CA2 Report
He Tianyu
A0177829J

A - Feature Measurement

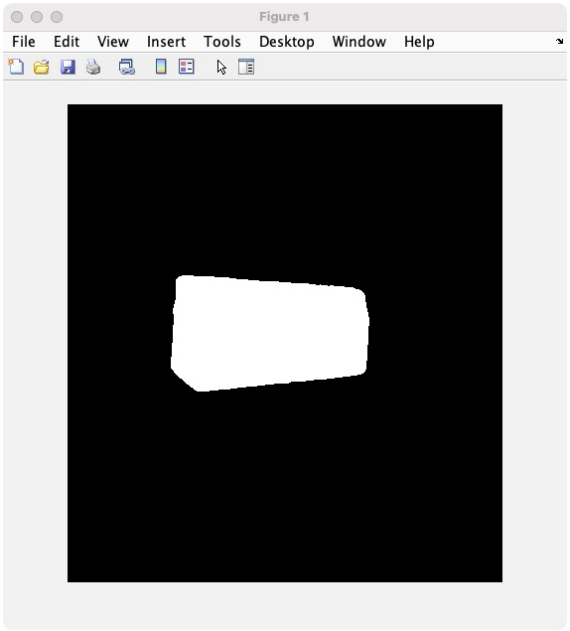
Test image is test1.bmp

test1.bmp

Implement intermeans.m to calculate the threshold, the threshold T is **114**.
Function intermean.m makes use of intermeans algorithm to calculate threshold. The result is given when T stabilizes.

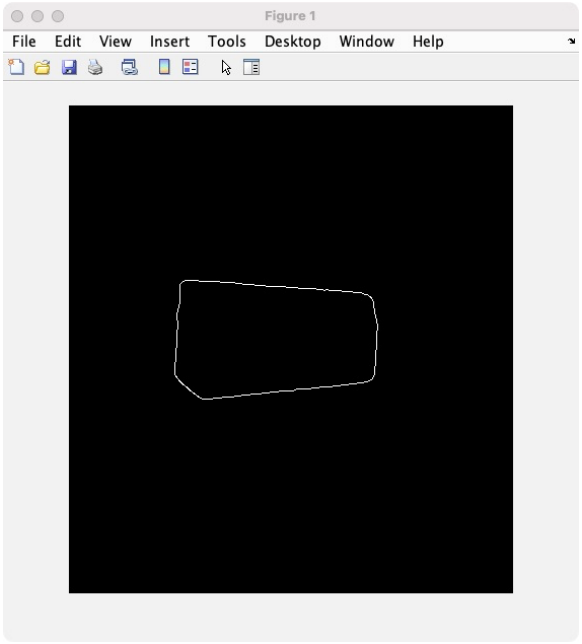
```
function [T,Iout] = intermeans(Iin)
    T_old = 0;
    T = mean(mean(Iin)); % start with initial T = average intensity
    while T_old ~= T % while not equal, end when Threshold stablized
        T_old = T;
        mean_1 = mean(mean(Iin(Iin < T_old)));
        mean_2 = mean(mean(Iin(Iin > T_old)));
        T = (mean_1 + mean_2) / 2; % update new Threshold value
    end
    T = ceil(T); % ceil the Threshold
    Iout = Iin > T; % for those pixels with value greater than threshold, display
end
```

The output image I1 is shown below



I1

Image I1 is the input image of features.m.
The input image I1 is a binary image. We flip the image vertically to set the origin from top left corner to bottom left corner.
With build-in Sobel edge detection method, the edge map is shown below.



Edges detected

The features are calculated.

perimeter **P = 567**
area **A = 19996**
compactness **C = 1.2794**
centroid is given by **$(\bar{x}, \bar{y})$ = (200.3465, 251.7960)**
invariant moment **ϕ_1 = 0.1983**

The detailed formula used are shown in the comments of the code.

Perimeter is given by counting the edge pixels, and Area is given by counting the pixels with value 1 since the input image is a binary image.

Compactness is given by formula

$$C = \gamma = \frac{P^2}{4\pi \times A}$$

```
function [P, A, C, xbar, ybar, phione] = features(Iin)
    % Iin is a binary image with only 1s and 0s
    % centralize the image to largest area: bwareafilt(I, n)
    Iin = bwareafilt(Iin,1);
    % flip image to set origin from top left to bottom left
    I_flip = flip(Iin,1);
    % use built-in function to track the edges
    I_edges = edge(Iin);

    % Numer of edge pixels
    P = sum(sum(I_edges));
    % Number of pixels within
    A = sum(sum(Iin));
    % Compactness = (perimeter)^2 / (4Pi*area)
    C = P^2 / (4 * pi * A);

    % Centroid: xbar = m10/m00, ybar = m01/m00
    % mpq = sum(sum(x^p * y^q * f(x,y)dxdy))
```

Centroid is given by formula

$\bar{x}$ = m10/m00
$\bar{y}$ = m01/m00

```
%m00 = summing up all the pixels
m00 = sum(sum(I_flip));
%m01 = sumall y * f(x,y)
%m10 = sumall x * f(x,y)
m01 = 0;
m10 = 0;
for y = 1:size(I_flip,1) % row by row (1 - row size)
    for x = 1:size(I_flip,2) % col by col (2 - col size)
        m01 = m01 + y * I_flip(y, x);
        m10 = m10 + x * I_flip(y, x);
    end
end
% Centroid: xbar = m10/m00, ybar = m01/m00
xbar = m10/m00;
ybar = m01/m00;
```

ϕ_1 is given by a series of formula

$$\phi_1 = \eta_{20} + \eta_{02}$$

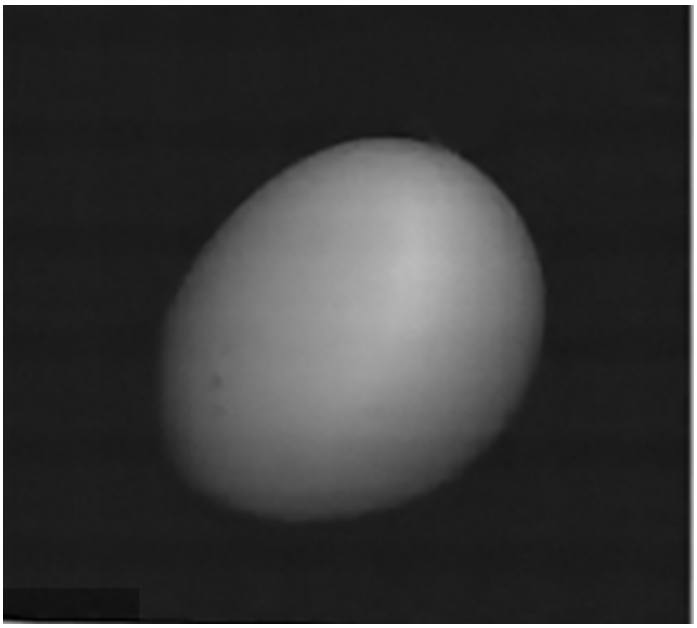
$$\eta_{20} = \frac{\mu_{20}}{\mu_{00}^2}, \eta_{02} = \frac{\mu_{02}}{\mu_{00}^2}$$

$$\mu_{20} = \sum_x \sum_y (x - \bar{x})^2 f(x,y), \mu_{02} = \sum_x \sum_y (y - \bar{y})^2 f(x,y)$$

```
% phi1 = eta20 + eta02
% eta20 = u20/u00^2
% upq = sumsum (x - xbar)^p * (y - ybar)^q * f(x,y) dxdy
% u20 = sumsum (x-xbar)^2 * f(x,y); u02 = sumsum (y-ybar)^2 * f(x,y)
% u00 = m00
% u20 and u02
u20 = 0;
u02 = 0;
for y = 1:size(I_flip,1) % row
    for x = 1:size(I_flip,2) % col
        u20 = u20 + (x - xbar)^2 * I_flip(y, x);
        u02 = u02 + (y - ybar)^2 * I_flip(y, x);
    end
end

u00 = m00;
eta20 = u20 / u00^2; % (2+0)/2+1 = 2
eta02 = u02 / u00^2;

phione = eta20 + eta02;
```

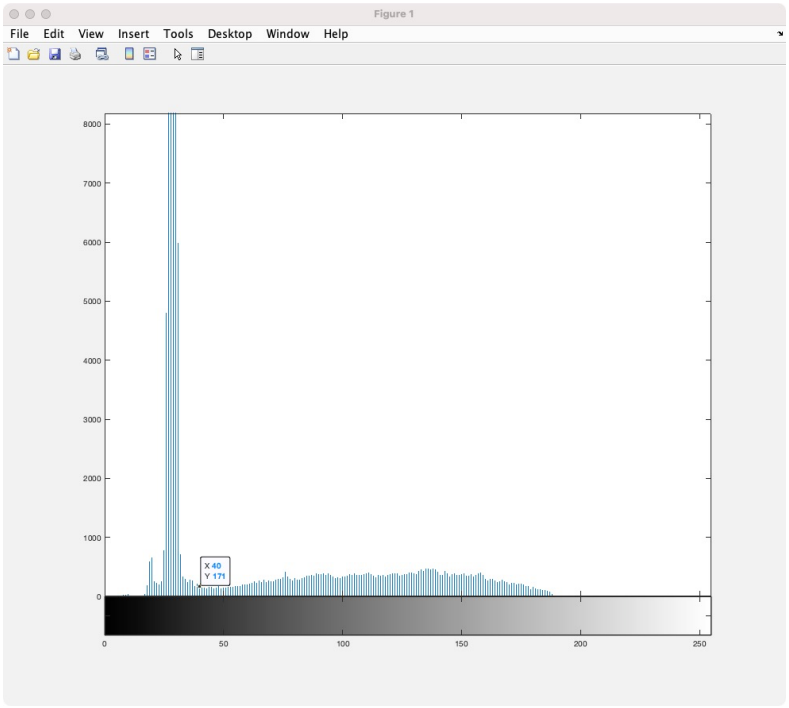
B – Feature Invariance

The test image is test2.bmp.



test2.bmp

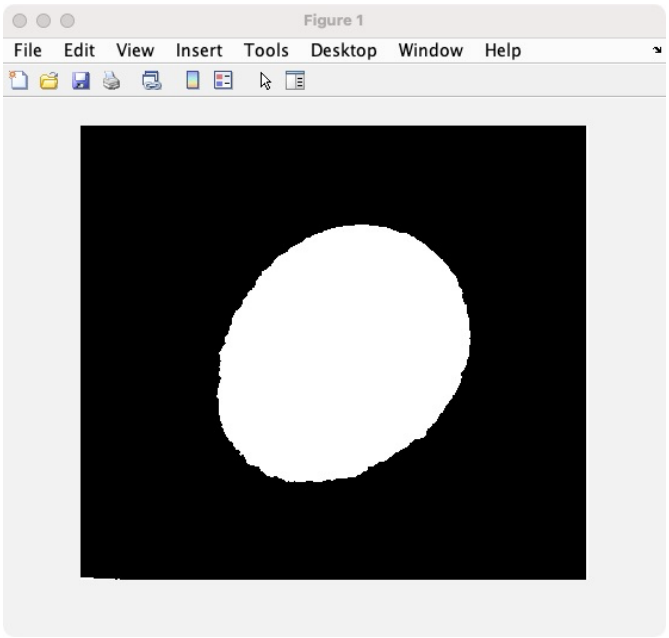
With build-in function imhist (), an optimal threshold **Topt is determined to be 40**.



While by intermeans algorithm, a threshold is given to be 80.

With threshold **T2 = 80,** the thresholded image is shown as below. The bottom left noise will be removed by centralization later in features.m

```
% centralize the image to largest area: bwareafilt (I, n)
 Iin = bwareafilt (Iin,1)
```

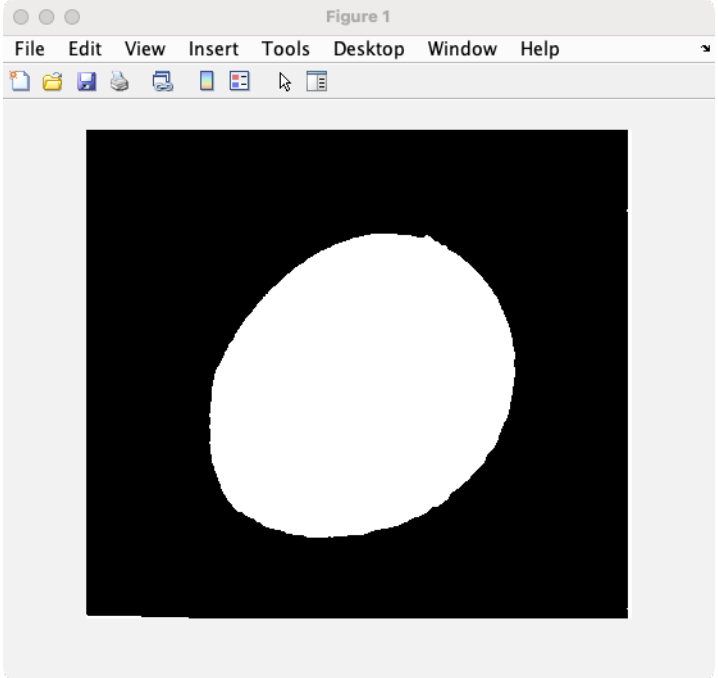


T2 = 80

The features given by features.m and threshold T2 are

P=640
A=35108
C=0.9284
$\bar{x}$**=220.1020**
$\bar{y}$**=189.5594**
ϕ**1=0.1630**

With threshold Topt = 40, the thresholded image is



Topt = 40

Compared to thresholded image with T2 = 80, thresholded image with Topt = 40 has similar shape while occupies larger area for the reason that pixels with gray level from 40 to 80 are now accepted as logical 1s.

The features returned by features.m are now

P=706
A=43093
C=0.9053
$\bar{x}$**=215.7495**
$\bar{y}$**=179.3796**
ϕ**1=0.1627**

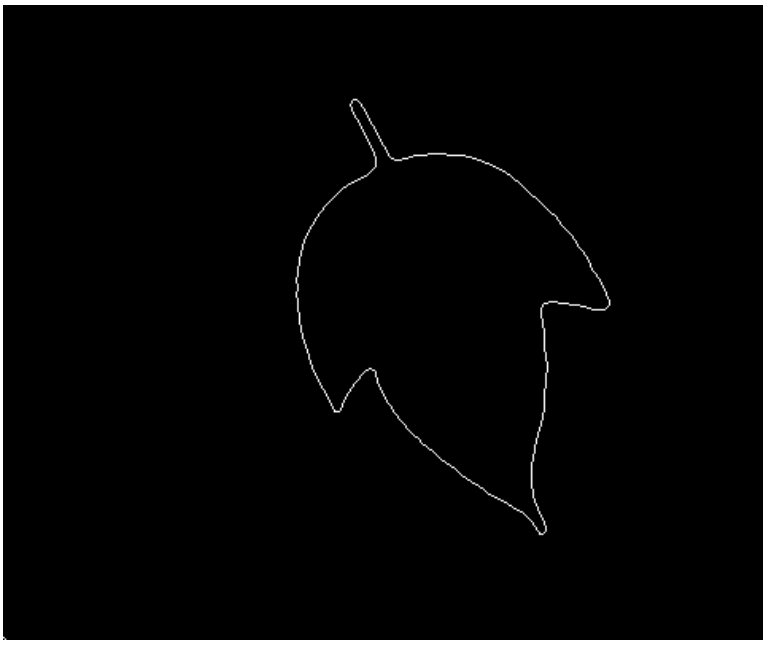
ϕ**1** basically remain the same, Compactness **C** decreases, **P, A** increase a lot, and centroid slightly shifts.

It can be concluded that with decreased threshold, while other factors do not change

1. Compactness may or may not decrease, up to its bounding shape.
2. Perimeter and Area increases.
3. Centroid may shift due to more pixels involved.
4. Invariant moment remains constant.

C. Boundary Plot

Test image is test3.bmp. There are two exceptional pixels at bottom left.



test3.bmp

The centroid may be estimated from the boundary pixels.
The coordinates of centroid are given by

$$\bar{x} = \frac{1}{N}\sum_i x_i, \bar{y} = \frac{1}{N}\sum_i y_i$$

```
% find centroid
xbar = 0;
ybar = 0;
for y = 1:size(I_flip,1)
    for x = 1:size(I_flip,2)
        xbar = xbar + x * I_flip(y, x);
        ybar = ybar + y * I_flip(y, x); % sum up all 1s
    end
end
% centroid
xbar = xbar / sum(sum(I_flip));
ybar = ybar / sum(sum(I_flip));
```

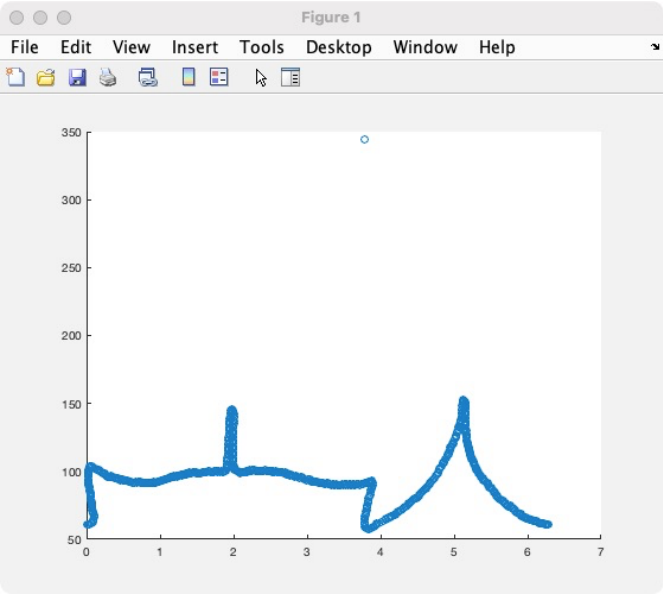
Then, the r and θ for each pixel are given by

$$r = \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}$$

$$\theta = \tan^{-1}\left(\frac{y - \bar{y}}{x - \bar{x}}\right)$$

```
r = zeros(0);
theta = zeros(0); % create 2 lists
% r = sqrt((x-xbar)^2+(y-ybar)^2)
% theta = tan-1((y-ybar)/(x-xbar))
for y = 1:size(I_flip,1)
    for x = 1:size(I_flip,2)
        if I_flip(y, x)
            r(end + 1) = sqrt((x - xbar)^2 + (y - ybar)^2);
            theta(end + 1) = mod(atan2(y - ybar, x - xbar), 2*pi);
        end
    end
end
```

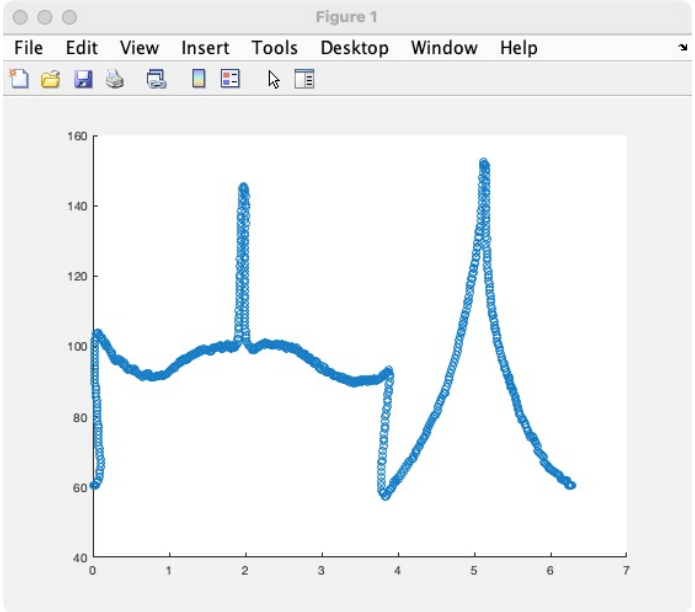
Lastly, the r – θ plot is given by build in function scatter ()



There are two exceptional points at θ = 3.776 and 3.78. They correspond to the two pixels at bottom left corner of image test3.bmp.

The two peaks correspond to the shape of the leaf. The first peak corresponds to upper sharp shape, while the second peak corresponds to bottom sharp shape.

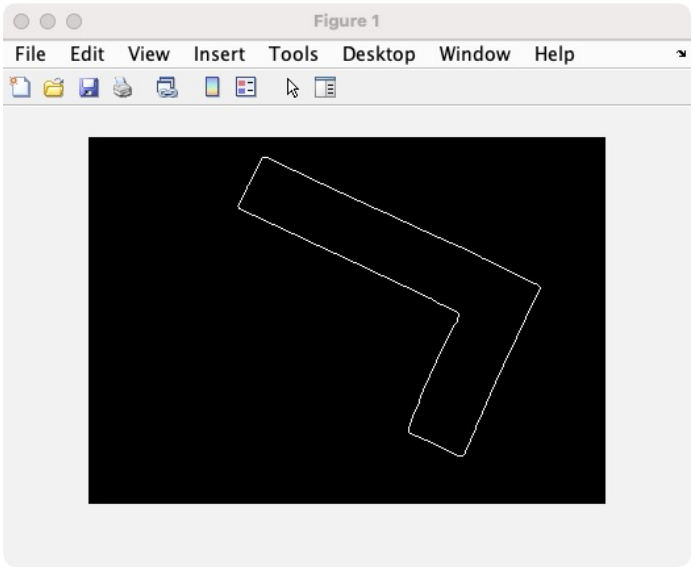
In case we want to eliminate the two exceptional points, we can set the boundary x > 50 instead of x >= 1 to remove the two exceptional pixels in test3.bmp, which gives the plot



D. Hough Transform

The test image is letter.bmp

Firstly, we use build in function edge() to compute the edge map, which is given as



We will then approach to do the Hough Transform using normal representation

$$(x_i, y_i) \rightarrow \rho = x_i \cos\theta + y_i \sin\theta$$

Where maxrho is given by following formula with x and y be the size of edge map.

$$\rho = \rho_i \cos(\theta - \theta_i)$$

$$\rho_i = \sqrt{x_i^2 + y_i^2}, \quad \theta_i = \tan^{-1}(y_i/x_i)$$

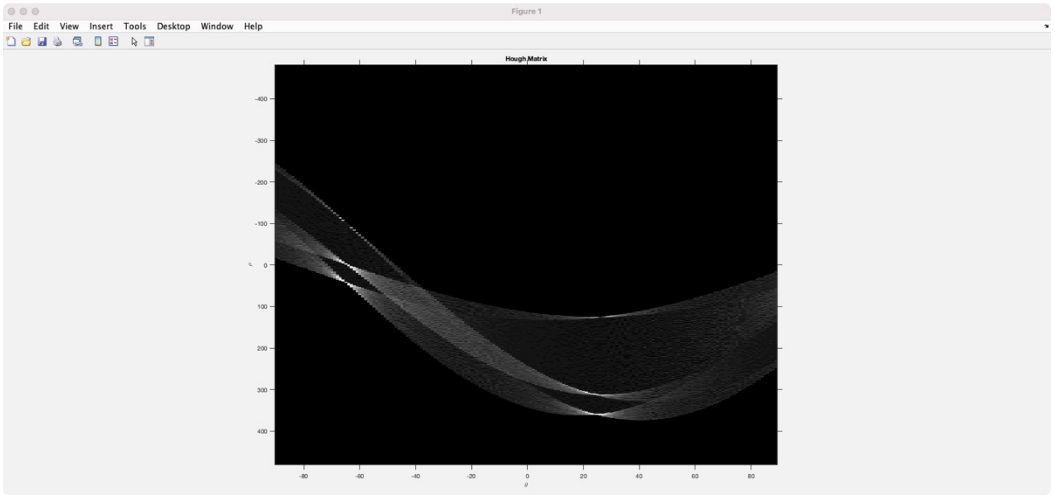
Since ρ has both positive and negative values, the total size required for accumulator array will be 2 * maxrho with θ in the range 1-180 degrees.

We will then store the $\rho - \theta$ pair in a 2-D accumulator array of size [2*maxrho, 180], and do the Hough Transform. A point in image space is represented by a sinusoid in $\rho - \theta$ plane.

```
    % 3. Hough Transform
    [y,x]=find(I_edge);
    [sy,sx]=size(I_edge); % 279 * 393

    totalpix = length(x); % 676 1s
    maxrho = round(sqrt(sx^2 + sy^2)); % 482

    HM = zeros(2*maxrho,180);

    for cnt = 1:totalpix
        cnt2 = 1;
        for theta = -pi/2:pi/180:pi/2-pi/180
            rho = round(x(cnt).*cos(theta) + y(cnt).*sin(theta));
            HM((rho+maxrho),cnt2) = HM(rho+maxrho,cnt2) + 1;
            cnt2 = cnt2 + 1;
        end
    end
```

We then plot the Hough Transform with following codes. This part is commented for later plot.

```
% make it brighter
%for i = 1:maxrho
%    for j = 1:180
%        HM(i,j) = HM(i,j) * 10;
%    end
%end

%Plot Hough Transform
%theta = rad2deg(-pi/2:pi/180:pi/2-pi/180);
%rho = -maxrho:maxrho-1;
%imshow(uint8(HM),[],'xdata',theta,'ydata',rho);
%xlabel('\theta'),ylabel('\rho')
%axis on, axis normal;
%title('Hough Matrix');
```



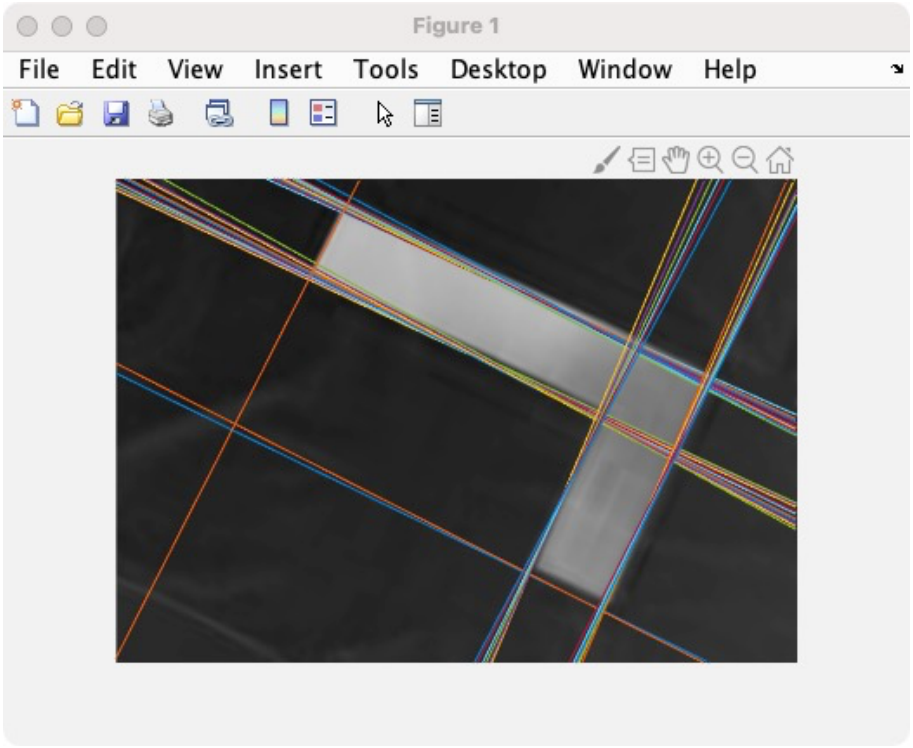
Hough Transform Plane

Here, we want to transform back and plot the straight lines by finding the most voted points Hough Transform plane.

Note we need to minor maxrho for every x-index of HM matrix since we added maxrho to avoid negative index in previous transform.

```
% Transform back & plot
figure;
imshow(I); hold on;
for i = 1:maxrho*2
    for j = 1:180
        if HM(i,j)>33
            ans_rho = i - maxrho;
            ans_x = 0:sx;
            ans_y = (ans_rho - ans_x*cosd(j-90))/sind(j-90);
            plot(ans_x,ans_y);hold on;
            xlim([0 rows])
            axis equal
        end
    end
end
```

By several times of practice, the threshold is set to be 33. The resulted letter_line.bmp is shown as



Finally, save the figure to a bmp file named 'letter_line.bmp'.

```
saveas(gcf,'letter_line.bmp')
```