A. Feature Measurement

2. Input image is test1.bmp (image I). Implement the intermeans algorithm to calculate the threshold T1 and use it to threshold image I. The output image is I1.

```matlab
%To calculate the intermeans threshold;
%input is the gray level image test1 .
%output is the threshold value T and the binary thresholded image Iout.

function [T ,Iout] = intermeans(Iin)

[h, g] = imhist(Iin);
h = h.';
T = round(mean2(Iin)); %initial threshold value
T_prev = NaN;

while T_prev ~= T
    T_prev = T;
    g_low = g(1):1:T_prev;
    mean_low = sum(g_low .* h(1:1:length(g_low))) / sum(h(1:1:length(g_low))); %u1

    g_high = T_prev+1:1:g(end);
    mean_high = sum(g_high .* h(length(g_low)+1:1:g(end)+1)) / sum(h(length(g_low)+1:1:g(end)+1)); %u2
    T = floor((mean_low + mean_high) / 2); %update threshold
end

T_norm = (T - g(1)) / (g(end) - g(1)); %normalize threshold for im2bw
Iout = im2bw(Iin, T_norm);

end
```

Fig. 1 Intermeans algorithm in MATLAB

```
>> run_A
T =
   113
```

Fig. 2 Threshold T1 value obtained from MATLAB

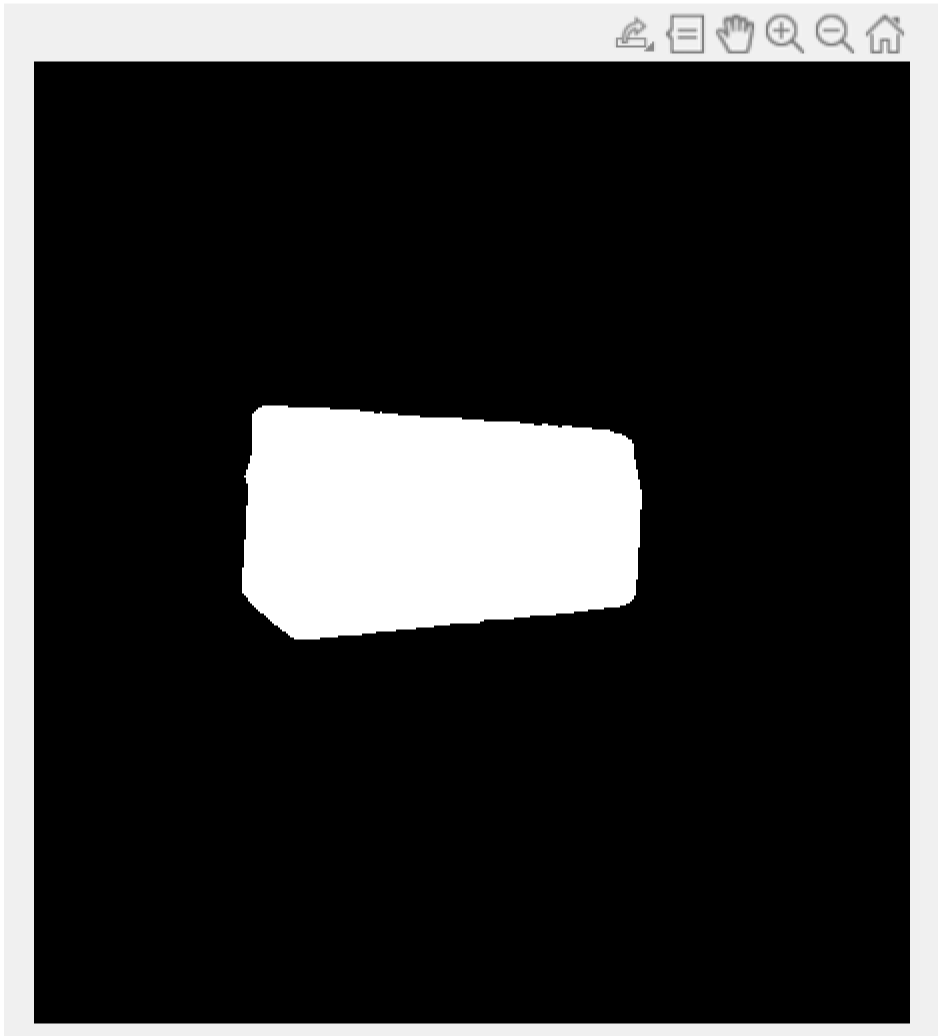

Fig. 3 Output of test1.bmp (image I)

With reference to Figure 2, the threshold T1 is 113. Using T1 to threshold image I, output image I1 is obtained and shown in Figure 3.

3. Algorithm to calculate image features – perimeter, area, compactness, centroid and first invariant moment.

```
%To compute the features
%input is the binary thresholded image
%outputs are the feature values
function [P, A, C, xbar, ybar, phione] = features(Iin)

biggest_Iin = bwareafilt(Iin, 1); %filter to get the largest boundary

%find area
struct_A = regionprops(biggest_Iin, 'Area');
A = struct_A.Area;

%find perimeter
struct_P = regionprops(biggest_Iin, 'Perimeter');
P = struct_P.Perimeter;

%find compactness
C = P^2 / (4 * pi * A);

%find centroid
[r, c] = size(Iin);
m = zeros(r, c);
B = flip(Iin); %flip matrix to make sure origin follow xy plane
for i = 0:1
    for j = 0:1
        for y = 1:r
            for x = 1:c
                m(i+1, j+1) = m(i+1, j+1) + ((x-1)^i*(y-1)^j*B(y,x));
            end
        end
    end
end
xbar = m(2,1)/m(1,1);
ybar = m(1,2)/m(1,1);
```

Fig. 4 Algorithm to calculate image features part I

```
%first invariant moment
u = [ 0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
for i = 0:3
    for j = 0:3
        for y = 1:r
            for x = 1:c
                u(i+1, j+1) = u(i+1, j+1) + ((x-1-xbar)^i*(y-1-ybar)^j*B(y,x));
            end
        end
    end
end

n = [ 0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
for i=0:3
    for j=0:3
        n(i+1, j+1) = u(i+1, j+1) / (u(1,1)^(1+(i+j)/2));
    end
end

phione = n(3,1)+ n(1,3); %n20 + n02

end
```

Fig. 5 Algorithm to calculate image features part II

```
P =

  567.0810


A =

       20016


C =

    1.2785


xbar =

  199.3693


ybar =

  250.8056


phione =

    0.1983
```

Fig. 6 Image features of image I1

With reference to Figure 6, image I1 has perimeter of 567.0810, area of 20016 and calculated compactness of 1.2785. Its centroid is located at (199.3693, 250.8056) and the first invariant moment is 0.1983.

B. Feature Invariance

2. The test image is test2.bmp (image J). What do you think is the optimum threshold Topt for segmenting the object accurately?

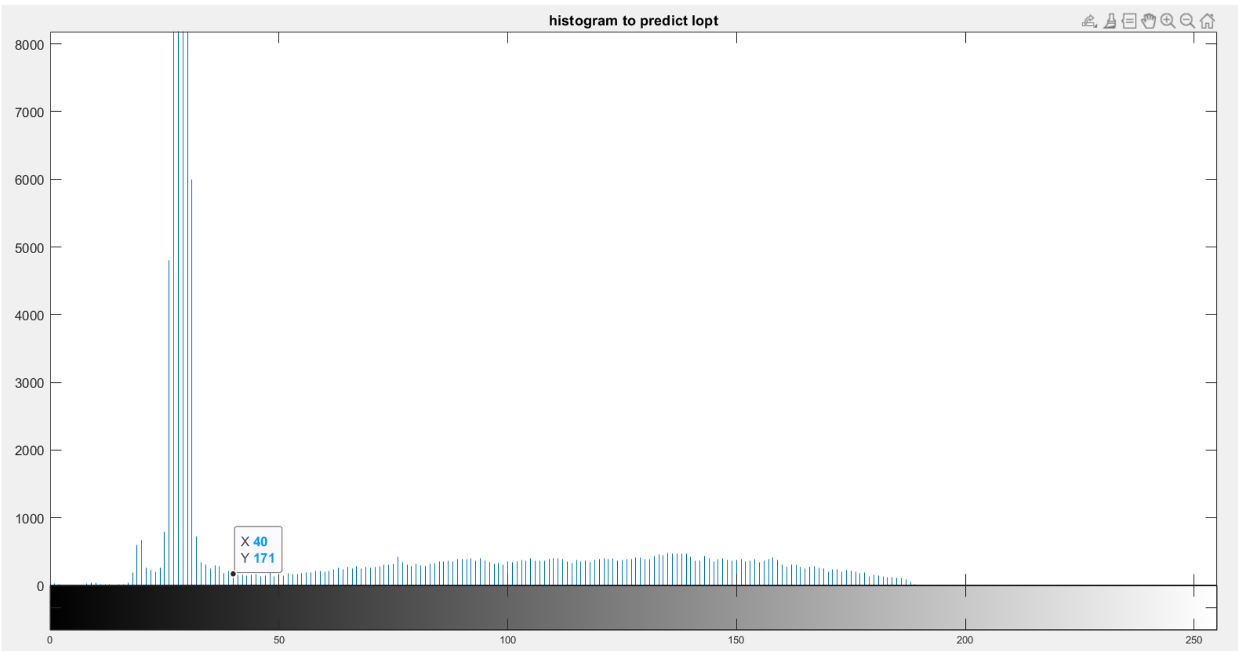

Fig. 7 Histogram plot of image J

Based on the histogram plot of image J, the predicted optimum threshold Topt is approximately 40. With reference to Figure 7, it can be observed that by thresholding at gray level 40, 2 prominent valleys will be obtained which is a good indication of the pixels corresponding to the object and the background.

3. Obtain the intermeans threshold T2 using intermeans.m.

```
>> run_B
T =
     79
```

Fig. 8 Threshold T2 value obtained from MATLAB

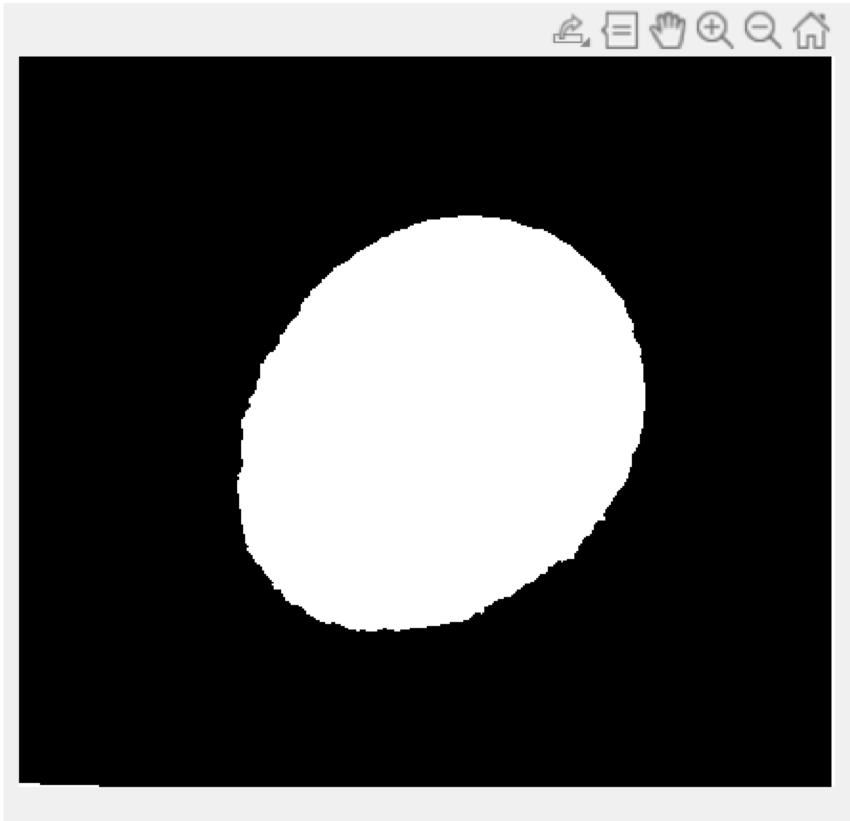

Fig. 9 Output of test2.bmp (image J) using threshold T2

With reference to Figure 8, the threshold T2 is 79. Using T2 to threshold image J, output image J is obtained and shown in Figure 9.

4. Threshold image J using T2 and measure the features using features.m.

```
P =

  698.3450


A =

       35412


C =

    1.0959


xbar =

  222.9953


ybar =

  187.9785


phione =

    0.1899
```

Fig. 10 Image features of image J after thresholding at T2

With reference to Figure 10, image J after thresholding at T2 has perimeter of 698.345, area of 35412 and calculated compactness of 1.0959. Its centroid is located at (222.9953, 187.9785) and the first invariant moment is 0.1899.

5. Threshold J with threshold Topt and measure the features using features.m. Compare the segmentation results obtained with T2 and Topt.

```
Topt =
    40


P =

  756.5600


A =

       43903


C =

    1.0375


xbar =

  219.5583


ybar =

  178.1873


phione =

    0.1886
```

Fig. 11 Image features of image J after thresholding at Topt

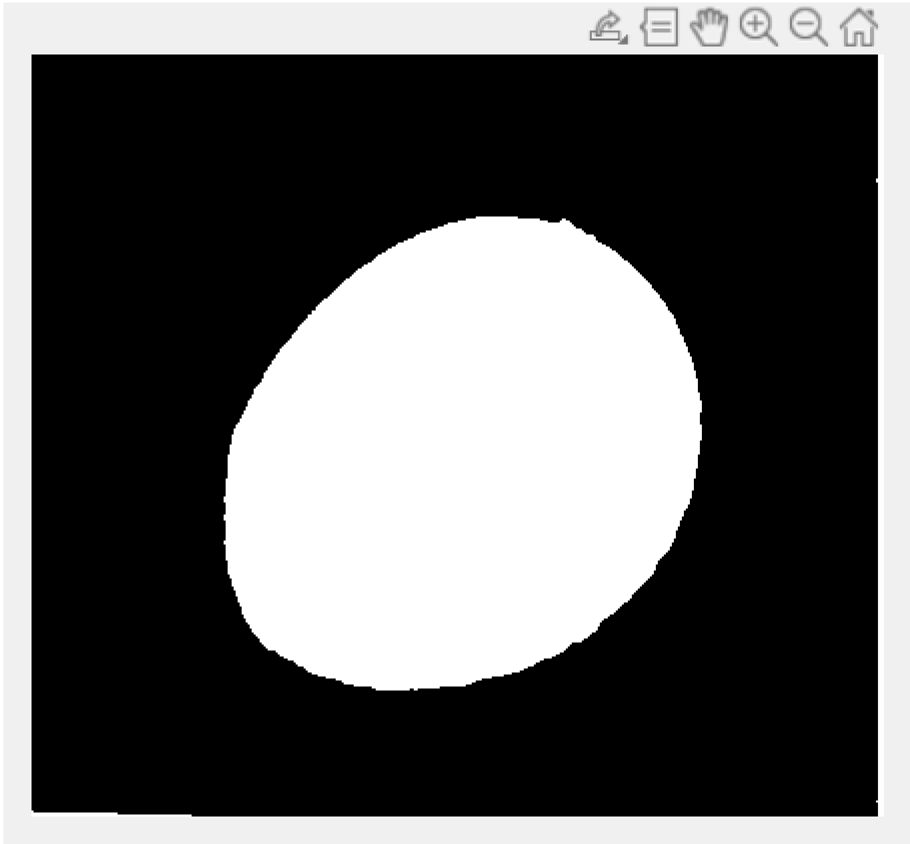

Fig. 12 Output of test2.bmp (image J) using threshold Topt

6. Compare the segmentation results obtained with T2 and Topt. Discuss the sensitivity of the measured feature values to the threshold values.

As shown in Figure 9 and Figure 12, segmentation result with Topt produces a smoother boundary across the oval shape compared to T2. This is because the Topt value is significantly lower than T2. By thresholding at Topt instead of T2, this means more gray levels will be mapped to 1. This is evident from the larger perimeter and area, and thus, compactness obtained using Topt compared to T2, as shown in Figure 10 and Figure 11.

C. Boundary Plot

```
%To compute the r-theta plot
%input is a boundary image 'test3.bmp'
%output is the array containing the rtheta value
function [r, theta] = rtheta(Iin)
Iin = im2bw(Iin);
[rows, cols] = size(Iin);
m = zeros(rows, cols);
B = flip(Iin); %flip matrix to satisfy xy plane
for i = 0:1
    for j = 0:1
        for y = 1:rows
            for x = 1:cols
                m(i+1, j+1) = m(i+1, j+1) + ((x-1)^i*(y-1)^j*B(y,x));
            end
        end
    end
end
xbar = m(2,1)/m(1,1);
ybar = m(1,2)/m(1,1);
radius = []; %store the radius
theta = []; %store the theta
for y = 1:rows
    for x = 1:cols
        if B(y,x) == 1
            temp_radius = sqrt((x - 1 - xbar) ^ 2 + (y - 1 - ybar) ^ 2);
            temp_theta = atan2d(y - 1 - ybar, x - 1 - xbar) + (360 *((y - 1 - ybar) < 0)); %if y-axis is negative value, add 360 to make it positive
            radius(end+1) = temp_radius;
            theta(end+1) = temp_theta;
        end
    end
end
r = radius;
end
```

Fig. 13 Algorithm to calculate r-theta plot

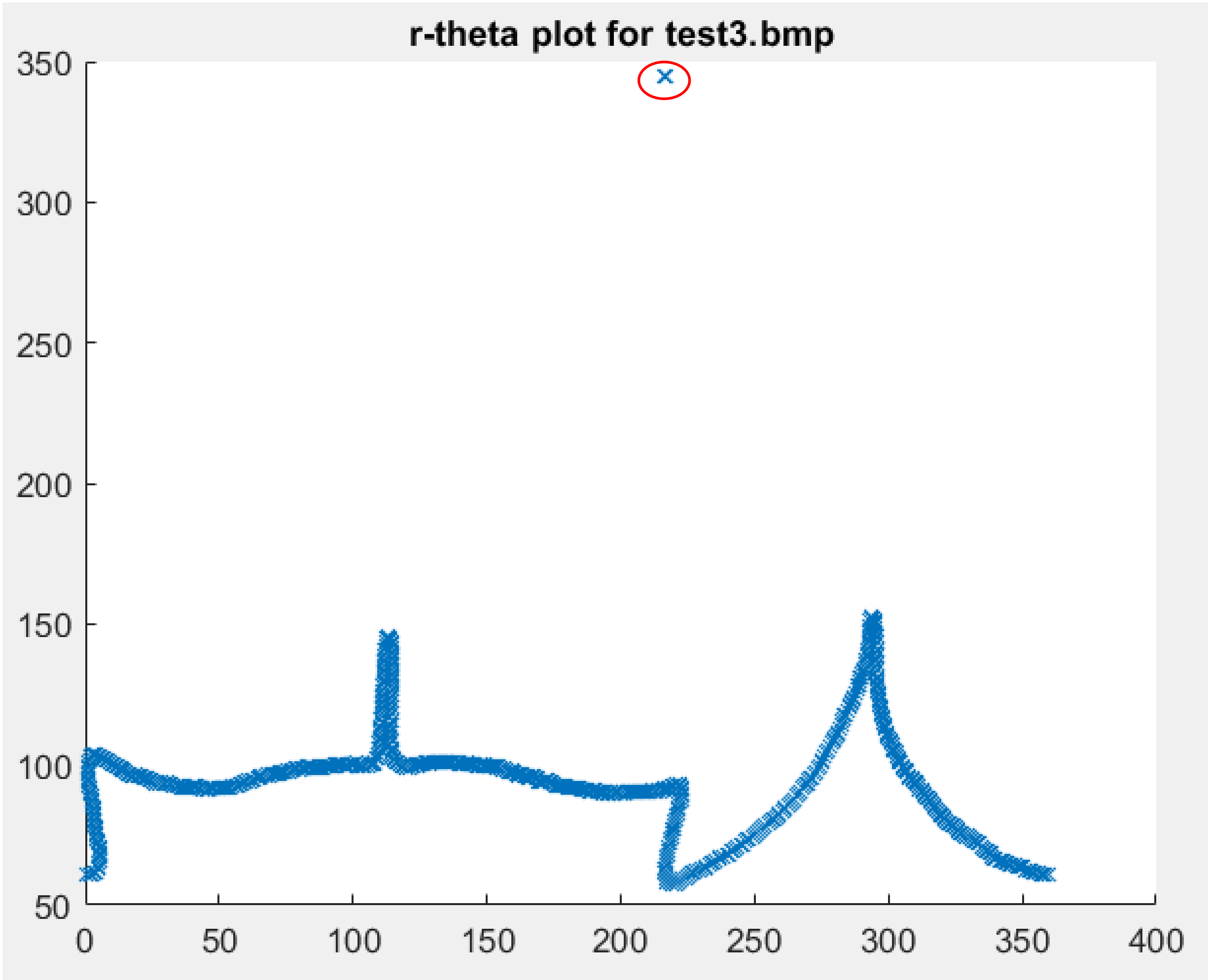

Fig. 14 r-theta plot for test3.bmp (image K)

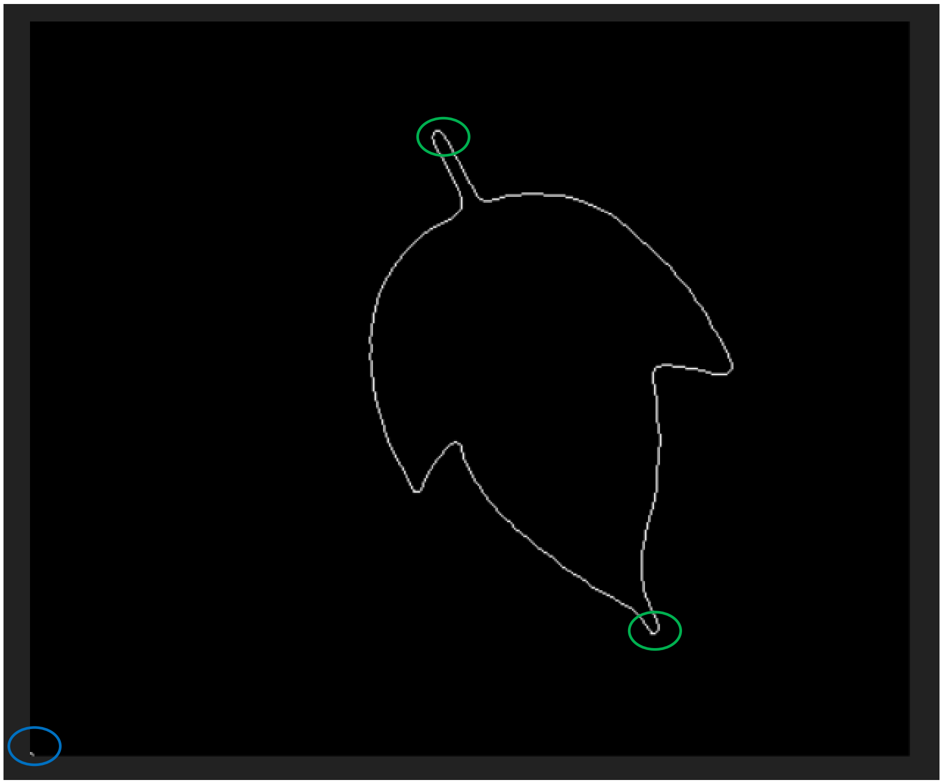

Fig. 15 image K

Using the algorithm shown in Figure 13, r-theta plot for image K can be obtained as seen in Figure 14. There is also an anomalous point that is observed from the r-theta plot of image K (circled in red in Figure 14). As shown in Figure 15, this point can be ignored as it actually corresponds to the white pixel circled in blue, which is not part of the boundary of the object. In addition, 2 spikes can be observed in the r-theta plot for image K, these 2 spikes correspond to the 2 boundary points furthest away from the centroid (circled in green in Figure 15).