The Report of MP2

I. Basic Concepts

1.1 Morphological Operators

Binary images may contain numerous imperfections. In particular, the binary regions produced by simple thresholding are distorted by noise and texture. Morphological operators pursue the goals of removing these imperfections by accounting for the form and structure of the image. These techniques can be extended to greyscale images.

Morphological operators often take a binary image and a structuring element as input and combine them using a set operator (intersection, union, inclusion, complement). They process objects in the input image based on characteristics of its shape, which are encoded in the structuring element.

Usually, the structuring element is sized 3×3 and has its origin at the center pixel. It is shifted over the image and at each pixel of the image its elements are compared with the set of the underlying pixels. If the two sets of elements match the condition defined by the set operator (e.g. if the set of pixels in the structuring element is a subset of the underlying image pixels), the pixel underneath the origin of the structuring element is set to a pre-defined value (0 or 1 for binary images). A morphological operator is therefore defined by its structuring element and the applied set operator.

For the basic morphological operators, the structuring element contains only foreground pixels (i.e. ones). These operators, which are all a combination of erosion and dilation, are often used to select or suppress features of a certain shape, e.g. removing noise from images or selecting objects with a particular direction.

Morphological operators can also be applied to gray-level images, e.g. to reduce noise or to brighten the image. However, for many applications, other methods like a more general spatial filter produces better results.

1.2 Algorithm and Implementation

In this homework, it is required to implement five functions: erosion, dilation, opening, closing and boundary. According to the Machine Vision, erosion and dilation are fundamental operations while opening, closing and even boundary are all compound operations based on fundamental operations.

Erosion of a binary image f by a SE s (denoted $f \ominus s$) produces a new binary image $g = f \ominus s$ with ones in all locations (r,c) of a structuring element's origin at which that structuring element s fits the input image f, i.e. g(r,c) = 1 is s fits f and 0 otherwise, repeating for all pixel coordinates (r,c). Erosion with small (r,c) square SE shrinks an image by stripping away a layer of pixels from both the inner and outer boundaries of regions. The holes and gaps between different regions become larger, and small details are eliminated. Erosion removes small-scale details from a binary image but simultaneously reduces the size of regions of interest, too.

Dilation of an image f by a SE s (denoted $f \oplus s$) produces a new binary image $g = f \oplus s$ with ones in all locations (r,c) of a structuring element's orogin at which that structuring element s hits the the input image f, i.e. g(r,c) = 1 if s hits f and 0 otherwise, repeating for all pixel coordinates (r,c).

Dilation has the opposite effect to erosion -- it adds a layer of pixels to both the inner and outer boundaries of regions. The holes enclosed by a single region and gaps between different regions become smaller, and small intrusions into boundaries of a region are filled in.
As referred above, many morphological operators are represented as combinations of erosion, dilation, and simple set-theoretic operations such as the complement of a binary image.
Opening in MP2 is implemented by an erosion followed by a dilation. That is,

$$f \circ s = (f \ominus s) \oplus s$$

And Closing in MP2 is implemented by a dilation followed by an erosion. That is,

$$f \cdot s = (f \oplus s) \ominus s$$

And Boundary is implemented based on erosion. That is,

$$b = f - (f \ominus s)$$

where b is the boundary of input image, f is the input image and s is SE(in this function it is defined as a 3*3 square matrix).
According to the above algorithm, it is simple to program the function. The key points of these functions are erosion and dilation, as others are implemented upon such two functions. When implementing these two, it is vital to establish a structure element (SE) and set the original point in the SE. In this homework, I set the center of SE as the original point, and normalize other points in the SE. Next scan the input image by the SE, and relabel the image via the instructions of different operators.

1.3 The purpose of the functions

There are 5 functions to be implemented in this assignment. The first two are Erosion and Dilation, which aim to change the shape of image based on established SE. The other three are Opening, Closing and Boundary, which are compound morphological operators and all are implemented based on the Erosion and Dilation. These functions are shown as followed.

```
function img_out1 = Erosion (img_in , SE);
function img_out2 = Dilation (img_in , SE);
function img_out3 = Opening (img_in , SE);
function img_out4 = Closing (img_in , SE);
function img_out5 = Boundary (img_in );
```

where img_out is the output image after morphological operator, img_in is the input image and SE is input Structure Element.

II. Results and Analysis

2.1 Test Results

After running the codes and test the five different functions via two images, there come the results as followed.

1) Erosion

The testing results of the function of Erosion are shown as Figure 1, which are the results of ‘gun.bmp’ and ‘palm.bmp’. For the ‘gun.bmp’, SE is set as 3*3 square matrix. And for the ‘plam.bmp’, SE is set as 4*2 matrix.

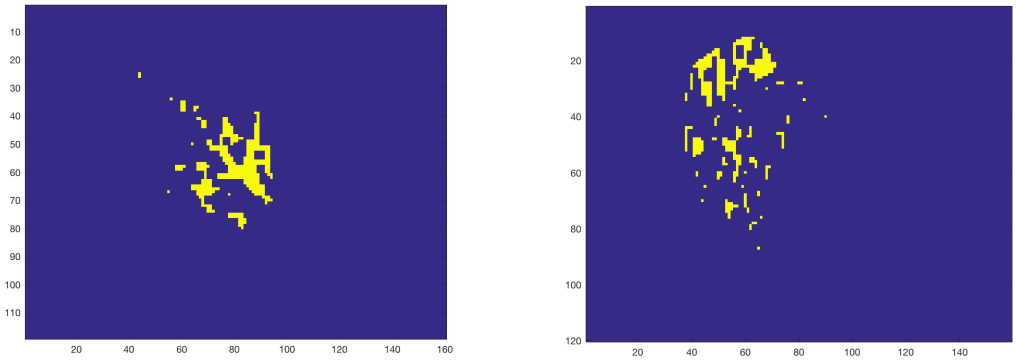

Figure 1 Testing Results of Erosion (Gun & Palm)

2) Dilation

The testing results of the function of Dilation are shown as Figure 2, which are the results of 'gun.bmp' and 'palm.bmp'. For the 'gun.bmp', SE is set as 3*3 square matrix. And for the 'plam.bmp', SE is set as 4*2 matrix.

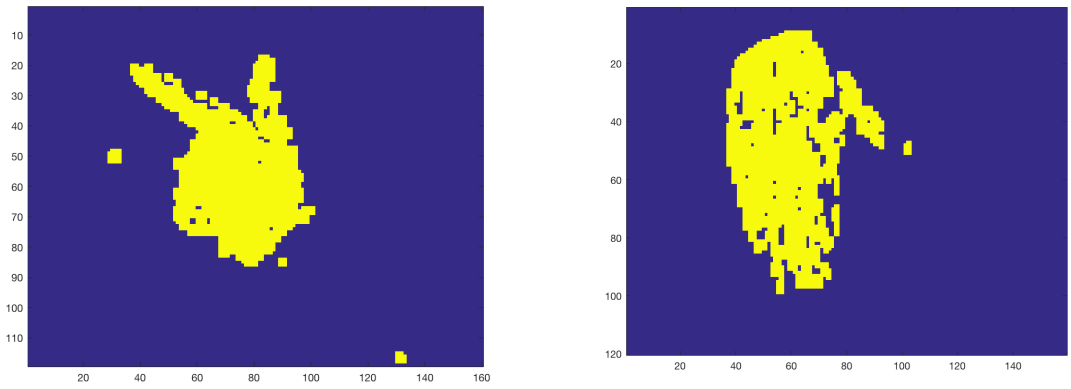

Figure 2 Testing Results of Dilation (Gun & Palm)

3) Opening

The testing results of the function of Opening are shown as Figure 3, which are the results of 'gun.bmp' and 'palm.bmp'. For the 'gun.bmp', SE is set as 3*3 square matrix. And for the 'plam.bmp', SE is set as 4*2 matrix.

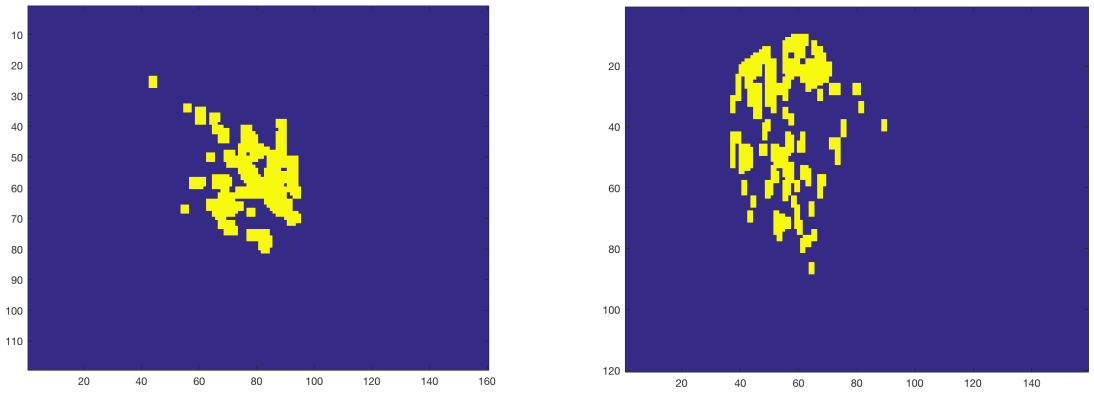

Figure 3 Testing Results of Opening (Gun & Palm)

4) Closing

The testing results of the function of Closing are shown as Figure 4, which are the results of 'gun.bmp' and 'palm.bmp'. For the 'gun.bmp', SE is set as 3*3 square matrix. And for the 'plam.bmp', SE is set as 4*2 matrix.

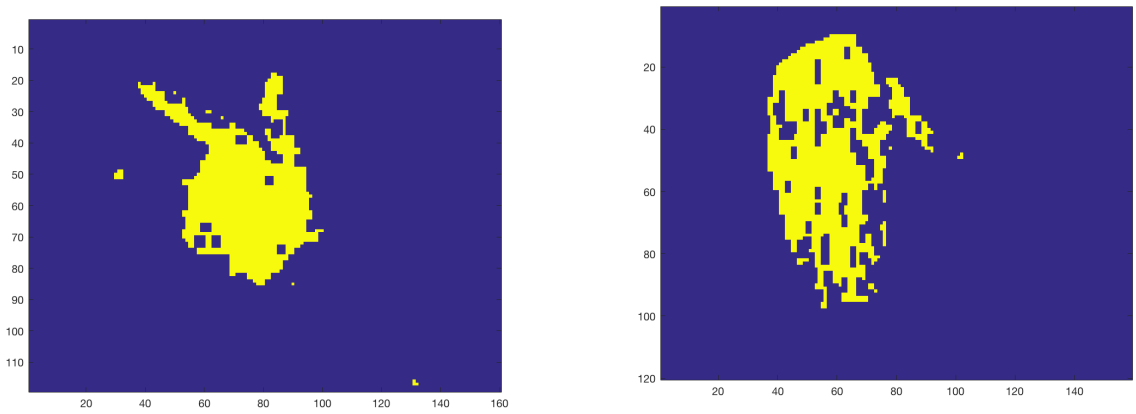

Figure 4 Testing Results of Closing (Gun & Palm)

5) Boundary

The testing results of the function of Boundary are shown as Figure 5, which are the results of 'gun.bmp' and 'palm.bmp'. For the 'gun.bmp', SE is set as 3*3 square matrix. And for the 'plam.bmp', SE is set as 4*2 matrix.

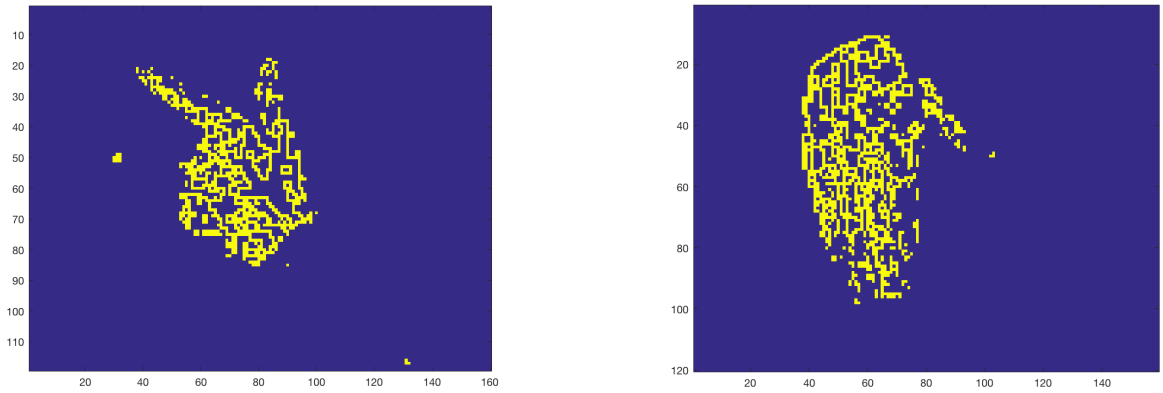

Figure 5 Testing Results of Boundary (Gun & Palm)

2.2 Result Analysis

From the above results of the different testing, it could be safely drawn a conclusion that all functions are implemented successfully. However, as for the last one named boundary, it should output the outmost boundary while the results turn out to be a little wield. In my opinion, I deem that it might output the boundaries of every small part of the image.

From the results, it could also be figured out that the effective of the morphological operator depends on the shape of established structure element(SE).

2.3 Summary

From this MP2, I get a better understanding of the core of morphological operators, especially the fundamental operators Erosion and Dilation. Through the testing of different operators, it is clearly to show the function of each operator.

III. Matlab Codes

```
function [l] = MP2(imagename,r,c)
%set up main function MP2(), where you can use any functions you want. It
%is used for testing each function. For the one you want to test, just
%activate the relative function and change the output of the main function.
    SE = ones(r,c);
    d = RdIm(imagename);
    %e = Erosion(d, SE);
    %f = Dilation(d, SE);
    %g = Opening(d , SE);
    %h = Closing(d , SE);
    l = Boundary(d);
end

function [modelX, modelY] = MStructureElement(SE)
%For the structure element(SE) you establish, set the original point at the
%center of SE, and collect extra pixels' indexs in SE based on the center
%point's coordinate.
    [Rs , Cs] = size(SE);
    centerr = int32(Rs/2);
    centerc = int32(Cs/2);
    X = zeros();
    Y = zeros();
    for i = 1 : Rs
        for j = 1 : Cs
```

```
            if SE(i, j) == 1
                X = [X, (i - centerr)];
                Y = [Y, (j - centerc)];
            end
        end
    end
    %we collect the x and y seperately
    modelX = X;
    modelY = Y;
end

function img_in = RdIm(imagename)
%set up a function for inputing the image you want
    img_in = imread(imagename,'bmp');
end

function img_out1 = Erosion(img_in , SE)
%establish the function of Erosion, where img_in and SE are the input while
%img_out1 is the output and it represents for the eroded image.
[r,c] = size(img_in);
finarr = zeros(r,c);
[modelX , modelY] = MStructureElement(SE);

%scan the whole img_in
for i = 1 : r
    for j = 1 : c
        pixel = img_in(i,j);
        if pixel ~= 0
        %for those pixels are not zero, define flagZero to judge the value
        %of neighbors of these pixels in the SE.
            flagZero = false;
            for k = 1 : length(modelX)
                x = (i + modelX(k));
                y = (j + modelY(k));
                if  x > 0  && x <= r  &&  y > 0 && y <= c
                %judge the boundary of the image, if there exists one zero
                %in the SE, than label such pixel as zero.
                    if img_in(x , y) == 0
                        finarr(i,j) = 0;
                        flagZero = true;
                        break;
                    end
                end
            end
            %If all neighbors are not zero, then label such pixel as one.
            if flagZero == false
                finarr(i,j) = 1;
            end
        end
    end
end
% From the upper operation, than we could get the eroded image.
img_out1 = finarr;
%If we want to test such function then output the following result.
%img_out1 = imagesc(finarr);
end

function img_out2 = Dilation(img_in , SE)
%establish the function of Dilation, where img_in and SE are the input while
%img_out2 is the output and it represents for the dilated image.
[r,c] = size(img_in);
finarr = zeros(r,c);
[modelX , modelY] = MStructureElement(SE);

%scan the whole img_in
for i = 1 : r
    for j = 1 : c
        pixel = img_in(i,j);
        if pixel == 0
         %for those pixels are zero, define flagZero to judge the value
        %of neighbors of these pixels in the SE.
            flagnotZero = false;
            for k = 1 : length(modelX)
                x = (i + modelX(k));
                y = (j + modelY(k));
                if  x > 0  && x <= r  &&  y > 0 && y <= c
                %judge the boundary of the image, if there exists a one
                %in the SE, than label such pixel as one.
```

```
                    if img_in(x , y) == 1
                        finarr(i,j) = 1;
                        flagnotZero = true;
                        break;
                    end
                end
            end
             %If all neighbors are zero, then label such pixel as zero.
            if flagnotZero == false
                finarr(i,j) = 0;
            end
        else
        %for those pixels equal to one, label them as one.
            finarr(i,j) = 1;
        end
    end
end
% From the upper operation, than we could get the dilated image.
img_out2 = finarr;
%If we want to test such function then output the following result.
%img_out2 = imagesc(finarr);
end

function img_out3 = Opening(img_in , SE)
%establish the function of Opening, where img_in and SE are the input while
%img_out3 is the output and it represents for the opening image. That is,
%we firstly erode the img_in, and then we dilate the eroded image, then we
%get the final opening image.
erodedimage = Erosion(img_in , SE);
dilatedimage = Dilation(erodedimage , SE);
img_out3 = imagesc(dilatedimage);
end

function img_out4 = Closing(img_in , SE)
%establish the function of Closing, where img_in and SE are the input while
%img_out4 is the output and it represents for the closing image. That is,
%we firstly dilate the img_in, and then we erode the dilated image, then we
%get the final closing image.
dilatedimage = Dilation(img_in , SE);
erodedimage = Erosion(dilatedimage , SE);
img_out4 = imagesc(erodedimage);
end

function img_out5 = Boundary(img_in)
%establish the function of Boundary, where img_in is the input and img_out5
%is the output and it represents for the boundary image.
%Firstly, we estailished a 3*3 square SE.
%Secondly, erode the img_in.
%Thirdly, scan the eroded image, judge the value of pixels in eroded
%image.If it is equal to 1, then relabel the pixel in the img_in as zero.
%Finally, the relabeled img_in is so-called boundary image.
[r , c] = size(img_in);
SE = ones(3,3);
erodedimage = Erosion(img_in , SE);
for i = 1 : r
    for j = 1 : c
        if erodedimage(i,j) == 1
            img_in(i,j) = 0;
        end
    end
end
img_out5 = imagesc(img_in);
end
```