, flag);
    return outimage;
}
```

5. Boundary Extraction

```
Image* BoundaryExtraction(Image* image) {
    //With 3*3 structural element
    int i;
    unsigned char* tempin, * tempout;
    Image* outimage;
    int* temp = (int*)malloc(sizeof(int) * image->Height * image->Width);

    tempin = image->data;
    outimage = Erosion(image, 1);
    tempout = outimage->data;

    for (i = 0; i < image->Height * image->Width; i++) {
        tempout[i] = tempin[i] - tempout[i];
    }

    return outimage;
}
```

6. Count the number of pixels in each white connected component

```
while (FindPoint(A, pixel, 255 , image->Height, image->Width)) {
    X1[pixel[0].y * image->Width + pixel[0].x] = 255;

    while (1) {
        unsigned char *Xk = (unsigned char*)malloc(sizeof(unsigned char) * size);
        memcpy(Xk, X1, size);
        X1 = DilationArr(X1, image->Height, image->Width);
        for (int i = 0; i < size; i++) {
            if (X1[i] > A[i]) {
                X1[i] = A[i];
            }
        }
        if (memcmp(Xk, X1, size) == 0) {
            break;
        }
        free(Xk);
    }
    num[count] = FindPoint(X1, pixel , 255, image->Height, image->Width);

    fprintf(fp, "The label of the %dth pixel is %d\n", count, num[count]);


    for (i = 0; i < size; i++) {
        if (B[i] < X1[i]) {
            B[i] = X1[i];
        }
    }
    memset(X1, 0, sizeof(unsigned char) * size);
    for (i = 0; i < size; i++) {
        A[i] -= B[i];
        if (A[i] < 100) {
            A[i] = 0;
        }
    }
    count++;
}
```

7. Extract Boundary-Overlapping-Only Particles

```
temp[0] = 255;

while (true) {
    unsigned char* Xk = (unsigned char*)malloc(sizeof(unsigned char) * image->Height * image->Width);
    memcpy(Xk, temp, size);
    temp = DilationArr(temp, image->Height, image->Width);
    for (i = 0; i < size; i++) {
        if (temp[i] > tempin[i]) {
            temp[i] = tempin[i];
        }
    }
    if (memcmp(Xk, temp, size) == 0){
        break;
    }
    free(Xk);
}
```

8. Extract Particle-Overlapping-Only Particles

```
    while (1) {
        unsigned char* Xk = (unsigned char*)malloc(sizeof(unsigned char) * size);
        memcpy(Xk, X1, size);
        X1 = DilationArr(X1, image->Height, image->Width);
        for (int i = 0; i < size; i++) {
            if (X1[i] > A[i]) {
                X1[i] = A[i];
            }
        }
        if (memcmp(Xk, X1, size) == 0) {
            break;
        }
        free(Xk);
    }
    num[count] = FindPoint(X1, pixel, 255, image->Height, image->Width);

    if (num[count] > n) {
        for (int i = 0; i < size; i++) {
            if (X2[i] < X1[i]) {
                X2[i] = X1[i];
            }
        }
    }
    count++;
    for (int i = 0; i < size; i++) {
        if (B[i] < X1[i]) {
            B[i] = X1[i];
        }
    }
    memset(X1, 0, sizeof(unsigned char) * size);
    for (int i = 0; i < size; i++) {
        A[i] -= B[i];
        if (A[i] < 100) {
            A[i] = 0;
        }
    }
}
for (int i = 0; i < size; i++) {
    tempout[i] = X2[i];
```

8. Extract None-Overlapping Particles

```
Image* NoneOverlapping(Image* image) {
    int size = image->Height * image->Width;
    Image* WithP, * WithB;
    Image* outimage = CreateNewImage(image, "NoneOverlapping", image->Width, image->Height);

    WithB = BoundaryOverlappingOnly(image);
    WithP = ParticalOverlappingOnly(image);

    unsigned char* tempin = image->data;
    unsigned char* temp1 = WithB->data;
    unsigned char* temp2 = WithP->data;
    unsigned char* tempout = outimage->data;


    for (int i = 0; i < size; i++) {
        tempout[i] = tempin[i] - temp1[i] - temp2[i];
    }

    return outimage;
}
```