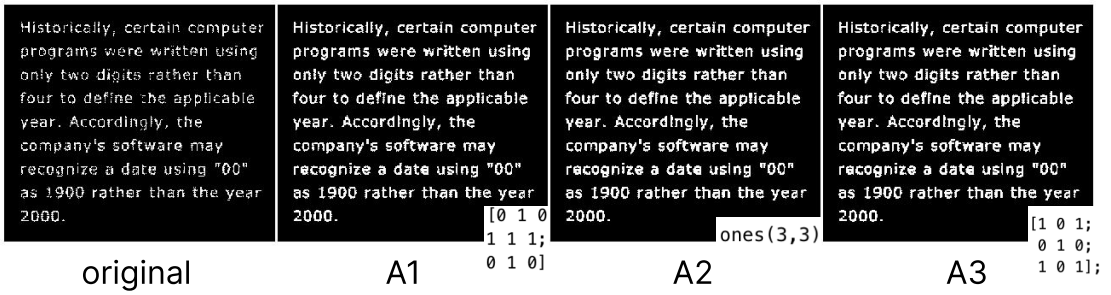
Task 1

Dilation

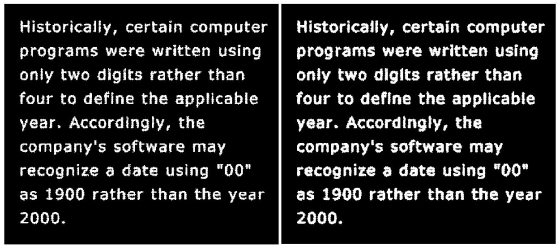
Dilation Operation



original A1 A2 A3

(SE of all ones) (SE diagonal cross)

The result pictures wasn't significantly different from each other, changing SE to all ones(A2) would make straight lines look better after dilation, like all the "f". A3 makes more



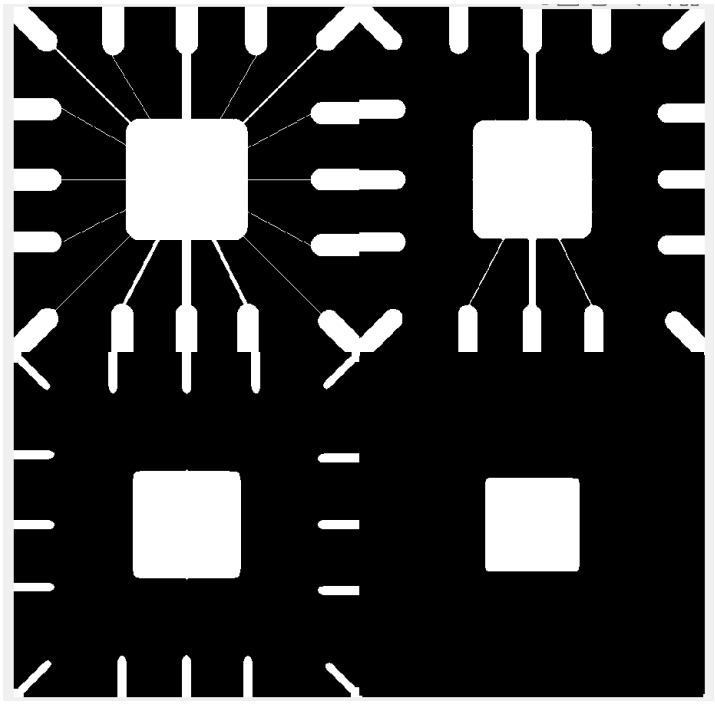
original Dilate twice with B1

```
A = imread('assets/text-broken.tif');
B1 = [0 1 0;
      1 1 1;
      0 1 0];    % create structuring element
B2 = ones(3,3);     % generate a 3x3 matrix of 1's
Bx = [1 0 1;
       0 1 0;
       1 0 1]; %Try to make the SE diagonal cross
A1 = imdilate(A, B1);
A2 = imdilate(A, B2);
A3 = imdilate(A, Bx);
A4 = imdilate(A1, B1);%dilate twice
montage({A,A1,A2,A3},'Size', [1 4])
%montage({A1,A4})
```

Dilate twice will add on the effects and creating thicker letters

Erosion

```
clear all
close all
A = imread('assets/wirebond-mask.tif');
SE2 = strel('disk',2);
SE10 = strel('disk',10);
SE20 = strel('disk',20);
E2 = imerode(A,SE2);
E10 = imerode(A,SE10);
E20 = imerode(A,SE20);
montage({A, E2, E10, E20}, "size", [2 2])
```



Increase the size of SE will cause more features to be removed

E2 erose thin lines with width smaller than 2, while E20 erose all the line segments around the picture, as shown in fig2

Task 2 Morphological Filtering

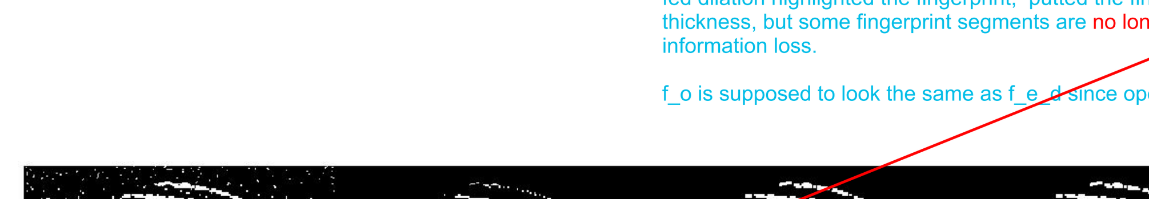
```
clear all
close all
f = imread('assets/fingerprint-noisy.tif');
SE = strel("square",3);
fe = imerode(f,SE);
fed = imdilate(fe, SE);
fo = imopen(f,SE);
montage({f,fe,fed,fo},'Size', [1 4]);
fo_ = imclose(fo,SE);
imshow(fo_);

w_gauss = fspecial('Gaussian', [7 7], 1.0);
g_gauss = imfilter(f, w_gauss, 0);
figure
imshow(g_gauss)
```

1. Read the image file 'finger-noisy.tif' into f.
2. Generate a 3x3 structuring element SE.
3. Erode f to produce f_e.
4. Dilate fe to produce f_e_d.
5. Open f to produce f_o.
6. f, f_e, f_e_d and f_o as a 4 image montage are shown on fig.1

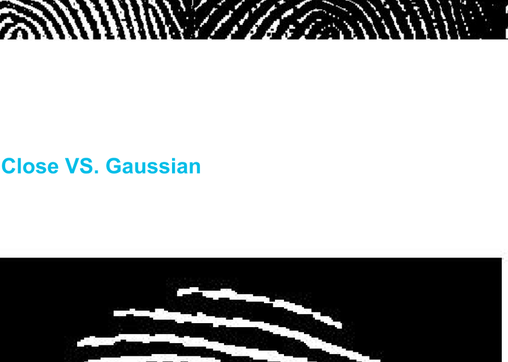
Comments:
f_e is erosed with structuring element SE3 that eliminate almost all of the noise feature in background

fed dilation highlighted the fingerprint, putted the fingerprint back to its original thickness, but some fingerprint segments are no longer continuos due to information loss.

f_o is supposed to look the same as f_e_d since opening=erosion+dilation



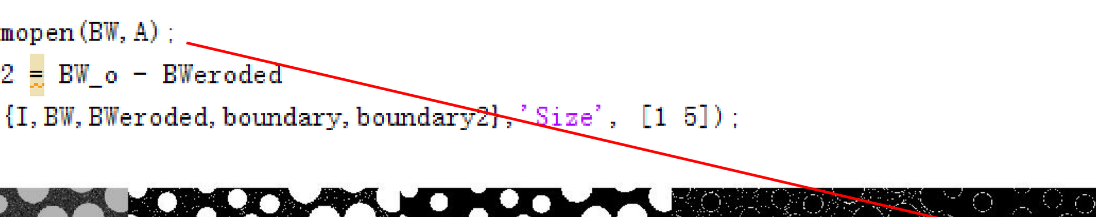
Opening and Close VS. Gaussian

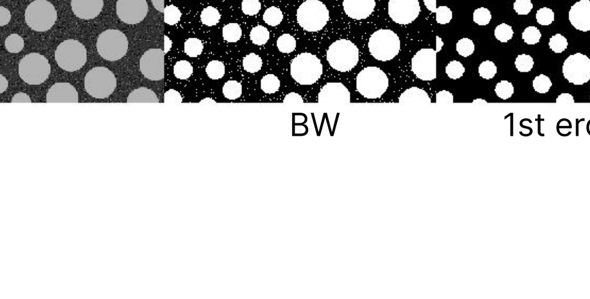


Generally, Gauusian filter can't fully eliminate the background noise, but the fingerprint lines are preserved better

Task 3 - Boundary detection

```
clear all
close all
I = imread('assets/blobs.tif');
I = imcomplement(I);
level = graythresh(I);
A = ones(3,3);      % generate a 3x3 matrix of 1's
BW = imbinarize(I, level);

BWeroded = imerode(BW,A);
boundary = BW - BWeroded
montage({I,BW,BWeroded,boundary},'Size', [1 4]);

BW_o = imopen(BW,A);
boundary2 = BW_o - BWeroded
montage({I,BW,BWeroded,boundary,boundary2},'Size', [1 5]);
```





Opening operation eliminates the background noise while preserving most of the details.

Task 4 - Function bwmorph - thinning and thickening

```
clear all
close all
f = imread('assets/fingerprint.tif');

level = graythresh(f);
BW = imbinarize(f, level);

for n = 1:5
    % Apply thinning operation with n iterations
    g = bwmorph(BW, 'thin', n);

    % Create a subplot for each thinned image
    subplot(1, 5, n); % Arranges plots in 1 row, 5 columns
    imshow(g); % Display the thinned image
    title(['n = ', num2str(n)]); % Title each subplot with the iteration number
end

sgtitle('Thinning Operations with Different Iterations'); % Super title for the whole figure

infinite = bwmorph(BW,'thin', inf);
imshow(infinite);
```

fig. 2

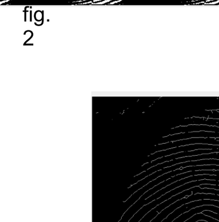
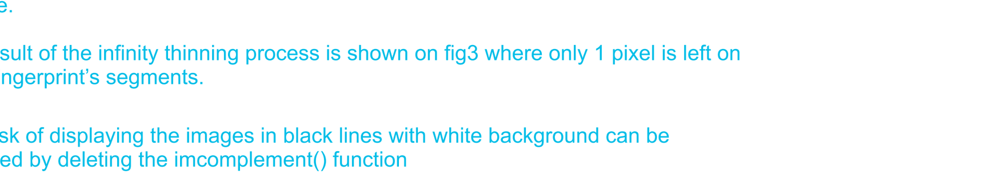


fig.3

By turning n into infinity, the function will repeat for infinity times until the image doesn't change.

The result of the infinity thinning process is shown on fig3 where only 1 pixel is left on each fingerprint's segments.

The task of displaying the images in black lines with white background can be achieved by deleting the imcomplement() function




```
clear all
close all
f = imread('assets/fingerprint.tif');
f = imcomplement(f);
level = graythresh(f);
BW = imbinarize(f, level);

for n = 1:5
    % Apply thinning operation with n iterations
    g = bwmorph(BW, 'thick', n);

    % Create a subplot for each thinned image
    subplot(1, 5, n); % Arranges plots in 1 row, 5 columns
    imshow(g); % Display the thinned image
    title(['n = ', num2str(n)]); % Title each subplot with the iteration number
end

sgtitle('Thickening Operations with Different Iterations'); % Super title for the whole figure
```

Thickening process the fingerprint's segment to have more pixels on each, as shown in fig6

fig.6

inverted

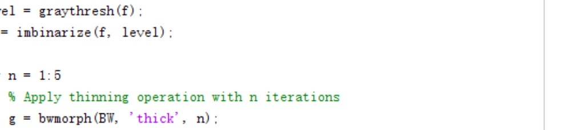


fig.7

An interesting finding is that the results of thickening the complement pictures(Black on white) have the same result image as thinning the original picture(white on black)
the 2 same results prove that the fuction only operates on white pixels, as it thicken the white pixel on the complement picture, it "squeeze" the space of the black pixels and therefore thining the finger prints.

Task 5 Connected Components and labels

```
t = imread('assets/text.png');
imshow(t)
CC = bwconncomp(t)

numPixels = cellfun(@numel, CC.PixelIdxList);
[biggest, idx] = max(numPixels);
t(CC.PixelIdxList{idx}) = 0;
figure
imshow(t)
```

- CC is a data structure that:

```
Connectivity: 8
   ImageSize: [256 256]
  NumObjects: 88
PixelIdxList: {1×88 cell}
```

- A connected component is a set of adjacent pixels in a binary image.
- Two possible definitions of what is "connected":
  - 4-connectivity — Pixels are connected if their edges touch.
  - 8-connectivity — Pixels are connected if their edges or corners touch.

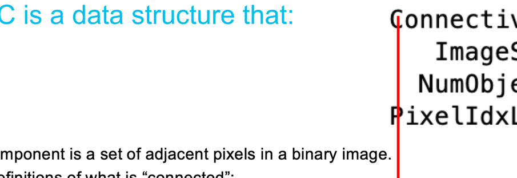


fig.2

8-connectivity means it considered both edges & corner touch when defining the connected components, as shown in figure 2

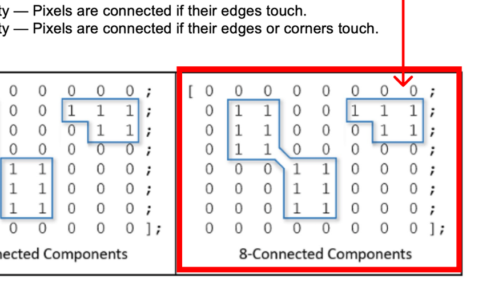


fig.3: CC.PixelIdxlist → cellfun(@numel,) → fig.4: numPixels

```
numPixels =

 列 1 至 18

    66    41    63    80    53    63    63    41    48    61    63    41    48   105    66    68    92    41

 列 19 至 36

    68     6    33    85    48    63   100    41    63    61    80    85     6     6    85    48    61    68

 列 37 至 54

    63    41    68    61    63    33    54    33    80     9     9     9    63    68    48   106     6    85

 列 55 至 72

    85    85     6    85    85    72    63    41    63    33    74    63    72    33    68    41    48     9

 列 73 至 88

    48     6     9    61   105    63    61    74    61    41    63    54    33    41     9     9
```

numPixels, which is an array containing the number of pixels in each of the detected connected components in the image. Just like the table in Lecture 6 slide 24, as shown in figure 5.

Connected component	No. of pixels in connected comp
01	11
02	9
03	9
04	39
05	133
06	1
07	1
08	743
09	7
10	11
11	11
12	9
13	9
14	674
15	85

fig.5

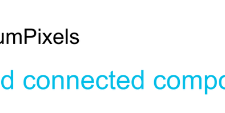
```
t = imread('assets/text.png');
imshow(t)
CC = bwconncomp(t);

numPixels = cellfun(@numel, CC.PixelIdxList);
[biggest, idx] = max(numPixels);
t(CC.PixelIdxList{idx}) = 0;
figure
imshow(t)
```

fig.6

The rest of the code is find the maximum value in the array, which represent the longest connected component, and then illuminate it by changing the value into zero.

The exact process would be change 106 to 0, as 106 is the max value.

After removing it from t, "ff" disappears as it is the longest one: the two "f"s connects each other in the picture.



Task 6 - Morphological Reconstruction

- MR is a better method that restores the original shapes of the objects that remain after erosion.

```
clear all
close all
f = imread('assets/text_bw.tif');
se = ones(17,1);
g = imerode(f, se);
%one erosion with the structuring element SE.
% The locations of the tall characters are extracted successively.
fo = imopen(f, se);     % perform open to compare
%simply dilating an eroded image does not always restore the original, fo
%is an example
fr = imreconstruct(g, f);
montage({f, g, fo, fr}, "size", [2 2])

%By eroding the image with a thin structuring element of size 17 × 1,
%we should be able to isolate characters that have an height of 17 pixels.

ff = imfill(f);
figure
montage({f, ff})
```

fig.1

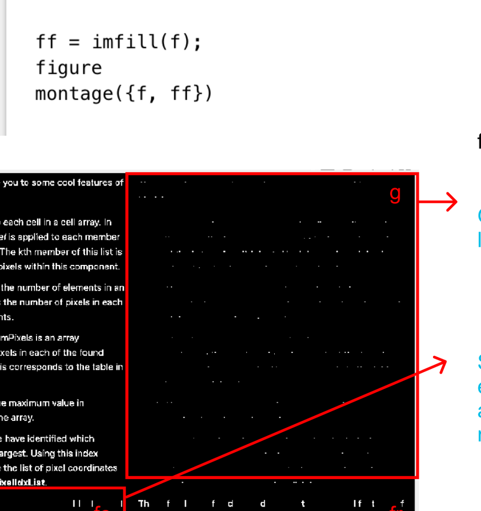


fig.2

One erosion with the structuring element SE ones(17,1) generally erodes all the tall letters

Simply dilating an eroded image does not always restore the original, fo is an example.
as the structuring element SE is a 17 x 1 matrix, part of the tall characters are reconstructed, as shown on fo.

```
fr = imreconstruct(g, f);
```

Apply the reconstruction operation using Matlab's imreconstruct function between the **marker g** and the **mask f**.

The MR method managed to restore the original picture much better as it has its comparison mechanism

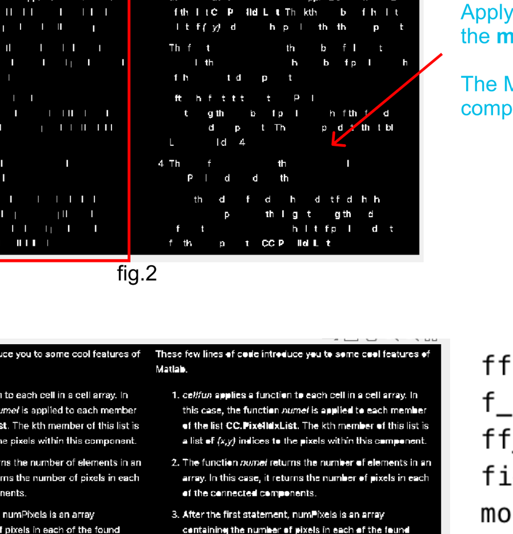


fig.3

```
ff = imfill(f);
f_c = imcomplement(f);
ff_c = imfill(f_c);
figure
montage({f, ff, f_c, ff_c}, "size", [2 2])
```

The hole in the characters "o", "p", "d", "a", "e", "b", and the number "4", are filled with white pixels

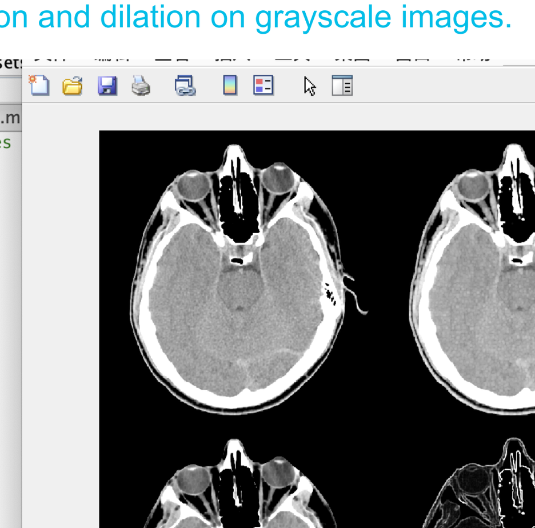
an interesting finding is still that: the matlab function only fill the holes that are surrounded by white pixels.

When I'm testing the function with complement picture (black characters white background), the function instead fill all the black characters that are surrounded by the white pixels (background). It results in an completely white pixel image as shown in fig3, ff_c.

Task 7 - Morphological Operations on Grayscale images

- in this task, we explore the effect of erosion and dilation on grayscale images.

```
% Task 7 - Morphological Operations on Grayscale images
clear all
close all
f = imread('assets/headCT.tif');
se = strel('square',3);
gd = imdilate(f, se);
ge = imerode(f, se);
gg = gd - ge;
montage({f, gd, ge, gg}, 'size', [2 2])
```



As before, it seems that the dilating and eroding fuctions are mainly focusing on the white area of the pic, this stays the same in the greyscale pic. But when we try to obtain the boundaries in binary pics, we will only get the outermost contour. In the case of the grey scale image, the contours of the internal structures of the brain is still visible, giving us more informations form the original pic.
This occurs because both the dilation and erosion have altered the pixel values at the edges of these internal structures, making their boundaries more pronounced when the difference is computed. This effect allows for the visualization of internal details in medical imaging, aiding in the identification of structures and potential abnormalities within the brain.

Challenge2:

The file 'assets/palm.tif' is a palm print image in grayscale. Produce an output image that contains the main lines without all the underlining non-characteristic lines.

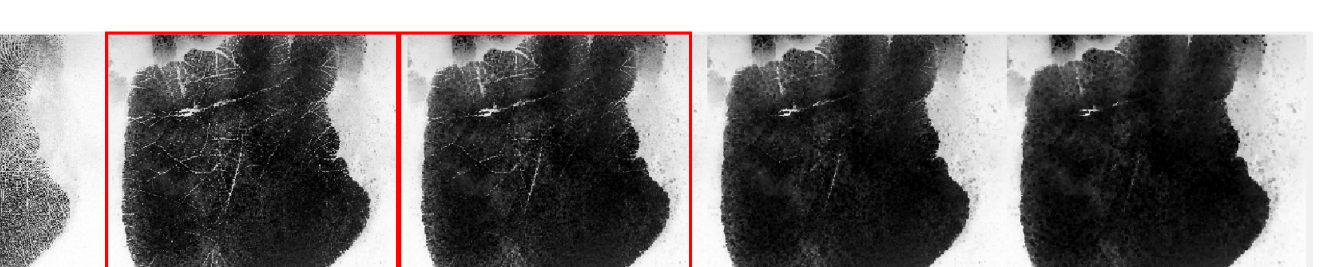


fig.1

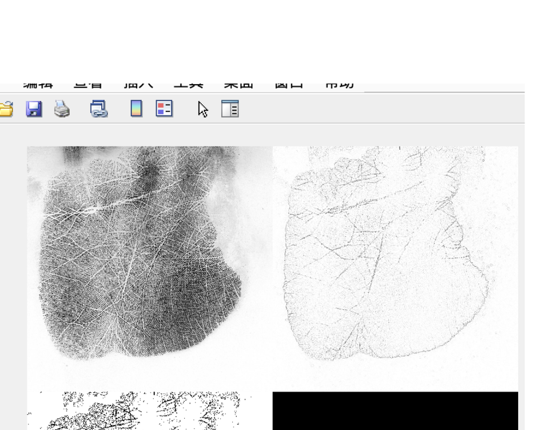
```
A = imread('assets/palm.tif');
SE5 = strel('disk',5);
SE6 = strel('disk',6);
SE8 = strel('disk',8);
SE10 = strel('disk',10);

E5 = imerode(A,SE5);
E6 = imerode(A,SE6);
E8 = imerode(A,SE8);
E10 = imerode(A,SE10);
montage({A, E5, E6, E8, E10}, "size", [1,5])
```

The structuring element of size 5,6,8,10 are used in the erosion testing to identify which is suitable for the presenting clear information of the characteristic lines.

As shown in fig1, on ideal one would be SE5 or SE6.

```
E6 = imerode(A,SE6);
E8 = imerode(A,SE8);
E10 = imerode(A,SE10);
montage({A, E5, E6, E8, E10}, "size", [1,5])
%E5,E6 test to be a proper outcome to proceed

%outline the picture by boundary detection
se = strel('square',3);
gd = imdilate(E6, se);
ge = imerode(E6, se);
gg = gd - ge;

%complement the image to compare with the original picture
gg = imcomplement(gg);
level = graythresh(gg);
BW = imbinarize(gg, level);
BW_e = imerode(BW, se);
BW_o = imopen(BW, se);
BW_bd = BW_o - BW_e;
montage({A, gg, BW})
```

```
BW_bd2 = BW_o - BW_e;
```



```
%testing which SE size is the best fit for underlining
E5 = imerode(A,SE5);
E6 = imerode(A,SE6);
E8 = imerode(A,SE8);
E10 = imerode(A,SE10);
montage({A, E5, E6, E8, E10}, "size", [1,5])
%E5,E6 test to be a proper outcome to proceed

%outline the picture by boundary detection
E5 = imcomplement(E5);
level = graythresh(E5);
BW = imbinarize(E5, level);

SE = ones(4,4);
BW_e = imerode(BW, SE);
%illeminate the noise
BW_o = imopen(BW, SE);
BW_bd = BW_o - BW_e;

%complement the image to compare with the original picture
Final_result = imcomplement(BW_bd);
montage({A,BW_bd, Final_result})
```

```
函数或变量 'BW_d' 无法识别。
```