Xi'an Jiaotong-Liverpool University

西交利物浦大學

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE412 Image and Video Processing

Lab5 - Morphological Operations

Student Name Zhiyu.Xu

Student ID 1926905

December 2019

Contents

Task 1 Morphological operation

1.1 Extract the boundary

1.1.1 Result and code

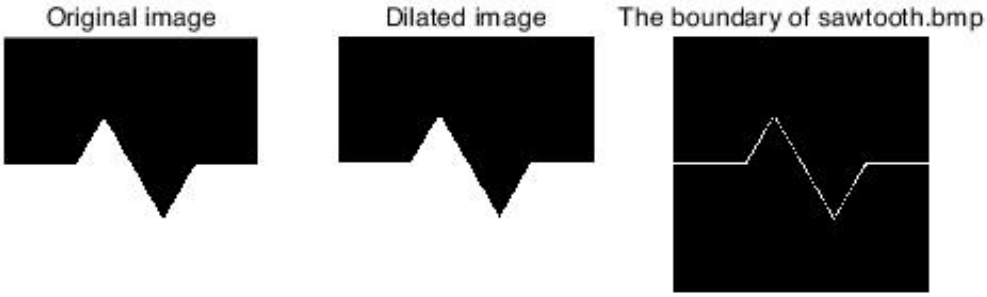

Figure 1 Result of extracting the boundary

```
% Task1 Morphological operation
%% (1)Extract the boundary
im_sawtooth = im2double(imread( 'sawtooth.bmp' ));
im_sawtooth = round(im_sawtooth);
SE_1 = strel('disk',2);
im_sawtooth_dilate = imdilate(im_sawtooth,SE_1);
im_sawtooth_bundary =
xor(im_sawtooth,im_sawtooth_dilate);
figure(1)
subplot(1,3,1);imshow(im_sawtooth);title('Original
image')
subplot(1,3,2);imshow(im_sawtooth_dilate);title('Dila
ted image')
subplot(1,3,3);imshow(im_sawtooth_bundary);title('The
boundary of sawtooth.bmp')
truesize
```

Code 1 Extract the boundary

1.1.2 Explanation

In the above code,first, format the original image. Change the gray value of 0~255 to 0~1, that is, change the format uin8 to format double. In order to make the subsequent image processing more intuitive, change the image gray value to an integer. 0 or 1. Then set a radius of a circular structuring element 2, and then the original image has been processed dilate, erode where not used because of corrosion original image would be reduced, the resulting boundary than The true boundary is small or panned down. Finally, using a logical exclusive OR operation, the boundary is determined. As shown in Figure 1.

1.2 The operations of erosion, dilation, opening, and closing

1.2.1 Result and code

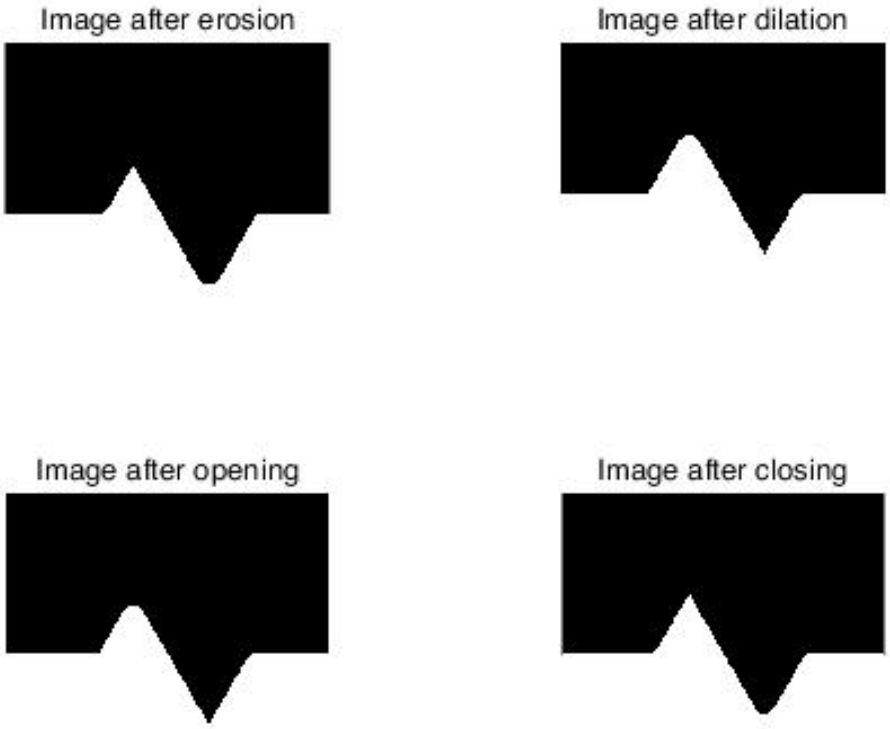

Figure 2 Result of four operations

operations	Erosion	Dilation	Opening	Closing
the number of foreground pixels	96521	116169	106117	106575

Table 1 The numbers of foreground pixels by four operations

```
%% (2)The operations of erosion, dilation, opening, and
closing
im_sawtooth = im2double(imread( 'sawtooth.bmp' ));
SE_2 = strel('disk',15);
im_sawtooth_e = imerode(im_sawtooth,SE_2); % Erosion
im_sawtooth_d = imdilate(im_sawtooth,SE_2); % Dilation
im_sawtooth_o = imopen(im_sawtooth,SE_2); % Opening
im_sawtooth_c = imclose(im_sawtooth,SE_2); % Closing
%Calculate pixels of foreground
Foreground_Numer_erision =
```

```
length(find(im_sawtooth_e==1));
Foreground_Number_dilation =
length(find(im_sawtooth_d==1));
Foreground_Number_opening =
length(find(im_sawtooth_o==1));
Foreground_Number_closing =
length(find(im_sawtooth_c==1));
%plot result
figure(2)
subplot(2,2,1);imshow(im_sawtooth_e);title('Image
after erosion');
subplot(2,2,2);imshow(im_sawtooth_d);title('Image
after dilation');
subplot(2,2,3);imshow(im_sawtooth_o);title('Image
after opening');
subplot(2,2,4);imshow(im_sawtooth_c);title('Image
after closing');
```

Code 2 Four operations

1.2.2 Explanation

First set the SE to a circle with a radius of 15. After Erosion operation, the foreground of the image will be reduced. After Dilation operation, the foreground of the image will be increased. The pixels are reduced to 96521, and the pixels are increased to 116169 after the Dilation operation. After the Erosion operation, the angle of the foreground image greater than 180 ° will become an arc, and after the while Dilation operation, the angle of the foreground image less than 180 ° will become an arc.

The opening operation erodes and then swells, and the closing operation erodes and swells, so the processed foreground pixels are basically the same as the original, and the deviation is not large. The image after the opening operation is similar to the expanded image, and the image after the closing operation is similar to the eroded image.

1.3 Repeat the opening operation

1.3.1 Result and code

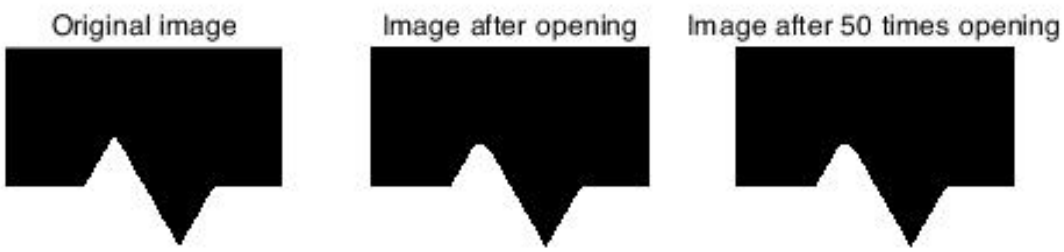

Figure 3 Repeat the opening operation

```
%% (3)Repeat the opening operation several times with the
same SE
im_sawtooth = im2double(imread( 'sawtooth.bmp' ));
im_sawtooth = round(im_sawtooth);
SE_2 = strel('disk',15);
im_sawtooth_o = imopen(im_sawtooth,SE_2);
im_sawtooth_o_repeat_50=im_sawtooth_o;
for i=1:49
im_sawtooth_o_repeat_50=imopen(im_sawtooth_o_repeat_50
,SE_2);%repeat 50 times
end
%same=1 means after repeating 50 tiomes opening
operation, the image
%doesn't change;
if im_sawtooth_o_repeat_50==im_sawtooth_o
    same=1;
else
    same=0;
end
figure(3)
subplot(1,3,1);imshow(im_sawtooth);title('Original
image');
subplot(1,3,2);imshow(im_sawtooth_o);title('Image
after opening');
subplot(1,3,3);imshow(im_sawtooth_o_repeat_50);title(
'Image after 50 times opening');
```

Code 3 Repeat the opening operation

1.3.2 Explanation

According to the judgment statement in the code, the obtained result is same = 1. We can find that when the SE is fixed, the image is always the same regardless of how many times the operation is repeated. This is because the opening operation does not change the image size, but only changes the shape of the foreground and background borders of the image according to the shape size of the SE.

Task 2 Car License Plate Recognition (1)

2.1 Binarization

2.1.1 Result and code

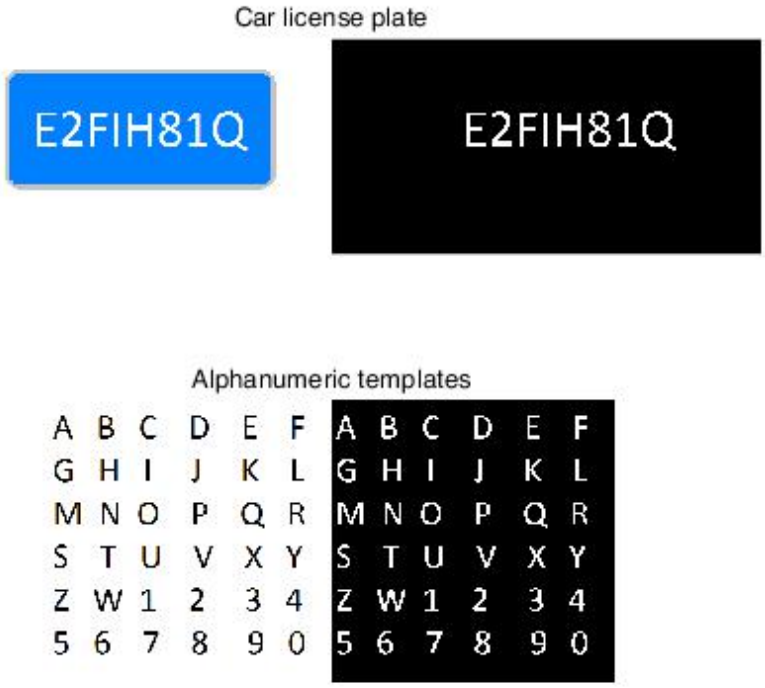

Figure 4 Result of binarization

```
%% binarize the license plate and the alphanumeric template images
im_CLP_original = imread('car_license_plate.bmp');
im_CLP = rgb2gray(im_CLP_original);
im_AT_original = imread('alphanumeric_templates .bmp');
im_AT_reversed = imcomplement(rgb2gray(im_AT_original));
im_CLP_binarized                                                    =
imbinarize(im_CLP,'adaptive','Sensitivity',0.2);
im_AT_binarized = imbinarize(im_AT_reversed,0.2);
figure(1)
subplot(2,1,1);imshowpair(im_CLP_original,im_CLP_binarized,'mo
ntage');title('Car license plate');
subplot(2,1,2);imshowpair(im_AT_original,im_AT_binarized,'monta
ge');title('Alphanumeric templates');
```

Code 4 Binarization

2.1.2 Explanation

The binarization process is shown in code 5. First, the license plate and

alphanumeric template images need to be processed from RGB to grayscale. Because the foreground of the alphanumeric template image (the alphanumeric characters) is black, and the background of the image is white, according to the requirements of the title, the color of the image needs to be reversed. Here, the imcomplement () function is used to achieve it.

Second, when binarizing the two images, the imbinarize () function is used. For the alphanumeric template image, the local adaptive image threshold function is used, so that only the character and number information in the license plate can be extracted as the foreground. The other content is the background, and finally the threshold is set to 0.2. The final result is shown in Figure 4.

2.2. Character detection

2.2.1 Result and code

Figure 5 Result of character detection

```
%% Character detection
im_CLP_binarized = imdilate(im_CLP_binarized,ones(1,3));
str =
detect_car_license_plate_v1(im_CLP_binarized,im_AT_binarized);
```

Code 5 Main code

```
function [str] = detect_car_license_plate_v1(im,im_SE)
%% Edges detection for car license plate
a = sum(im);
edge1 = find_edge1(a);
%% Edges detection for template
[r,c] = size(im_SE);
f = @(block_struct)sum(block_struct.data);
sumc(:) = blockproc(im_SE,[1 c],f);
sumr(:) = blockproc(im_SE,[r 1],f);
edgec = find_edge2(sumc);
edger = find_edge2(sumr);
edge2 = [edger;edgec];
%% Cut the templates
n = 1;
for i = 1:2:11
    for j = 1:2:11
        temp =
imcrop(im_SE,[edge2(1,j),edge2(2,i),edge2(1,j+1)-edge2(1,j),ed
ge2(2,i+1)-edge2(2,i)]);
        vp(n) = libpointer('voidPtr',temp);
        n = n + 1;
    end
end
```

```
%% detect symbols
symbols = ['A','B','C','D','E','F',...
           'G','H','I','J','K','L',...
           'M','N','O','P','Q','R',...
           'S','T','U','V','X','Y',...
           'Z','W','1','2','3','4',...
           '5','6','7','8','9','0'];
str(1) =' ';
for i = 1:36
    test = vp(i).Value;
    temp = imerode(im,test);
    s = sum(temp);
    if(sum(s)==1||sum(s)==2)
        for j = 1:1:512
           if (s(j) == 1)
               break;
           end
        end
        out = check(j,edge1);
        str(out) = symbols(i);
    end
end
str = string(str);
end
```

Code 6 Function of character detection v1

```
function array = find_edge1(input)
[~,c] = size(input);
temp = input(1);
array = zeros(1,16);
n = 1;
for i = 1:c
    if ((temp~=input(i))&&((temp==0)||(input(i)==0)))
        array(n) = i;
        n = n + 1;
    end
    temp = input(i);
end
end
```

Code 7 Function of find_edge1

```
function array = find_edge2(input)
[~,c] = size(input);
temp = input(1);
array = zeros(1,12);
n = 1;
for i = 1:c
    if ((temp~=input(i))&&((temp==0)||(input(i)==0)))
        array(n) = i;
```

```
        n = n + 1;
    end
    temp = input(i);
end
end
```

Code 8 Function of find_edge2

```
function out = check(in,list)
    [~,c] = size(list);
    for i = 1:1:c
       if (in <= list(i))
           break;
       end
    end
    out = i/2;
end
```

Code 9 Function of check

2.2.2 Explanation

Because the license plate characters have some mismatches with the character size of the character table, the image of the license plate character needs to be inflated first, so it will be more efficient to use the character table to identify the license plate.

The function of license plate character recognition is shown in code 5. The function contains three modules. The first module is to separate the alphanumeric templates according to each character. The principle is to find the index of the edge of each character through the vertical and horizontal directions, locate the position of each character by the index, and Image cut by imcrop() function.

The second module is character recognition. Here, the previously cut 36 characters are erode processed with the license plate image respectively. If the recognition is successful, there will be one or two foreground pixels. If the recognition fails, the image will be all after King (all black). However, in the recognition of the character "I", a plurality of discrete pixels will be recognized, as shown in Fig. 6, because there are multiple characters containing "I", and here a hit-miss needs to be used for processing.

Figure 6 Detected result of "I"

The third module is to sort and output the recognized characters, and detect the license plate edge by the previous algorithm. Then use the image of the recognized

character to compare the position with the character edge of the license plate to get the sorted character of the license plate.

Task 3 Car License Plate Recognition (2)

3.1 Hit and miss

3.1.1 Result and code

Figure 7 Result of hit-miss

```
% Task3 Car License Plate Recognition (2)
%% binarize the license plate and the alphanumeric template images
im_CLP_original = imread('car_license_plate.bmp');
im_CLP = rgb2gray(im_CLP_original);
im_AT_original = imread('alphanumeric_templates .bmp');
im_AT_reversed = imcomplement(rgb2gray(im_AT_original));
im_CLP_binarized =
imbinarize(im_CLP,'adaptive','Sensitivity',0.2);
im_AT_binarized = imbinarize(im_AT_reversed,0.2);
%% Character detection
im_CLP_binarized = imdilate(im_CLP_binarized,ones(1,3));
str =
detect_car_license_plate_v2(im_CLP_binarized,im_AT_binarized);
```

Code 10 Main code

```
function [str] = detect_car_license_plate_v2(im,im_SE)
%% Edges detection for car license plate
a = sum(im);
edge1 = find_edge1(a);
%% Edges detection for template
[r,c] = size(im_SE);
f = @(block_struct)sum(block_struct.data);
sumc(:) = blockproc(im_SE,[1 c],f);
sumr(:) = blockproc(im_SE,[r 1],f);
edgec = find_edge2(sumc);
edger = find_edge2(sumr);
edge2 = [edger;edgec];
%% Cut the templates
n = 1;
for i = 1:2:11
    for j = 1:2:11
        temp =
imcrop(im_SE,[edge2(1,j),edge2(2,i),edge2(1,j+1)-edge2(1,j),ed
ge2(2,i+1)-edge2(2,i)]);
```

```
        vp(n) = libpointer('voidPtr',temp);
        n = n + 1;
    end
end
%% detect symbols
symbols = ['A','B','C','D','E','F',...
           'G','H','I','J','K','L',...
           'M','N','O','P','Q','R',...
           'S','T','U','V','X','Y',...
           'Z','W','1','2','3','4',...
           '5','6','7','8','9','0'];
str(1) =' ';
for i = 1:36
    test = vp(i).Value;
    SE_1 = imerode(test,ones(1,3));
    SE_2 = imdilate(test,ones(3,5))-imdilate(test,ones(1,3));
    temp = bwhitmiss(im,SE_1,SE_2);
    if i==9
        temp(:,1)=0;
    end
    s = sum(temp);
    if(sum(s)==1)
        for j = 1:1:512
           if (s(j) == 1)
               break;
           end
        end
        out = check(j,edge1);
        str(out) = symbols(i);
    end
end
str = string(str);
end
```

Code 11 Function of character detection v2

3.1.1 Explanation

Code 9 is a modification of the core algorithm based on code 5 to change erode processing to hit-miss processing.

The core of the hit-miss algorithm is to detect both the inside of the target and the outside of the target. The intersection of the two is the result of the final detection. As shown in code 9, SE_1 is used to detect the inside of the target, and SE_2 is used to detect the outside of the target, which can be completed by bwhitmiss (). This syntax is equivalent to imerode (BW, SE_1) & imerode (~ BW, SE_2). Therefore, the problem of recognizing multiple characters "I" in Task2 was solved.

The final recognition result is shown in Figure 7.