EEE412-IMAGE PROCESSING-LAB 5

Author: Zhipeng Ye

Student ID: 1926908

Faculty: MSc Multimedia Telecommunications

Data: 6th December 2019

Morphological operation on the image of *im_sawtooth*. (**40 marks**)

Firstly, load the image *sawtooth.bmp* as *im_sawtooth*.

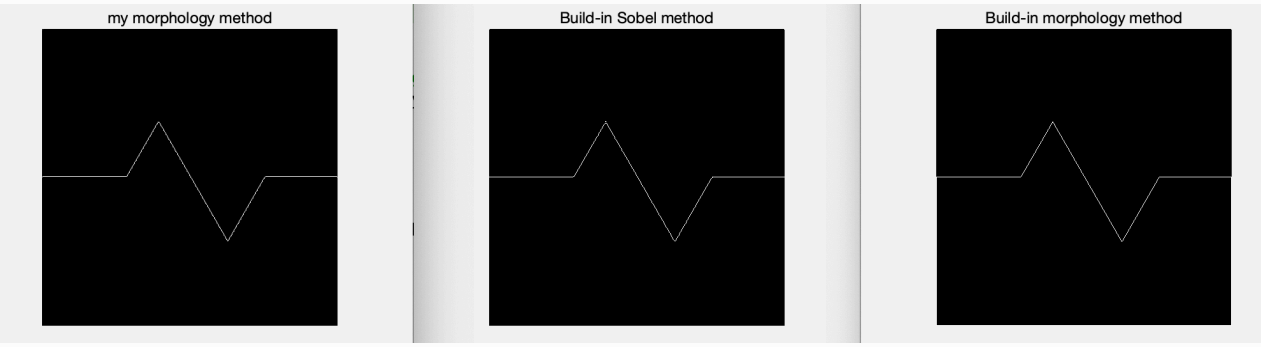
(1) Extract the boundary of the image, and show it in the report. (**10** marks)

Code

```
clear
clc

im_sawtooth = imread('sawtooth.bmp');

% dilate method return dilated image - image
im_sawtooth_boundary = get_boundary(im_sawtooth);
figure(1)
imshow(im_sawtooth_boundary);
title('my morphology method');

% build-in sobel operation
im_sawtooth_edge = edge(im_sawtooth,'sobel');
figure(2)
imshow(im_sawtooth_edge);
title('Build-in Sobel method');

% build-in function
im_sawtooth_perim = bwperim(im_sawtooth,4)
figure(3)
imshow(im_sawtooth_perim)
title('Build-in morphology method');
```

```
function boundary_edge = get_boundary(im)

disk_se = strel('disk',1);

im_dialtion = imdilate(im, disk_se);

% we can use subtraction to get the same result
boundary_edge = xor(im_dialtion,im);

end
```

Result



Analyze

I use three methods to solve this question. Firstly, I use the morphology method to get boundary. Secondly, I use soble operation to get boundary. Thirdly, I use build-in morphology funtion to get boundary. Consequently, all methods are ok for this question.

---

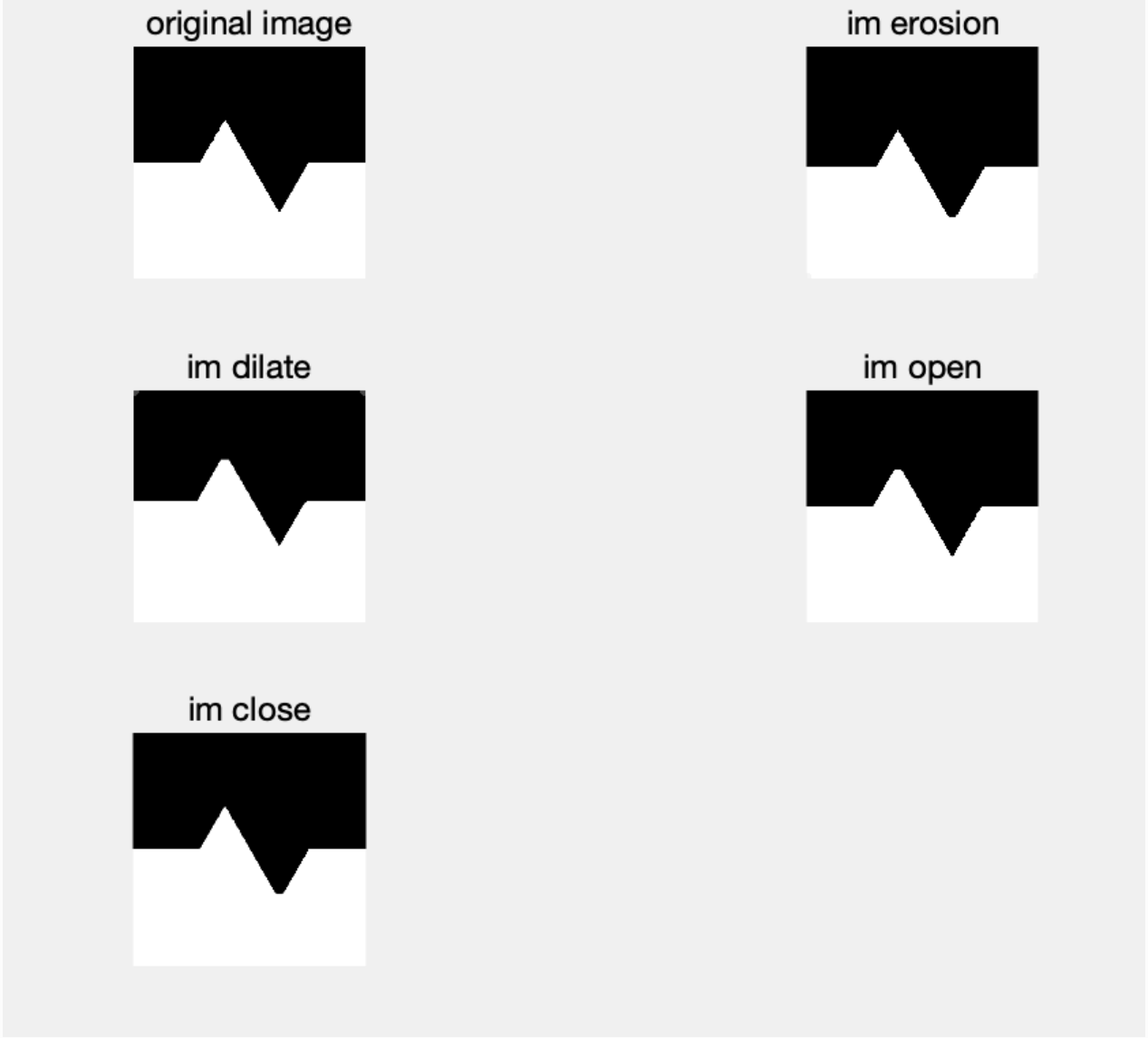
(2) Do the operations of erosion, dilation, opening, and closing. Please use the function of strel to create the structuring element with the shape of disk (You can set your preferred radius). Show the results after each operations and calculate the number of foreground pixels. Write your comments on comparing the results of erosion and opening. (**20** marks).

Code

```
clear
clc

im = imread('sawtooth.bmp');
subplot(3,2,1)
imshow(im)
title('original image')
disk_se = strel('disk',10);

im_erosion = imerode(im,disk_se);
subplot(3,2,2)
imshow(im_erosion)
```

```
title('im erosion')
im_erosion_pixels = sum(sum(im_erosion == 255));


im_dilate = imdilate(im,disk_se);
subplot(3,2,3)
imshow(im_dilate)
title('im dilate')
im_dilate_pixels = sum(sum(im_dilate == 255));


im_open = imopen(im,disk_se);
subplot(3,2,4)
imshow(im_open)
title('im open')
im_open_pixels = sum(sum(im_open == 255));


im_close = imclose(im,disk_se);
subplot(3,2,5)
imshow(im_close)
title('im close')
im_close_pixels = sum(sum(im_close == 255));
```

Result




```
im_erosion_pixels =

        99375

im_dilate_pixels =

       112737

im_open_pixels =

       106039

im_close_pixels =

       106239
```

Analyze

Comparing the results of erosion and opening,

The outward Angle of background become smoother and circular in erosion operation.

For open operation, the ourtward angle of foreground become smoother and circular.

The number of foreground pixels in erosion is smaller than that in the open picture, because of the erosion operation is a method to reduce the area of foreground.

---

(3) Repeat the opening operation several times with the same SE. What do you find? And Why? (**10** marks)

Code

```
clear
clc

im_sawtooth = imread('sawtooth.bmp');

disk_se = strel('disk',20);

im_open = imopen(im_sawtooth, disk_se);
im_open_pixels = sum(sum(im_open == 255));

for i = 1:10
    im_open = imopen(im_open, disk_se);
    im_open_pixels = sum(sum(im_open == 255));
    sprintf('The pixels of open is %d',im_open_pixels)
end
```

Result

```
ans =

    'The pixels of open is 105756'


ans =

    'The pixels of open is 105756'


ans =

    'The pixels of open is 105756'
```

The result doesn't change.

Analyze

I do opening operation serveral times, the result is that the pixels of foregroun doesn't change forever. This is because that opening operation is a idempotent operation. This property can be presented by this formula.

Car License Plate Recognition (1) (**30 marks**)

In this task you will learn how to recognize the alphanumeric characters on a license plate using morphological image processing.

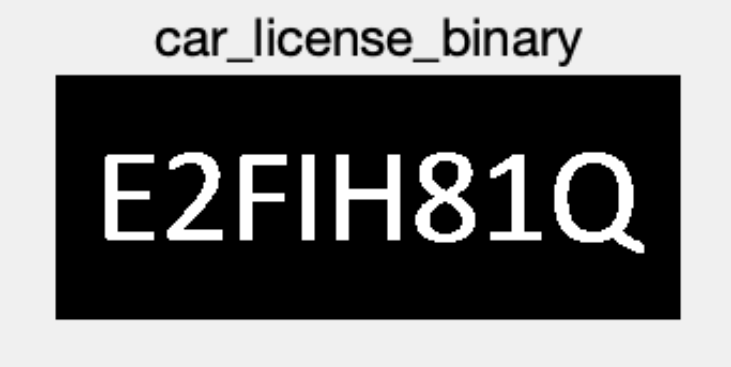
Firstly, you need to binarize the license plate (im) and the alphanumeric template images; the background of all these images should be black, whereas, the foreground representing the objects that need to be detected (i.e., in this case the alphanumeric characters) is white. While doing so use the same threshold for all the binarization operations. In you report submit the binarized plate image, and describe your binarization approach. **(10 marks)**

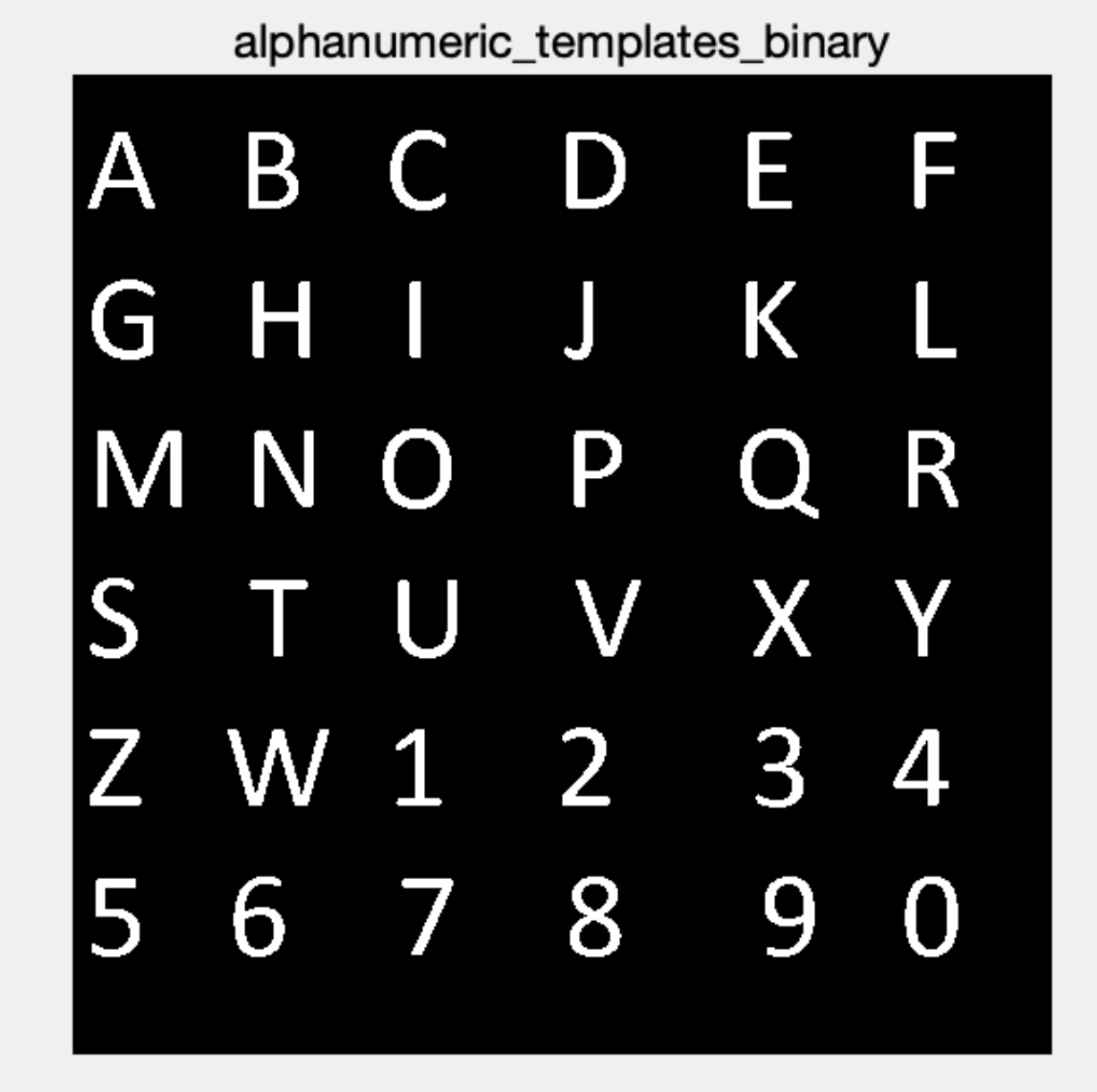
Code

```
clear
clc

car_license_plate = imread('car_license_plate.bmp');
alphanumeric_templates = imread('alphanumeric_templates .bmp');

% because I use the build-in function to Binarizate image
% however the threshold of im2bw must at 0-1
% therefore I choose 0.5 as a threshold
my_threshold = 0.5;

car_license_binary = im2bw(car_license_plate,my_threshold);

% segment car license binary
index = find(sum(~car_license_binary,1) == max(sum(~car_license_binary,1)))

left = min(index)

right = max(index)

index = find(sum(~car_license_binary,2) == max(sum(~car_license_binary,2)))

top = min(index)

bottom = max(index)

car_license_binary = car_license_binary([top:bottom],[left:right])

alphanumeric_templates_binary = im2bw(alphanumeric_templates, my_threshold);

alphanumeric_templates_binary = ~alphanumeric_templates_binary;
figure(1)
imshow(car_license_binary)
title('car\_license\_binary')
```

```
figure(2)
imshow(alphanumeric_templates_binary)
title('alphanumeric\_templates\_binary')
```

Result





Analyze

For this task, I use build-in function of im2bw to get binary image, Furthermore, in order to locate car license plate, I segment the picture to get a perfect result. For alphabet, I use negation operation to make sure the letter is white.

Perform character detection by using the erosion operation on the binarized version of the image. There might be some small mismatches along the character boundaries between the license plate and the template images. To eliminate this mismatch, you might use a smaller version of the characters in the templates, which could be achieved by eroding the templates before using them in the detector.

Write a Matlab function which detects the car license plate and outputs it as a string. The function should have the following declaration:

```
function [str] = detect_car_license_plate_v1(im, ..
```

Include in your report the detected car license plate, and comment on your finding, if there were any anomalies then comment on them and explain why they happened. **(20 marks)**

Code

Main

```
clear
clc

car_license_plate = imread('car_license_plate.bmp');

alphanumeric_templates = imread('alphanumeric_templates .bmp');

car_character_v1 = detect_car_license_plate_v1(car_license_plate,
alphanumeric_templates);
```

Detect

```
function  car_character =
detect_car_license_plate_v1(car_license_plate,alphanumeric_templates)

my_threshold = 0.5;

car_license_binary = im2bw(car_license_plate,my_threshold);

% segment car license
character_im = segment_car_license(car_license_binary);

alphanumeric_templates_binary = im2bw(alphanumeric_templates, my_threshold);
```

```
% hashmap key is string, value is image
map_container = im_map_character(alphanumeric_templates_binary);

alphabet = map_container.keys;

length_alphabet = length(alphabet);
length_character = length(character_im);

car_character = [];

for i = 1:length_character
    for j = 1:length_alphabet

        E_strel = map_container(alphabet{j});

        %         erode
        im_test = imerode(character_im{i},E_strel);
        is_mattched = sum(sum(im_test)) == 1;

        if is_mattched
            break;
        end
    end

    if is_mattched
        car_character = [car_character alphabet{j}];
    end
end

end
```

Segment

```
function character_im = segment_car_license(car_license_binary)

[m,n] = size(car_license_binary);

index = find(sum(~car_license_binary,1) == max(sum(~car_license_binary,1)))

left = min(index)

right = max(index)

index = find(sum(~car_license_binary,2) == max(sum(~car_license_binary,2)))

top = min(index)

bottom = max(index)

car_license_binary = car_license_binary([top:bottom],[left:right])

[m,n] = size(car_license_binary);
```

```
left = min(find(sum(car_license_binary,2) >= 1)) -1;
right = max(find(sum(car_license_binary,2) >= 1)) +1;
car_license_binary = car_license_binary([left: right],:);
column_position_list = [];

start_signal = 1

for i = 1:n
    %     detect point of incision
    is_segment_point_left = sum(car_license_binary(:,i)) >= 1;
    is_segment_point_right = sum(car_license_binary(:,i)) < 1;
    if is_segment_point_left & start_signal
        column_position_list = [column_position_list i];
        start_signal = ~start_signal;
    elseif is_segment_point_right & ~start_signal
        column_position_list = [column_position_list i];
        start_signal = ~start_signal;
    end
end

length_position = length(column_position_list);

character_im = {};

for i = 1:2:length_position
    left = column_position_list(i)-3;
    right = column_position_list(i+1)+3;
    character_im = [character_im ;car_license_binary(:,(left:right))];
end

end
```

Segment and match

```
function map_container = im_map_character(alphanumeric_templates_binary)

[m,n] =size(alphanumeric_templates_binary)

alphanumeric_templates_binary = ~alphanumeric_templates_binary;

row_position_list = []

start_signal = 1

for i = 1:m
    %     detect point of incision
    is_segment_point_top = sum(alphanumeric_templates_binary(i,:)) >= 1;
    is_segment_point_bottom = sum(alphanumeric_templates_binary(i,:)) < 1;
    if is_segment_point_top & start_signal
        row_position_list = [row_position_list i-1];
        start_signal = ~start_signal;
    elseif is_segment_point_bottom & ~start_signal
        row_position_list = [row_position_list i+1];
        start_signal = ~start_signal;
```

```
    end

end


column_position_list = [];

start_signal = 1

for i = 1:n
    %     detect point of incision
    is_segment_point_left = sum(alphanumeric_templates_binary(:,i)) >= 1;
    is_segment_point_right = sum(alphanumeric_templates_binary(:,i)) < 1;
    if is_segment_point_left & start_signal
        column_position_list = [column_position_list i-1];
        start_signal = ~start_signal;
    elseif is_segment_point_right & ~start_signal
        column_position_list = [column_position_list i+1];
        start_signal = ~start_signal;
    end
end

length_position = length(row_position_list);

character_im = {};

for i = 1:2:length_position

    for j = 1:2:length_position
        top = row_position_list(i);
        bottom = row_position_list(i+1);
        left = column_position_list(j);
        right = column_position_list(j+1);
        character_im = [character_im
;alphanumeric_templates_binary((top:bottom),(left:right))];
    end
end

character_dictionary = {'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'}

% map character with image
map_container = containers.Map(character_dictionary,character_im)

end
```

Result

```
car_character_v1 =

    'E2FIH81C'
```

Analyze

There is some anomalies happening in detection. The expected result is 'E2FIH81Q', however the experiment result is 'E2FIH81C'. This is because, 'C' and 'Q' have similar structure, when using 'C' to erode 'Q' , there will be also a white point. Therefore, when we just use erode to recognize car license plate, it will cause anomalies to some extent. Maybe we can use hitandmiss to avoid this in next question.

Furthermore, I think Morphological isn't a good method to recognize object, Because it isn't robust enough for different situation. if the object has different magnitude, different angle, different shape... this method will fail to recognize.

For this question, I think we can use distance to recognize object.

Car License Plate Recognition (2) (**30 marks**)

Repeat the previous task using a hit-miss filter instead of the erosion operation. Write a

Matlab function for this task which has the following declaration:

```
function [str] = detect_car_license_plate_v2(im, ...
```

(10 marks)

For the hit-miss filter and for the foreground Structure Element use the eroded template as in task 2. For the background Structure Element use the outline around the character. Comment on the advantages of this approach and compare it with the one used in task 2. **(20 marks)**

Code

```
function  car_character =
detect_car_license_plate_v2(car_license_plate,alphanumeric_templates)

my_threshold = 0.5;

car_license_binary = im2bw(car_license_plate,my_threshold);

% segment car license
character_im = segment_car_license(car_license_binary);

alphanumeric_templates_binary = im2bw(alphanumeric_templates, my_threshold);
```

```
map_container = im_map_character(alphanumeric_templates_binary);

alphabet = map_container.keys;

length_alphabet = length(alphabet);
length_character = length(character_im);

car_character = [];

for i = 1:length_character

    for j = 1:length_alphabet

        E_strel_1 = map_container(alphabet{j});

        disk_se = strel('disk',1);

        E_strel_1_erosion = imdilate(E_strel_1,disk_se);

        E_strel_2 = get_boundary(E_strel_1_erosion);

        %         hit and miss
        im_test = bwhitmiss(character_im{i},E_strel_1,E_strel_2);

        is_mattched = sum(sum(im_test)) ==1;

        if is_mattched
            break
        end
    end

    if is_mattched
        car_character = [car_character alphabet{j}];
    end
end

end
```

Result

```
car_character_v2 =

    'E2FIH81Q'
```

Analyze

As the result is expected, the car license plate is totally recognized successfully.

The advantage of this method , compaing with erosion simplify, is that it can recognize the outline of character. Because this method use and operation to get intersection to avoid some anomalies which happen in task2.

Here is a mathematical proof.

$$Y = (A \ominus H) \cap (A^c \ominus M)$$