EE4704 Image Processing and Analysis

Semester 1, AY2021/22

CA2Report

A. Feature Measurement

To calculate the threshold T and use it to threshold I, the T should be initialized as average intensity of I, which is done by mean2. the Code of Intermeans is shown as Figure1 below.

```
% To calculate the intermeans threshold;
% input is the gray level image 'test1.bmp'
% output is the threshold value T and the binary thresholded
% image Iout.
function [T,Iout] = intermeans(Iin)
T = mean2(Iin);%mean2 need Image Processing Toolbox for average intensity
T_new = 0;
while T ~= T_new
    T_new = T;
    R1 = mean2(Iin(Iin <= T));
    R2 = mean2(Iin(Iin > T));
    T = round((R1 + R2)/2);
end
Iout = Iin > T;%binary matrix
end
```

Fig1.Code of Intermeans

By run_A, the threshold image for test1 is shown as figure2 below with T = 113:

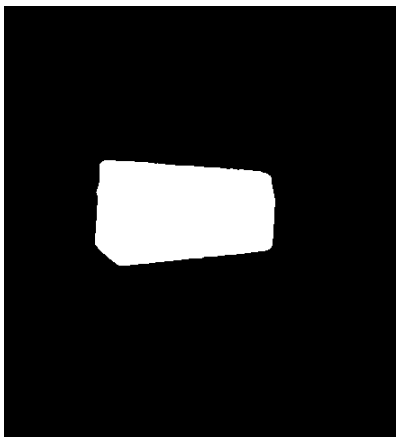

Fig2.Threshold Image of Test1

However, by run_B, a part of the missing part of the lower right corner of the picture was found. In order to remove the influence of non-central objects, a filter was added:

Iout = bwareafilt(Iout,1); here only 1 main object is kept.

In feature.m, six features are generated: perimeter, area, compactness, centroid, invariant moment $\phi 1$.

- Perimeter

To calculate perimeter, the boundary of figure should be got first. The bwperim function can keep the boundary of Iin in 8 directions. And the regionprops function can get the perimeter of boundary. The code of perimeter is shown as figure 3 below:

```
%perimeter
Iin_Boundary = bwperim(Iin, 8);
P = regionprops(Iin_Boundary,'Perimeter').Perimeter;
```

Fig3.Code of Perimeter

- Area

To calculate area, the pixel number within boundary should be calculated. To filter the outside noise, imfill function is used to fill the inside part of boundary. The code of area is shown as below:

```
%area
I_fill = imfill(Iin_Boundary, 'holes');%fill with perimeter above
A = sum(sum(I_fill));
```

Fig.4 Code of Area

- Compactness

Compactness = Perimeter^2 / (4*Pi*Area), which can be directly derived from perimeter and area.

- Centroid

To calculate centroid, the m00, m10, and m01 should be calculated first.

$m00 = sum_x (sum_y (f(x,y)))$

$m10 = sum_x (sum_y (x*f(x,y)))$

$m01 = sum_x (sum_y (y*f(x,y)))$.

Beside calculate for moment values, the coordinate should be change from matrix coordinate (X_{matrix}, Y_{matrix}) to left-bottom-origin coordinate (x, y).

X_{matrix} = length of picture – y

Y_{matrix} = x + 1

The code of centroid is shown as figure 5 below:

```
%centroid
m00 = 0;
m10 = 0;
m01 = 0;
for x = 0:size(I_fill,2)-1
    for y = 0:size(I_fill,1)-1
        for i = 1:3
            if i == 1
                m00 = m00 + 1 * I_fill(size(I_fill,1)-y, x+1);
            end
            if i == 2
                m10 = m10 + x * I_fill(size(I_fill,1)-y, x+1);
            end
            if i == 3
                m01 = m01 + y * I_fill(size(I_fill,1)-y, x+1);
            end
        end
    end
end
xbar = m10 / m00;
ybar = m01 / m00;
```

Fig5.Code of Centroid

- invariant moment $\phi 1$

Like centroid above, to calculate invariant moment, m00, m20, m02, m10, m01 is needed for u00, u20 and u02.

The code of invariant moment is shown as figure 6 below:

```
%invariant moment ϕ1
%  ϕ1 = u20/(u00)^2 + u02/(u00)^2
u00 = m00;
m20 = 0;
m02 = 0;
for x = 0:size(I_fill,2)-1
    for y = 0:size(I_fill,1)-1
        for i = 1:2
            if i == 1
                m20 = m20 + x^2 * I_fill(size(I_fill,1)-y, x+1);
            end
            if i == 2
                m02 = m02 + y^2 * I_fill(size(I_fill,1)-y, x+1);
            end
        end
    end
end
u20 = m20 - xbar*m10;
u02 = m02 - ybar*m01;
phione = u20/(u00)^2 + u02/(u00)^2;
end
```

Fig6.Code of Invariant Moment

By run_A, the features of test1 are shown as table1 below:

Perimeter	567.0810	**x_bar**	199.3693
Area	20016	**y_bar**	250.8056
Compactness	1.2785	**phione**	0.1983

Table1. Features for Test1

B. Feature Invariance

The histogram of test2 is shown as figure 7 below:

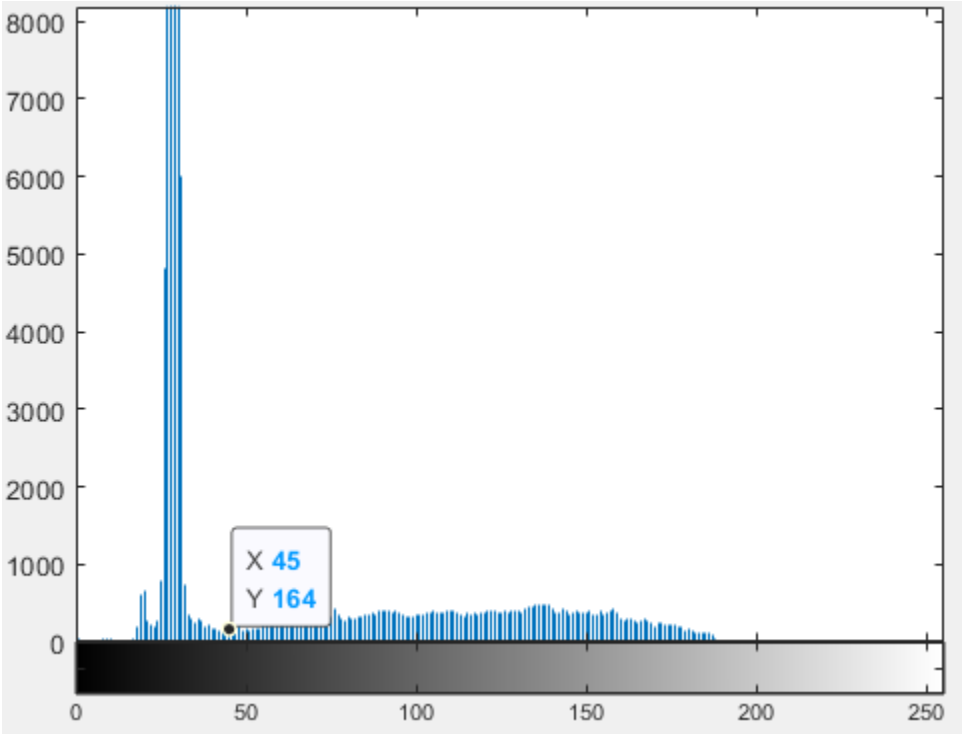

Fig7.Histogram of Test2

Shown by figure 7, the highest histogram should be dark park of central object. To get the object entirely, the grey level after 45 is seem as background. Therefore, Topt = 45 by histogram.

After threshold by generated J = 80 and Topt = 45, the test2 is shown as figure8 and figure9 below:

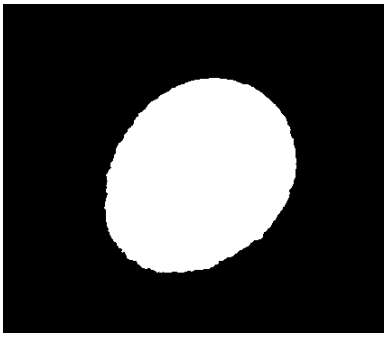

Fig8.Threshold Test2 with J

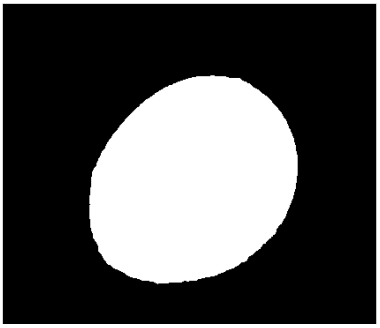

Fig9.Threshold Test2 with Topt

The figure generated by Topt is larger than J since threshold is smaller. And the boundary of Topt is smoother.

The feature of J and Topt figures is shown as table2 below:

	Perimeter	**Area**	**Compactness**	**x_bar**	**y_bar**	**phione**
J = 80	697.9230	35108	1.1041	219.1020	188.5594	0.1630
Topt = 45	750.2240	43141	1.0382	215.1180	179.0665	0.1626

Table2.Features for J and Topt

For perimeter and area, when threshold decrease the perimeter and area will increase, since more pixels will be included in.

For compactness, since Topt is smoother, the compactness of Topt is smaller.

The centroid is slightly shifting but can be ignored as the same.

The phione stay the same since shape of object is the same.

C. Boundary Plot

To calculate the $r - \theta$ values for test3, average of x and average of y should be calculated first.

x_bar = (1/N) *sumi(xi); y_bar = (1/N)*sumi(yi)

However, here also use the same coordinate transform mentioned before.

The code of average x and y is shown as figure 10 below:

```
function [r, theta] = rtheta(Iin)
Iin = Iin > 0;
% calculate average for x and y
xbar = 0;
ybar = 0;
for x = 0:size(Iin,2)-1
    for y = 0:size(Iin,1)-1
        xbar = xbar + x * Iin(size(Iin,1)-y, x+1);
        ybar = ybar + y * Iin(size(Iin,1)-y, x+1);
    end
end
xbar = xbar / sum(sum(Iin));
ybar = ybar / sum(sum(Iin));
```

Fig10.Code of Average of xy for Rtheta

After getting the center coordinate of x and y, r = sqrt ((x – x_bar)^2 + (y – y_bar)^2) and theta = arctan(y – y_bar, x – x_bar). Since the function atan2 will generate the negative value of angle, the negative angle will plus 2*Pi to range in (0, 2*Pi). The code of $r - \theta$ values is shown as below in figure 11:

```
r = zeros(sum(sum(Iin)),1);
theta = zeros(sum(sum(Iin)),1);
i = 0;
for x = 0:size(Iin,2)-1
    for y = 0:size(Iin,1)-1
         if Iin(size(Iin,1)-y, x+1) == 1
            i = i + 1;
            r(i) = sqrt((x - xbar)^2 + (y - ybar)^2);
            theta(i) = atan2(y - ybar, x - xbar);
         end
    end
end
for i = 1:size(theta,1)
    if theta(i)<0
        theta(i) = theta(i) + 2*pi;
    end
end
end
```

Fig11.Code of $r-\theta$ values for Rtheta

The $r-\theta$ plot for test3 is shown as figure 12 below:

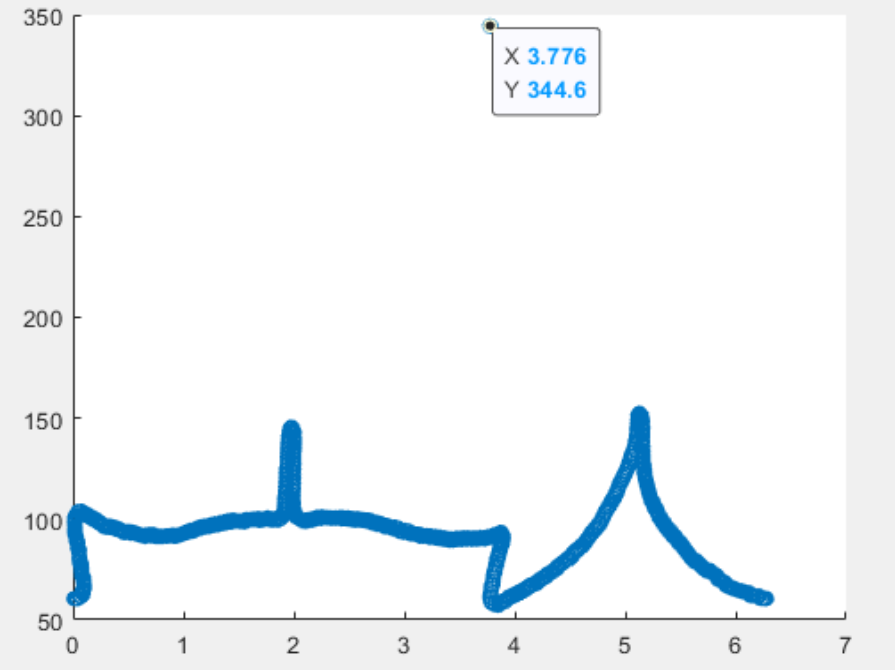

Fig12.R-theta Plot for Test3

There are two outside line points at (3.776,344,6), because there are two white point at left bottom corner of test3. If need to filter out the two points, the bwareafilt function can also be use, the filtered plot is shown as figure 13 below:

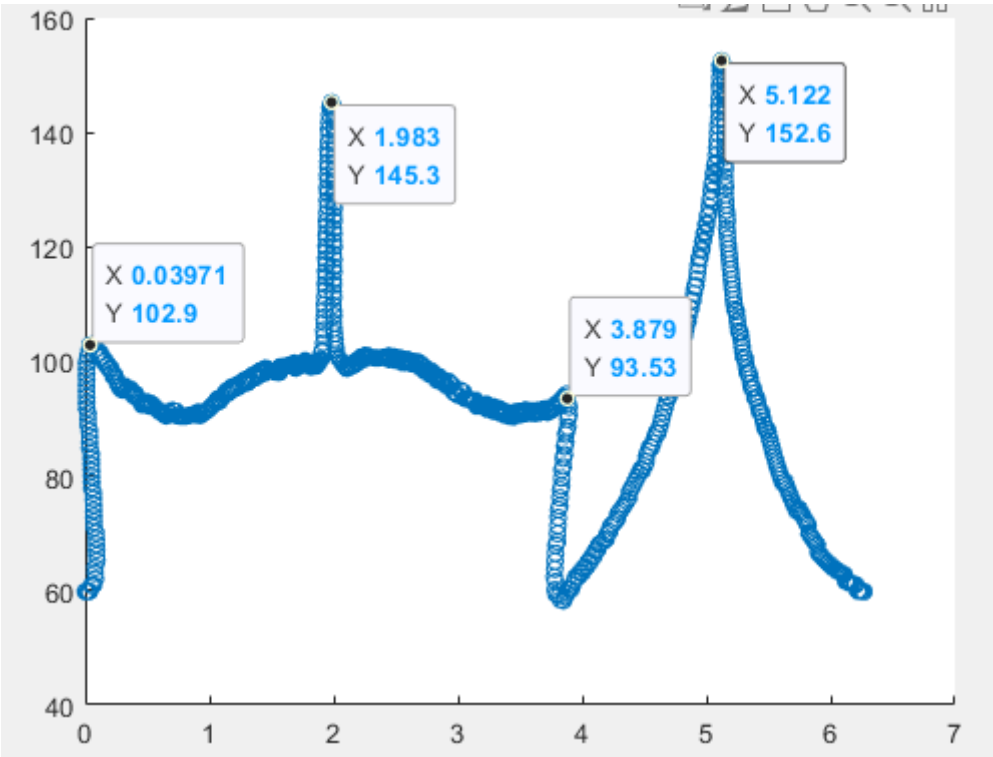

Fig13. Filtered Plot

For peaks, peak at 1.983 is according to petiole and peak at 5.122 is according to highest tip of leaf and peaks at 0.03971 and 3.879 are according to other to tips of leaf.

D. Hough Transform

For hough transform, the edge plot of letter should be generated first, use the same boundary function as part A, which is shown as figure14 below:

```
%--- 1.
I = imread('./letter.bmp');
figure(1);
imshow(I);
hold on;
%Compute the edge map of the test image.
[T,Iout] = intermeans(I);
I_edge = bwperim(Iout,8);
figure(2);
imshow(I_edge);
```

Fig14.Code of Edge Plot

Here slope-intercept is chased to denote and detect straight lines. Y = aX + b, here the range of a is (-a_range,a_range), the range of b is (-b_range,b_range), and a_step and b_step are total number of a and b. a_de and b_de are difference between two neighbor values. For each a, a b value is generated with edge points, and the b will be rounded to nearest b in bin.

However, here also use the same coordinate transform mentioned before.

The code of hough transform is shown as figure15 below:

```
%Hough Transform
a_range =20;
a_step = 81;
b_range = 50000;
b_step = 20001;
a_de = 2*a_range/(a_step-1);
b_de = 2*b_range/(b_step-1);
a = zeros(1,a_step);
for i = 1:a_step
    a(i) = -a_range + a_de*(i-1);
end
b = int32(linspace(-b_range,b_range,b_step));
A = zeros(a_step,b_step);
for x = 0:size(I_edge,2)-1
    for y = 0:size(I_edge,1)-1
        for i = 1:a_step
            if I_edge(size(I_edge,1)-y, x+1) == 1
                b_new = int32(-x * a(i) + y);
                if b_new < -b_range + b_de
                    b_r = -b_range;
                elseif b_new > b_range - b_de
                    b_r = b_range;
                elseif mod(b_new,b_de) == 0
                    b_r = b_new;
                elseif mod(b_new,b_de) >= b_de/2
                    b_r = (idivide(b_new,b_de,'floor') + 1) * b_de;
                else
                    b_r = idivide(b_new,b_de,'floor') * b_de;
                end
                j = b_r / b_de;
                j = j + (b_step-1)/2 + 1;
                A(i,j) = A(i,j) + 1;
            end
        end
    end
end
```

Fig15.Code of Hough Transform

To find n highest number in accumulator matrix A, A is sorted in increase to column first. Then the last n numbers position can be retransformed to original A position to get a and b value. Therefore, corresponds a and b values will be found.

The code of find n highest is shown as figure16 below:

```
%find n highest
n=20;
[AS,pos] = sort(A(:));
a_result = zeros(n,1);
b_result = zeros(n,1);
for i = 0:n-1
    if mod(pos(a_step*b_step-i),a_step)~=0
        a_result(i+1) = mod(pos(a_step*b_step-i),a_step)*a_de-a_range-a_de;
        a_mod =  mod(pos(a_step*b_step-i),a_step);
    else
        a_result(i+1) = a_step*a_de-a_range-a_de;
        a_mod = a_step;
    end
    b_result(i+1) = (pos(a_step*b_step-i)-a_mod)/a_step*b_de-b_range;
```

Fig16.Code of Find n Highest

However, some overlaps lines may appear at the same straight line in picture. To avoid overlap, the width of picture is divided as 8 points (take eight equidistant points on the X axis). Lines with similar a value and have similar y in one of 8 points, the line is regarded as overlap line as before, will not be plot.

The code of avoid overlap is shown as figure17 below:

```
        %avoid overlap
        flag = 0;
        for j = 1:i+1
            if (abs(a_result(j)-a_result(i+1))<=a_de) && (j~=i+1)
                for n =0:8
                    x_mid = size(I,2)/8*n;
                    if (abs(a_result(j)*x_mid+b_result(j)-a_result(i+1)*x_mid-b_result(i+1))<=10)
                        flag = 1;
                        break;
                    end
                end
            end
            if flag == 1
                break;
            end
        end
        if flag == 0
            p1_x = 0;
            p1_y = a_result(i+1)*p1_x+b_result(i+1);
            p2_x = 500;
            p2_y = a_result(i+1)*p2_x+b_result(i+1);
            p1 = [size(I,1)-p1_y,p1_x+1];
            p2 = [size(I,1)-p2_y,p2_x+1];
            figure(1);
            plot([p1(2),p2(2)],[p1(1),p2(1)],'Color','r','LineWidth',2)
        end
    end
    saveas(gcf,'letter_line','bmp');
```

Fig17.Code of Avoid Overlap

The line letter is shown in figure18 below:

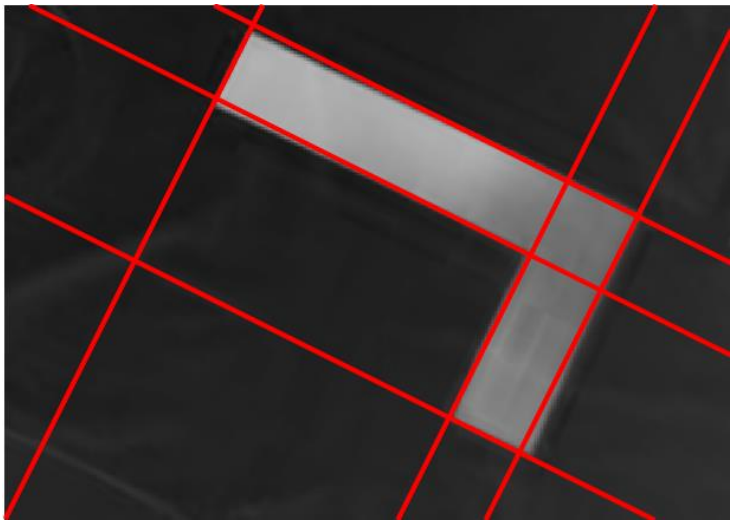

Fig.18 Line Letter