

# CSE185

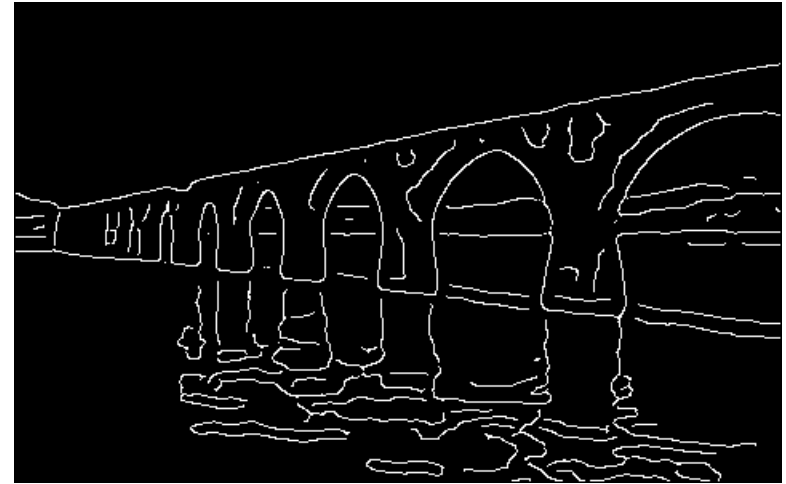
## Introduction to Computer Vision

### Lab 08: Hough Transform

Instructor: Prof. Ming-Hsuan Yang  
TA: Tiantian Wang & Tsai-Shien Chen

# Line Fitting

- Given an input image, detect straight lines from the edge map



# Line Equation in Cartesian Coordinate

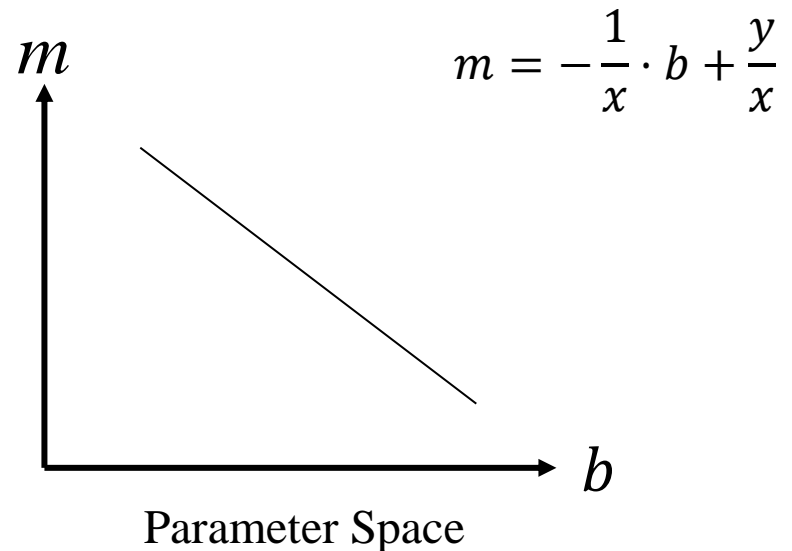
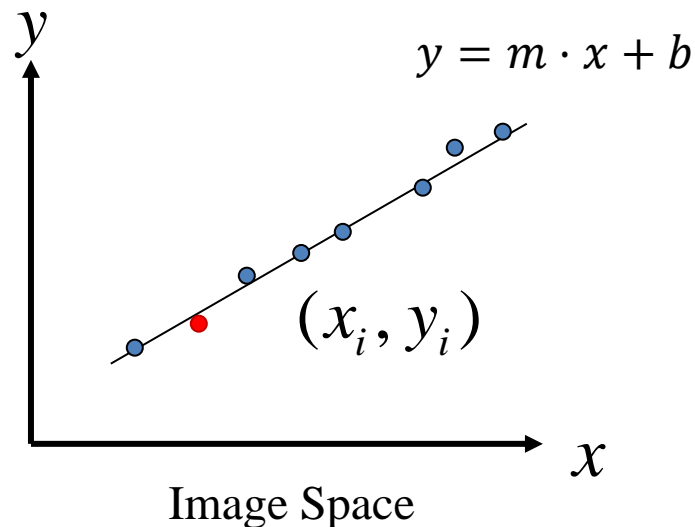
- In image space:

$$y = m \cdot x + b$$

- In parameter/Hough space:

$$m = -\frac{1}{x} \cdot b + \frac{y}{x}$$

$m$ : slope/gradient  
 $b$ : intercept/offset



# Line Equation in Cartesian Coordinate

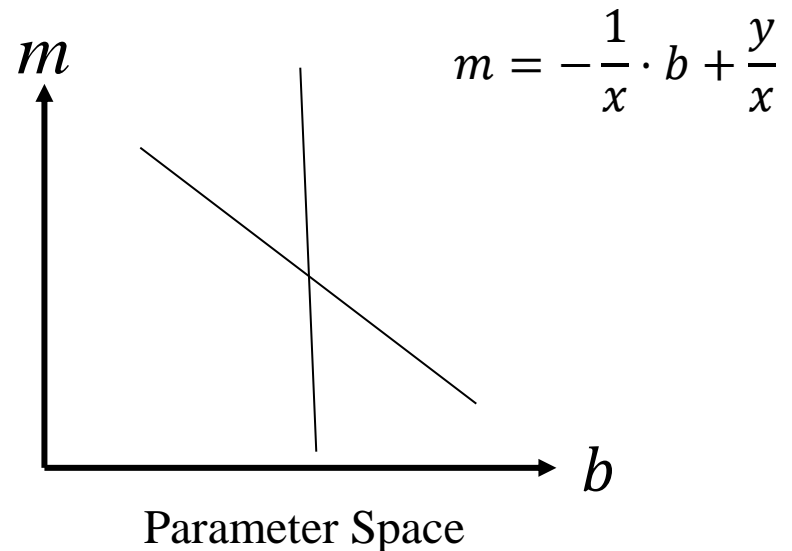
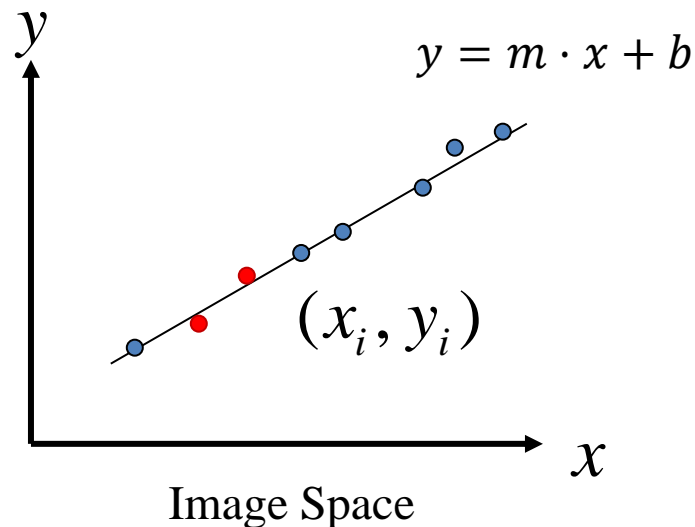
- In image space:

$$y = m \cdot x + b$$

- In parameter/Hough space:

$$m = -\frac{1}{x} \cdot b + \frac{y}{x}$$

$m$ : slope/gradient  
 $b$ : intercept/offset



# Line Equation in Cartesian Coordinate

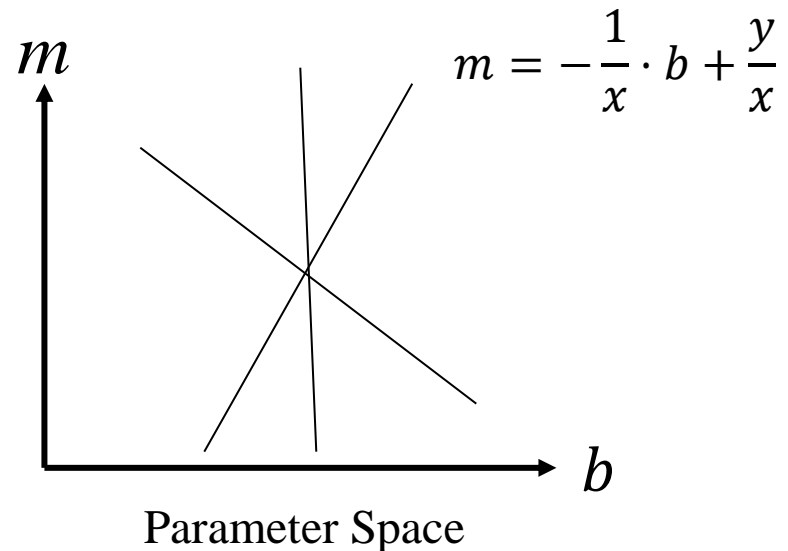
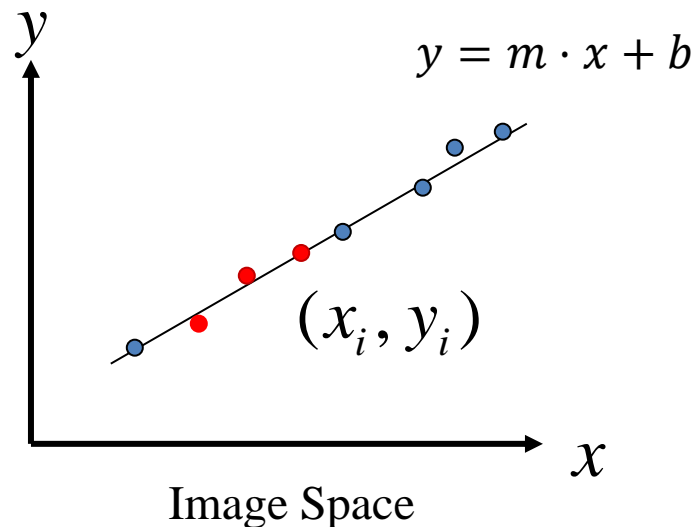
- In image space:

$$y = m \cdot x + b$$

- In parameter/Hough space:

$$m = -\frac{1}{x} \cdot b + \frac{y}{x}$$

$m$ : slope/gradient  
 $b$ : intercept/offset



# Line Equation in Cartesian Coordinate

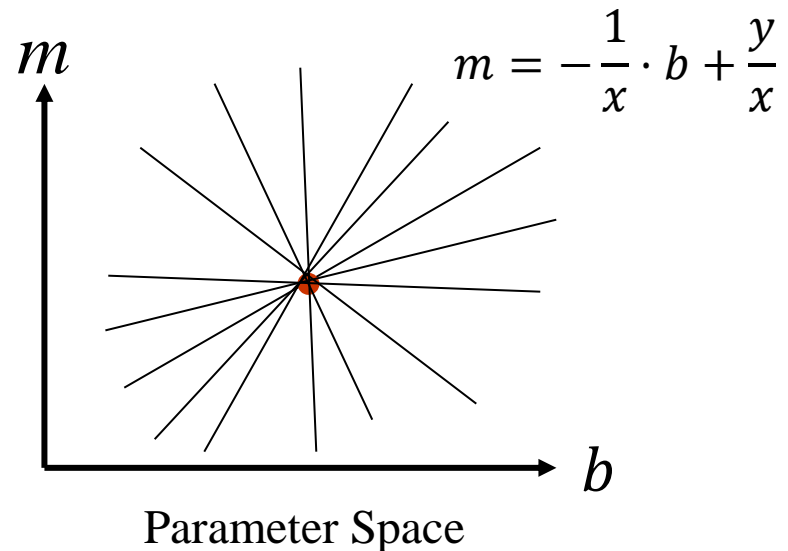
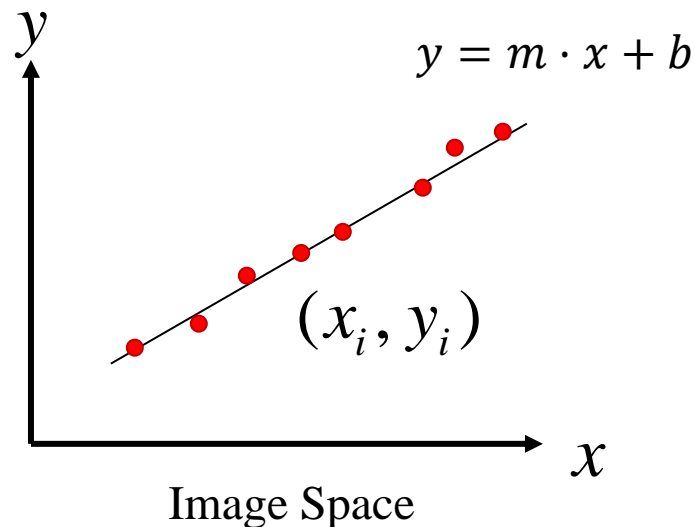
- In image space:

$$y = m \cdot x + b$$

- In parameter/Hough space:

$$m = -\frac{1}{x} \cdot b + \frac{y}{x}$$

$m$ : slope/gradient  
 $b$ : intercept/offset



# Line Equation in Cartesian Coordinate

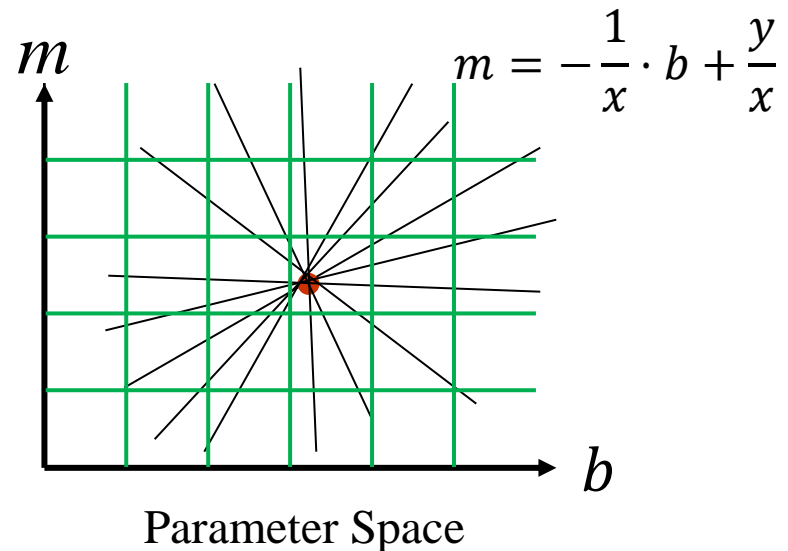
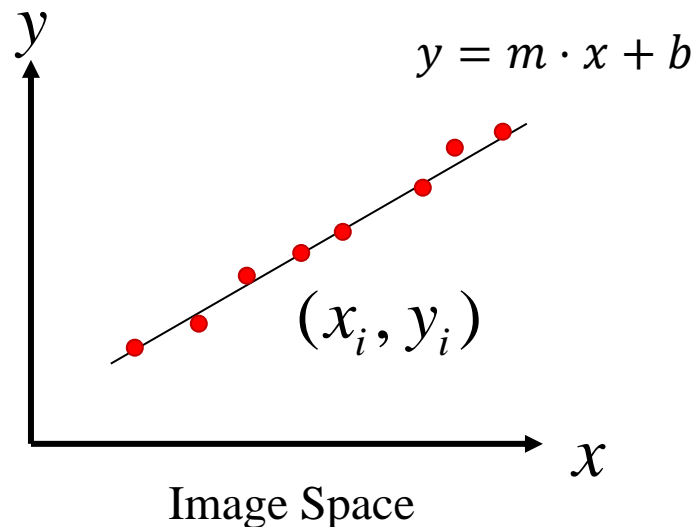
- In image space:

$$y = m \cdot x + b$$

- In parameter/Hough space:

$$m = -\frac{1}{x} \cdot b + \frac{y}{x}$$

$m$ : slope/gradient  
 $b$ : intercept/offset



# Hough Transform

---

- Algorithm:

```
quantize parameter space (m, b)
create a 2D accumulate matrix V

for each (x, y) in edge map:
    for each b:
        compute m = -1 / x * b + y / x
        add vote to V
    end
end

find the maximal votes in V
find the corresponding value of m and b
```



# Hough Transform

---

- In `hough_transform.m`

```
function [m, b] = hough_transform(edge_map)

%% find x, y position from edge map
[edge_y, edge_x] = find(edge_map);

%% range of b
H = size(edge_map, 1);
b_range = -H : 1 : H;

%% range of m
m_step = 0.01;
m_max = 5;
m_min = -m_max;
m_range = m_min : m_step : m_max;

%% create vote matrix
V = zeros(length(m_range), length(b_range));
```

# Hough Transform

---

- In `hough_transform.m`

```
%% create vote matrix
V = zeros(length(m_range), length(b_range));

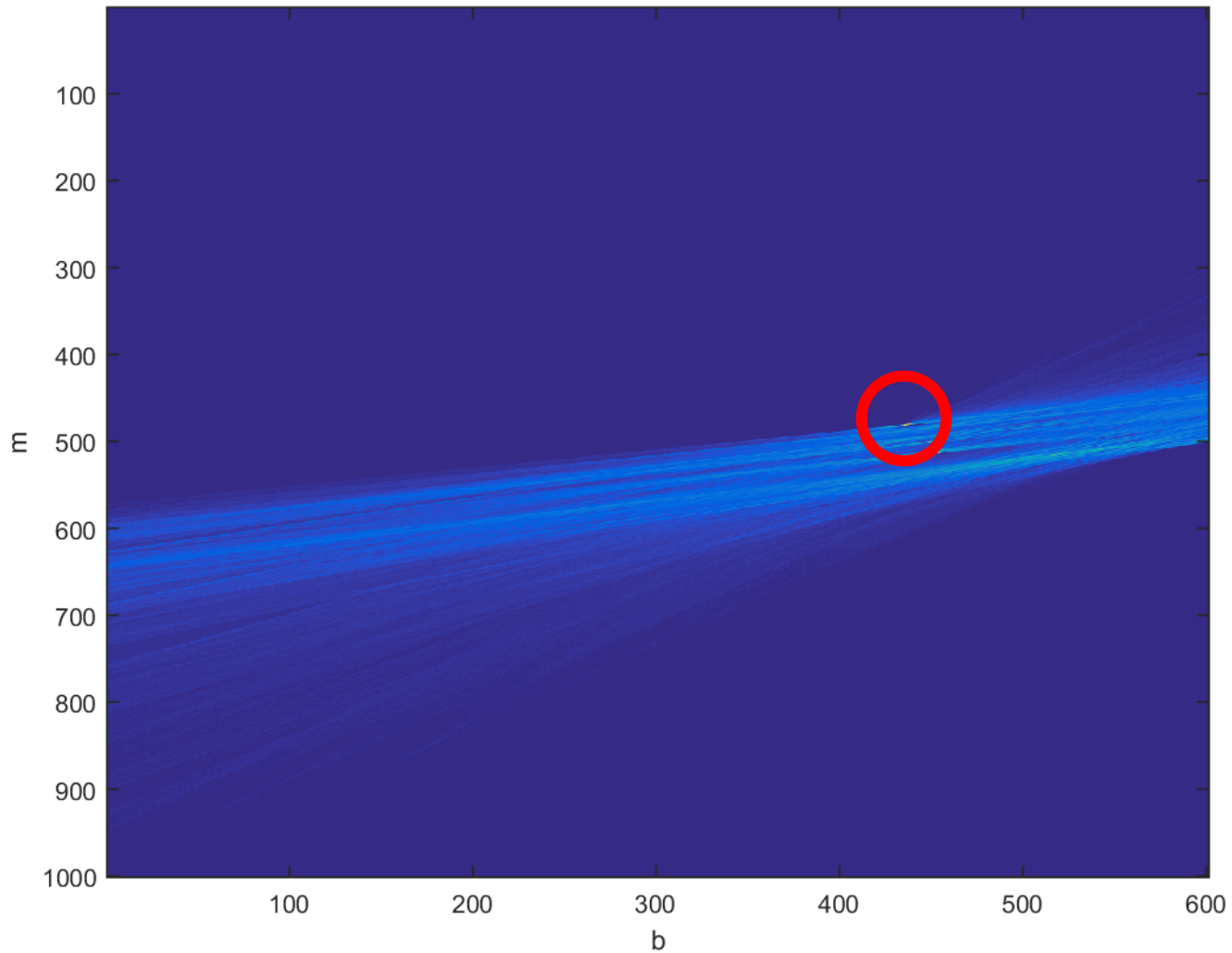
%% TODO: add votes

%% plot votes
figure, imagesc(V); xlabel('b'); ylabel('m');

%% find the maximal vote
max_vote = max(V(:));
[max_m_index, max_b_index] = find(V == max_vote);
m = m_range(max_m_index);
b = b_range(max_b_index);
```

# Visualization of vote matrix

---



# Detected Line

---

Your implementation (mb)



# Hints

---

- for each (x, y) in edge map:

```
for i = 1:length(edge_y)
    y = edge_y(i);
    x = edge_x(i);
```

- for each b:

```
for b_index = 1:length(b_range)
    b = b_range(b_index);
```

# Hints

---

- Before you add votes to V, check the range of m:

$$m_{\min} \leq m \leq m_{\max}$$

- You should convert m, b to their indexes in matrix V

```
m_index = round( (m - m_min) / m_step ) + 1;
```

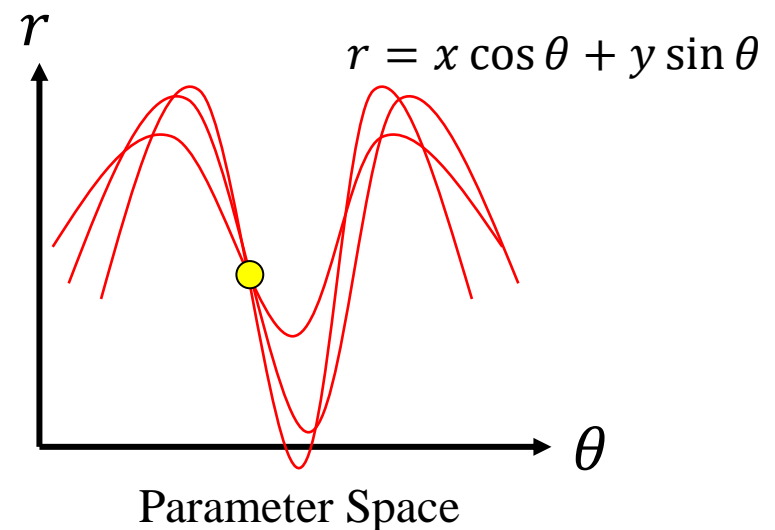
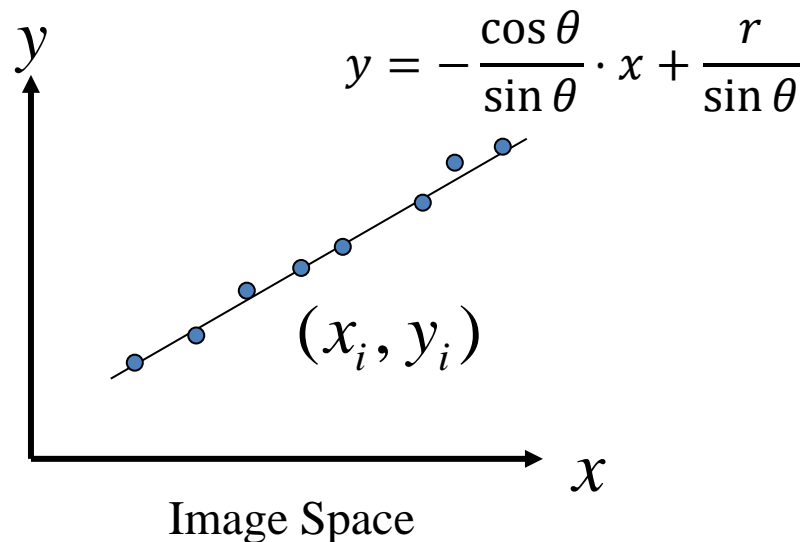
```
V(m_index, b_index) = V(m_index, b_index) + 1;
```

add 1 because  
index start from 1

# Line Equation in Polar Coordinate

- $y = m \cdot x + b$  cannot represent vertical lines ( $m \rightarrow \infty$ ):
- Use polar representation:

$$r = x \cos \theta + y \sin \theta$$



# Hough Transform

---

- Algorithm:

```
quantize parameter space (r, theta)
create a 2D accumulate matrix V

for each (x, y) in edge map:
    for each theta:
        compute  $r = x * \cos(\theta) + y * \sin(\theta)$ 
        add vote to V
    end
end

find the maximal votes in V
find the corresponding value of r and theta
```



# Hough Transform

---

- In `hough_transform_polar.m`

```
function [r, theta] = hough_transform_polar(edge_map)

%% find x, y position from edge map
[edge_y, edge_x] = find(edge_map);

%% range of r
H = size(edge_map, 1);
W = size(edge_map, 2);
r_max = round(sqrt(H^2 + W^2));
r_min = -r_max;
r_step = 1;
r_range = r_min : r_step : r_max;

%% range of theta
theta_step = 0.01;
theta_range = -pi/2 : theta_step : pi/2;
```

# Hough Transform

---

- In `hough_transform_polar.m`

```
%% create vote matrix
V = zeros(length(r_range), length(theta_range));

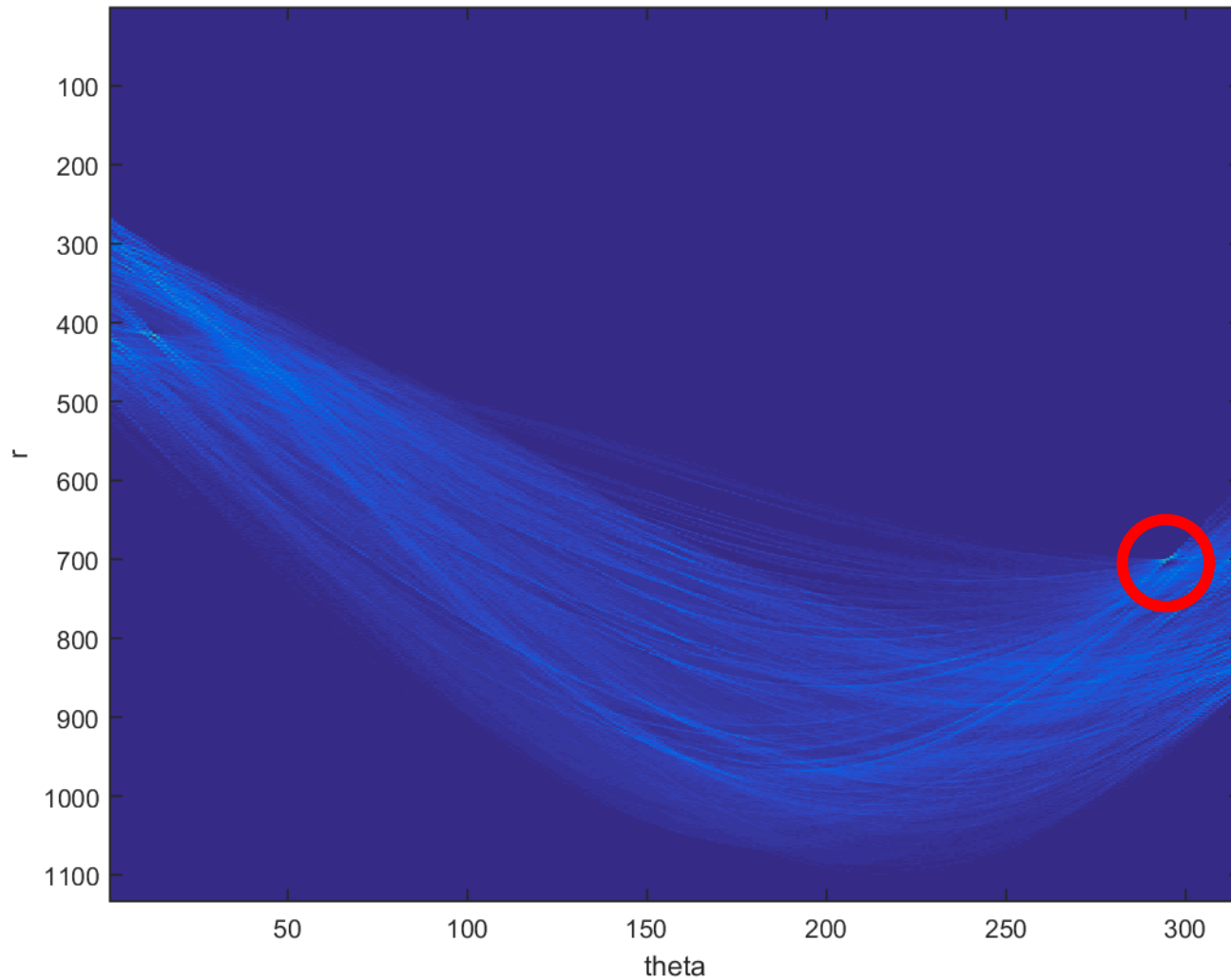
%% TODO: add votes

%% visualize votes
figure, imagesc(votes);
xlabel('theta'); ylabel('r');

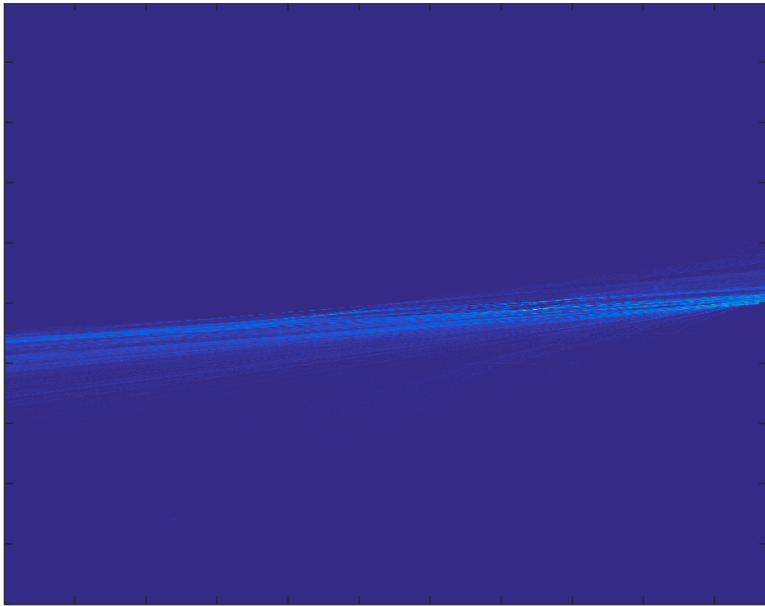
%% find the maximal vote
max_vote = max(votes(:));
[max_r_index, max_theta_index]
    = find( votes == max_vote );
r = r_range(max_r_index);
theta = theta_range(max_theta_index);
```

# Visualization of vote matrix

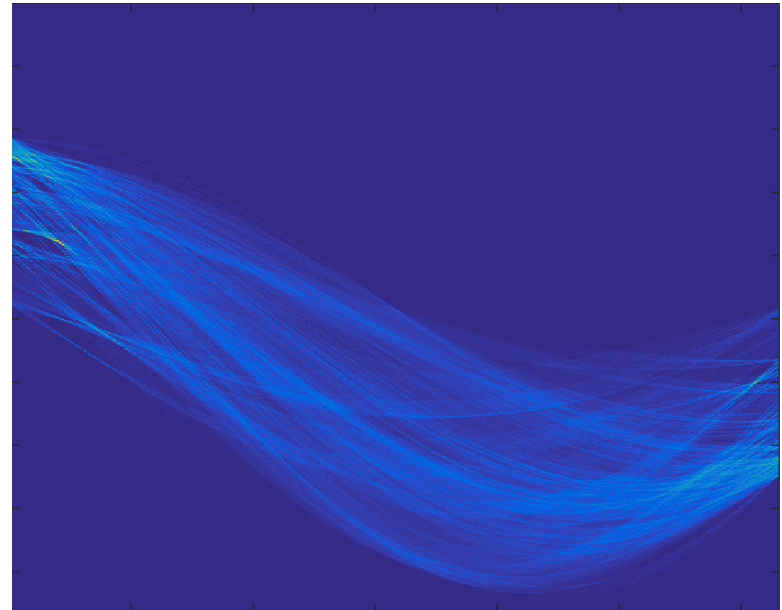
---



# Outputs



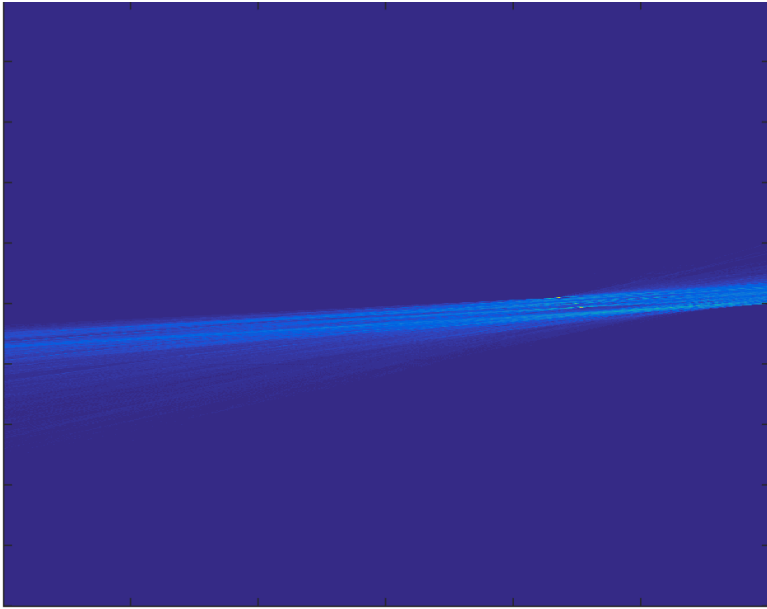
Cartesian space  $(m, b)$



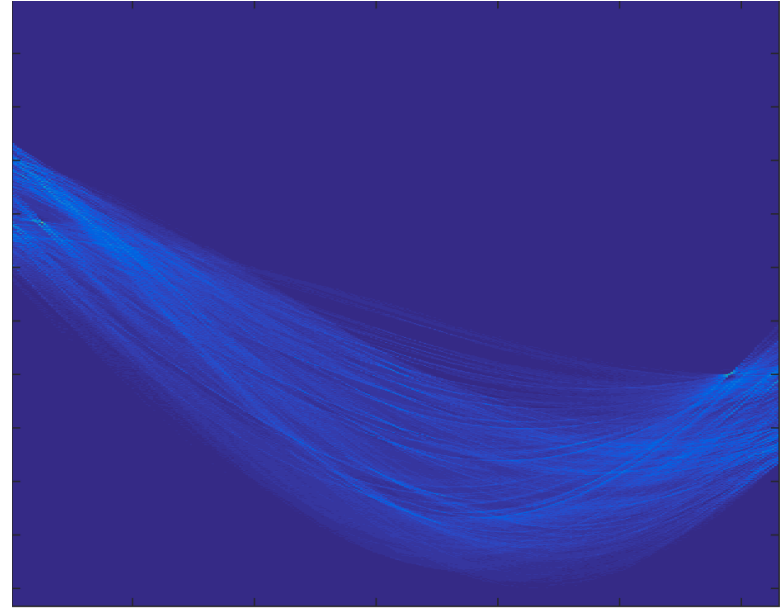
Polar space  $(r, \theta)$



# Outputs



Cartesian space  $(m, b)$



Polar space  $(r, \theta)$



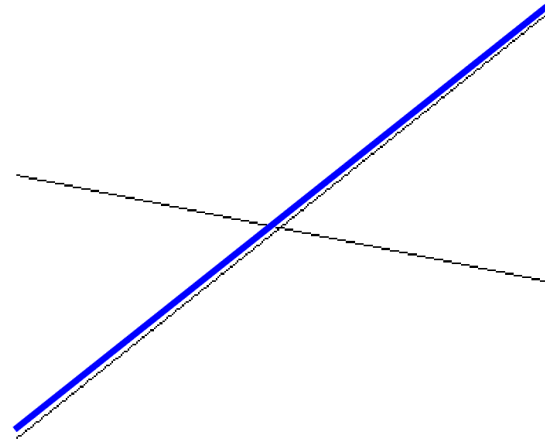
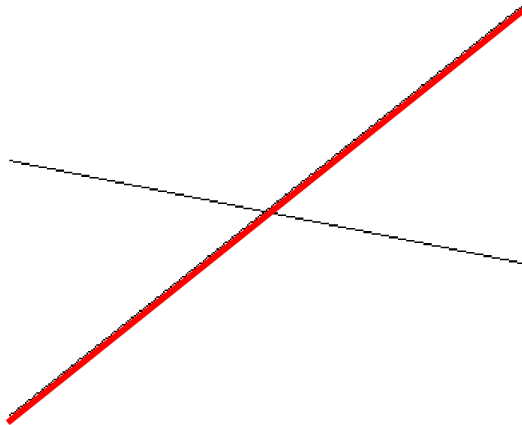
# Outputs



Cartesian space  $(m, b)$



Polar space  $(r, \theta)$



# Lab Assignment 08

---

- Implement `hough_transform.m` and `hough_transform_polar.m`
- Upload `lab08.m`, `hough_transform.m`, `hough_transform_polar.m` and `XXX_mb_line.png`, `XXX_polar_line.png` (XXX = lines, bridge, hill) separately