Bauhaus-Universität Weimar

Image Analysis and Object Recognition

Assignment 3

Hough line detection

SS 2017

(Course notes for internal use only!)

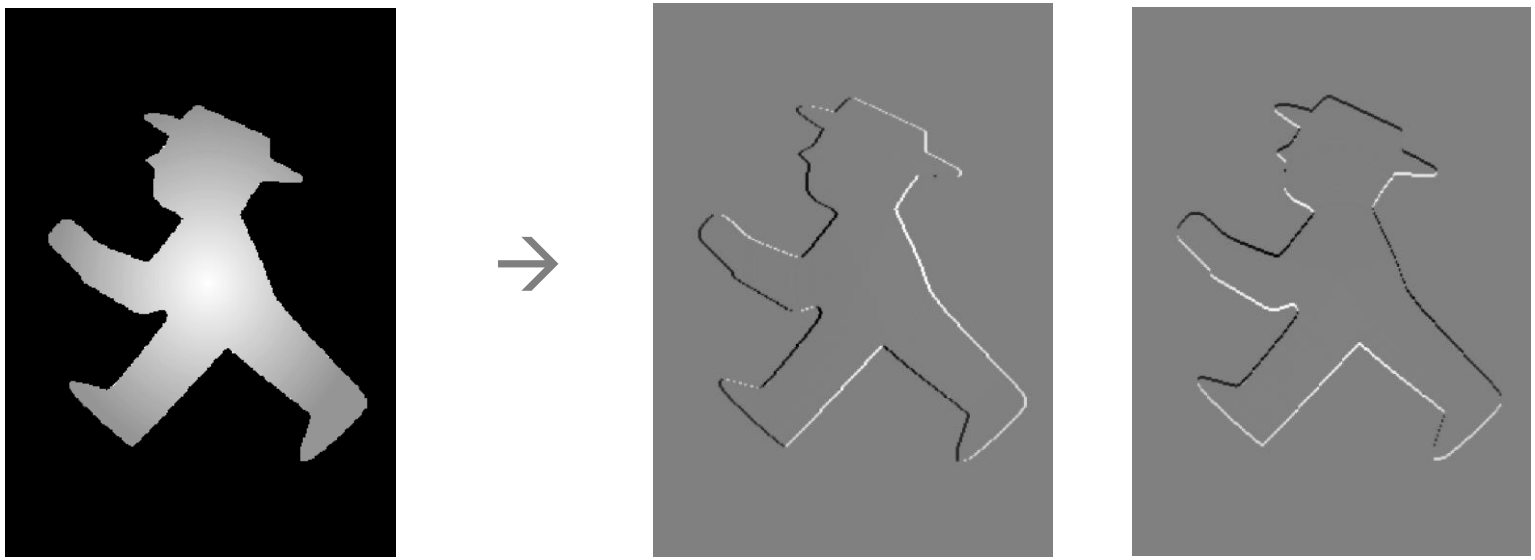
Assignment 2

A: GoG filtering

- Use data type double: value range?

→ value range [0,…,1]: *Image = mat2gray(mean(Image,3));*

→ value range [0,…,255]: *Image = mean(Image,3);*



- Plot GoG results: *figure, imshow(GoG_x, []);*

Assignment 2

A: GoG filtering

- Gradient magnitudes: same results for different value ranges

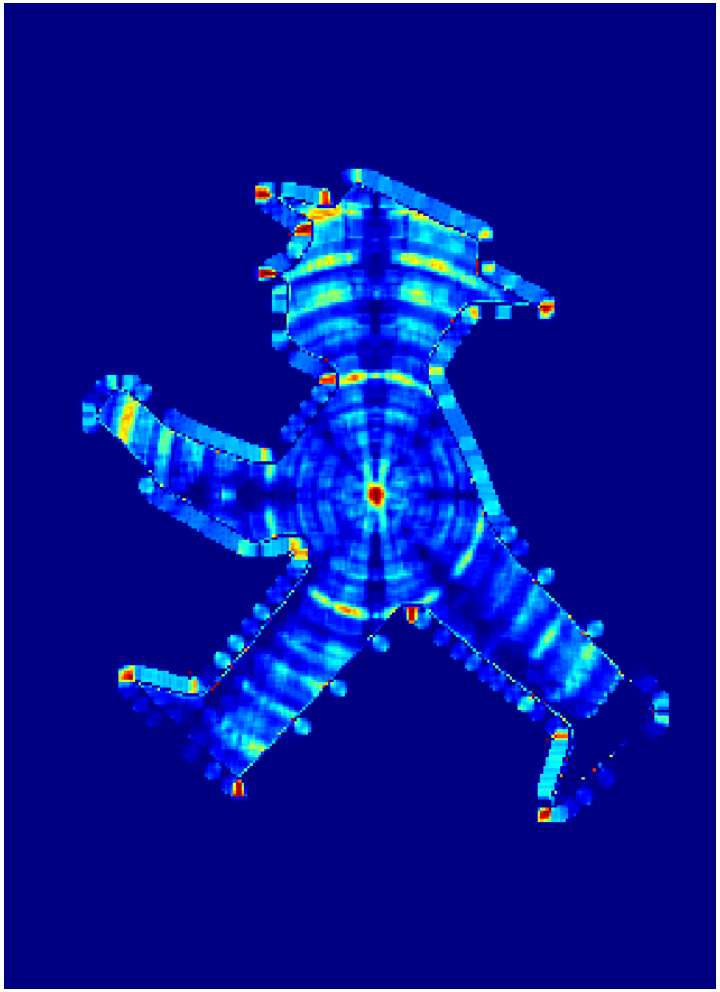
Image values between
0 and 255 (double)

Image values between
0.0 and 1.0 (double)

Assignment 2

B: Förstner interest points

- Roundness q
- Independent of input image value range
- $q = \frac{4 \cdot det(M)}{trace(M)^2}, \; 0 \leq q \leq 1$
- $q = NaN \rightarrow det(M) = trace(M) = 0$
- Threshold $t_q = 0.5$



Assignment 2

B: Förstner interest points

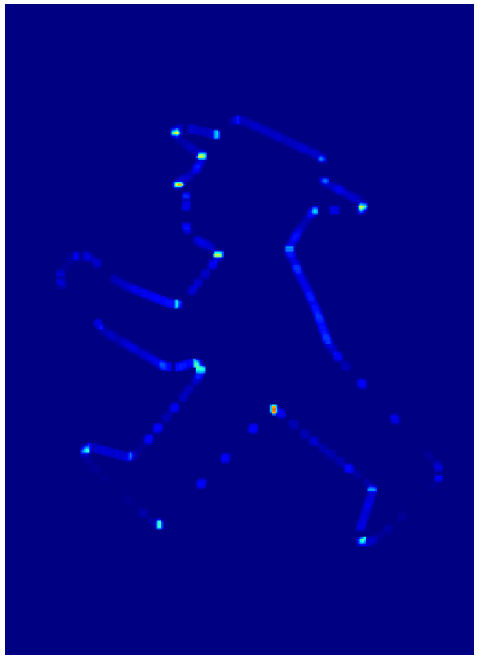
- Cornerness w

- Threshold **depends** on Image value range!
  - **Image:** $[0, \dots, 1]$:

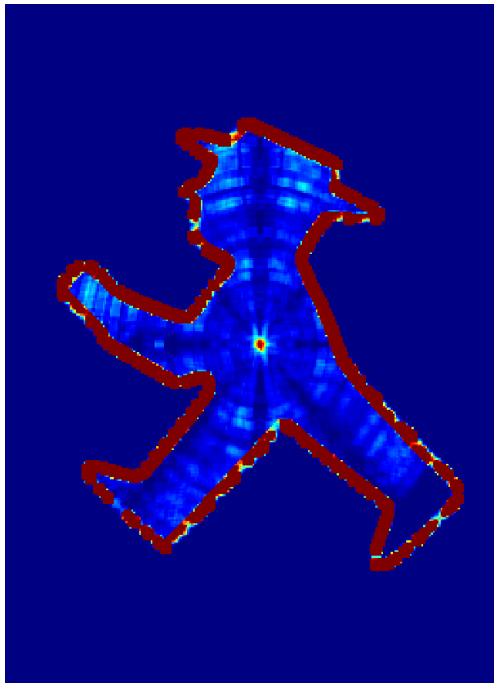
  → **Threshold** $t_w = 0.004$
  - **Image: double**

  → **Threshold** $t_w = 20.0$

- $w = \frac{trace(M)}{2} - \sqrt{\left(\frac{trace(M)}{2}\right)^2 - det(M)},$ $w > 0$



$[0, \dots, 1] \rightarrow \max = 0.77$



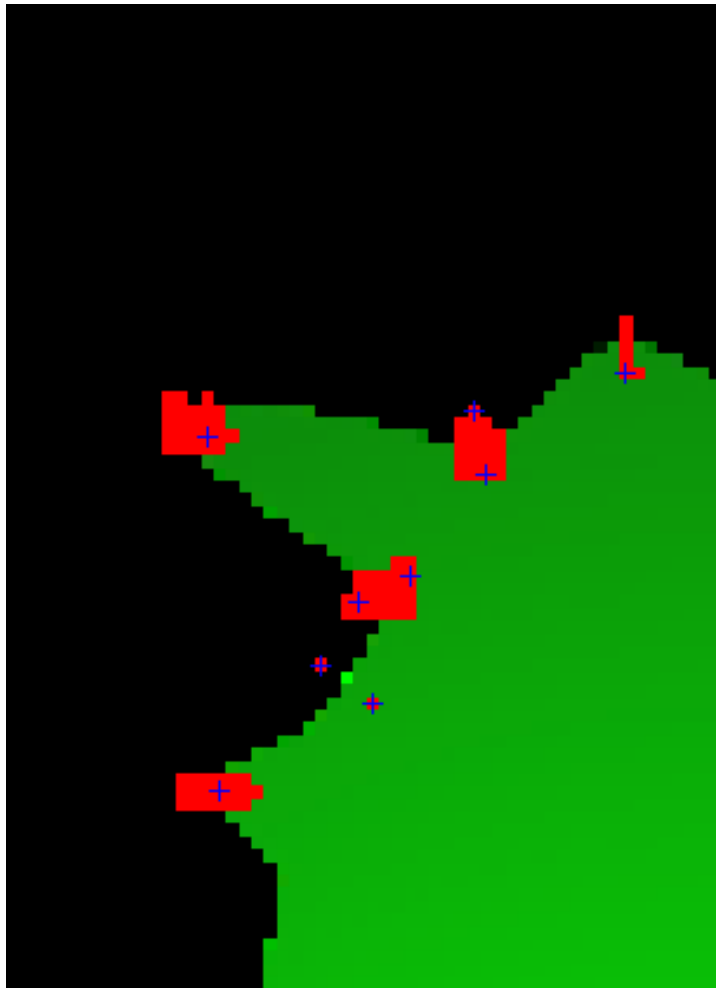
$double \rightarrow \max = 5749$

Assignment 2

B: Förstner interest points

- Find local maxima in w or q:
  - ***imregionalmax***

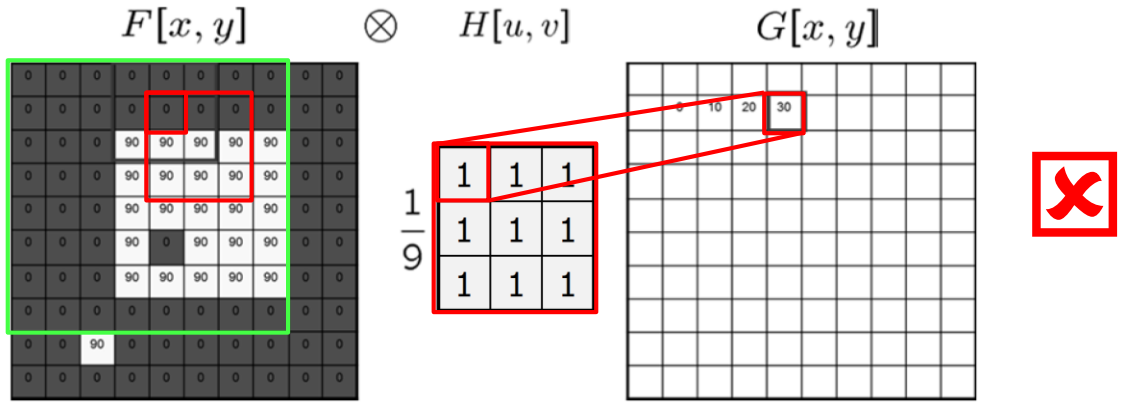
```
% identify local maxima
peaks = imregionalmax(W.*Q);
% get indices of maxima
[peaks_row, peaks_col] = find(peaks);
% plot them
hold on
plot(peaks_col, peaks_row, 'b+');
hold off
```



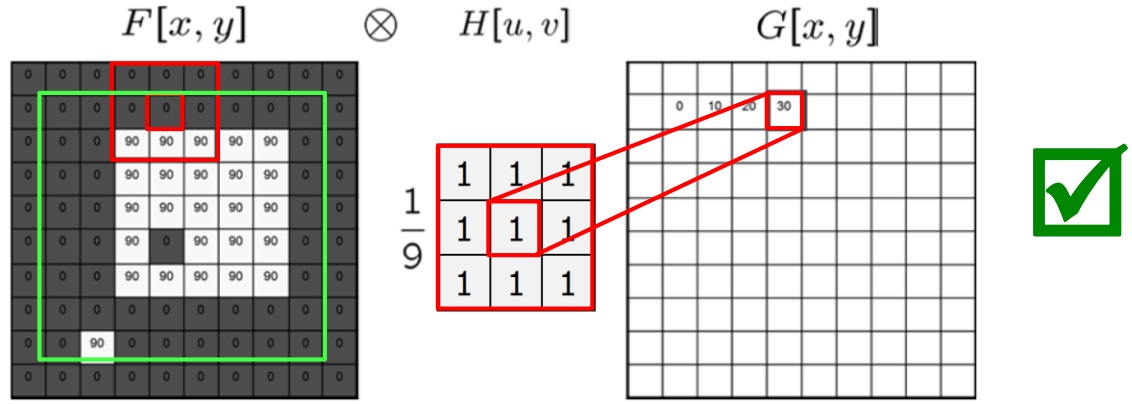
Assignment 2

Filtering / Convolution / Morphological operators

```
i_s % image size
k_s % kernel size 3
for x = 1 : i_s(1)-k_s+1
    for y = 1 :  i_s(2)-k_s+1
        f = F(x:x+k_s-1,y:y+k_s-1);
        G(x,y) = sum(sum(f.*H));
    end
end
```




```
%----------------------------------------
r % kernel radius r = 1
for x = 1+r : i_s(1)-r
    for y = 1+r : i_s(2)-r
        f = F(x-r:x+r,y-r:y+r);
        G(x,y) = …
    end
end
```



Assignment 3

- **Hough line detection**

- **Aims**
  - Understanding the concept of Hough-voting
  - Implement a voting algorithm

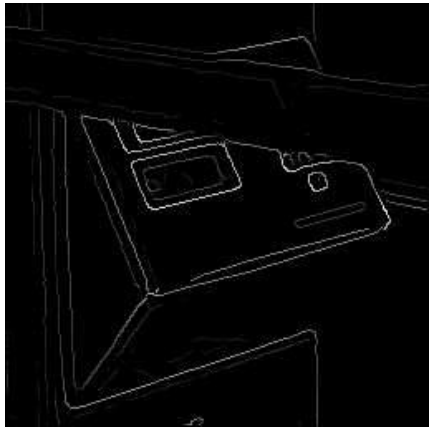
  → Detect and parameterize lines in images

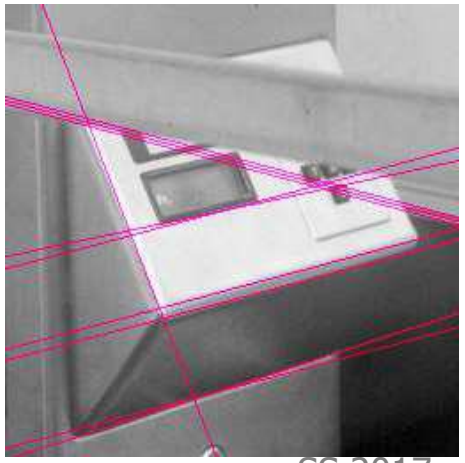
Assignment 3 - Outline

- Computation of Gradient images (GoG, A2)
- Apply threshold (A1) on gradient magnitudes (A2) → binary edge image

→ Already solved

- Use this binary edge image to compute Hough-voting table
  - Polar coordinates
  - Use edge directions
- Find local maxima in table (A2)
- Identify and plot the lines

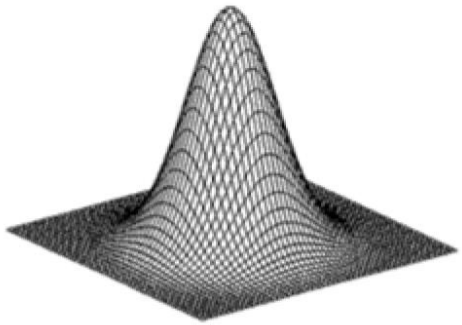


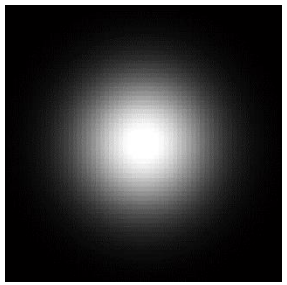


2D GoG filtering

- Gaussian filter

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

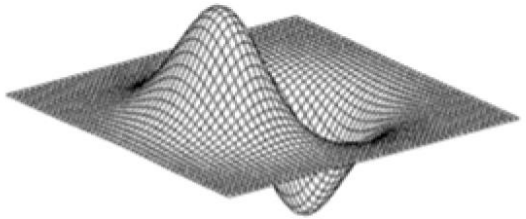


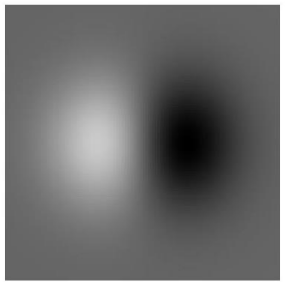


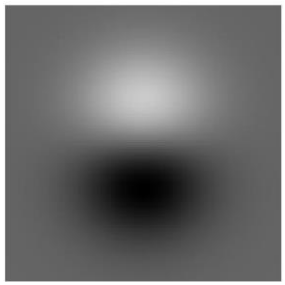
- Gradient of Gaussian

$$\frac{\partial G(x,y,\sigma)}{\partial x} = -\frac{x}{2\pi\sigma^4} exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

$$\frac{\partial G(x,y,\sigma)}{\partial y} = -\frac{y}{2\pi\sigma^4} exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$







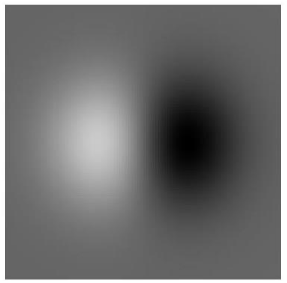
2D GoG filter computation

$$\frac{\partial G(x,y,\sigma)}{\partial x} = -\frac{x}{2\pi\sigma^4} exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

1) Define standard deviation, e.g. $\sigma = 0.5$

2) "Size" of filter kernel from center pixel: $r = |3 \cdot \sigma| = 2.0$

3)
$$c_x = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}; \quad c_y = c_x^T$$



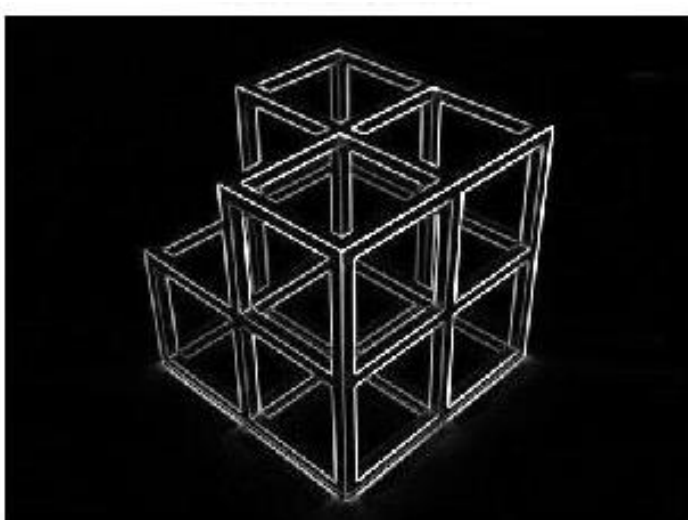
4) Compute filter using c_x and c_y for x and y

$$G_x = \frac{\partial G(x,y,\sigma)}{\partial x} = \begin{bmatrix} 0.0000 & 0.0001 & 0.0000 & -0.0001 & -0.0000 \\ 0.0002 & 0.0466 & 0.0000 & -0.0466 & -0.0002 \\ 0.0017 & 0.3446 & 0.0000 & -0.3446 & -0.0017 \\ 0.0002 & 0.0466 & 0.0000 & -0.0466 & -0.0002 \\ 0.0000 & 0.0001 & 0.0000 & -0.0001 & -0.0000 \end{bmatrix}; \quad G_y = \frac{\partial G(x,y,\sigma)}{\partial y} = \frac{\partial G(x,y,\sigma)}{\partial x}^T$$

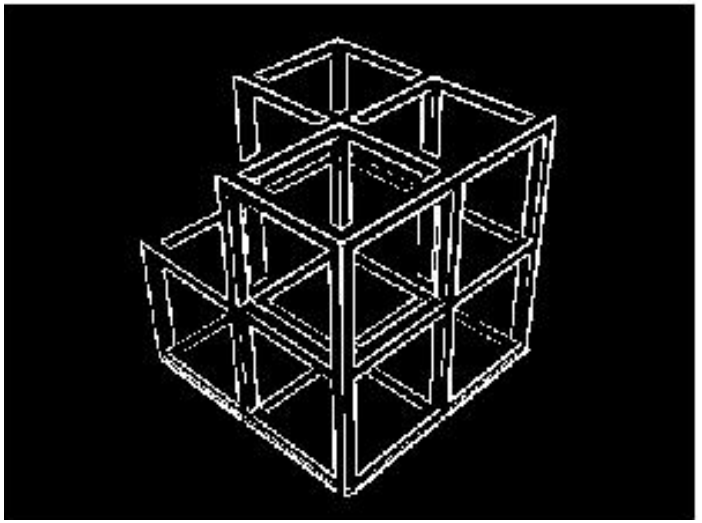
2D GoG filtering

$$G = \sqrt{(I_x)^2 + (I_y)^2}$$

Thresholding → Binary Image



→



Question: For human beings it is easy to see that the binary image contains several lines. But how can they be found and parameterized automatically?

Hough line detection

- Line equation: $y = xm + b$

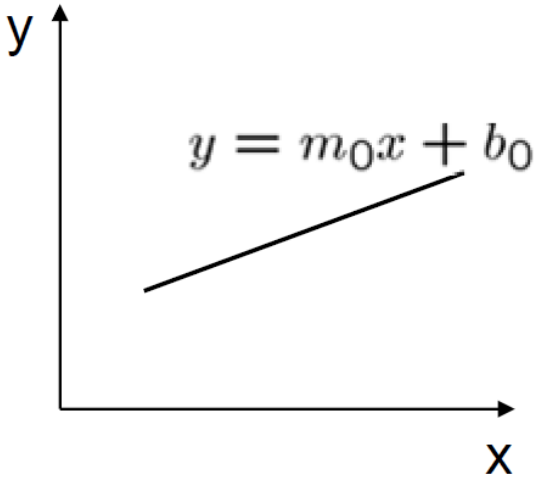
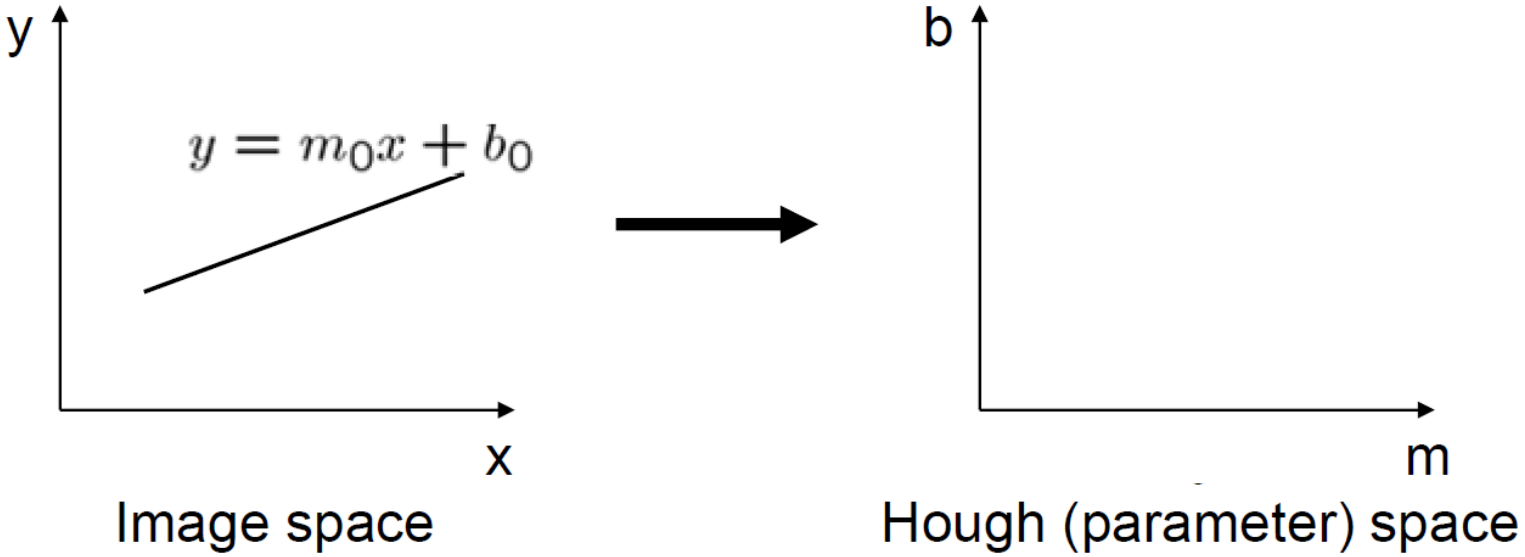


Image space

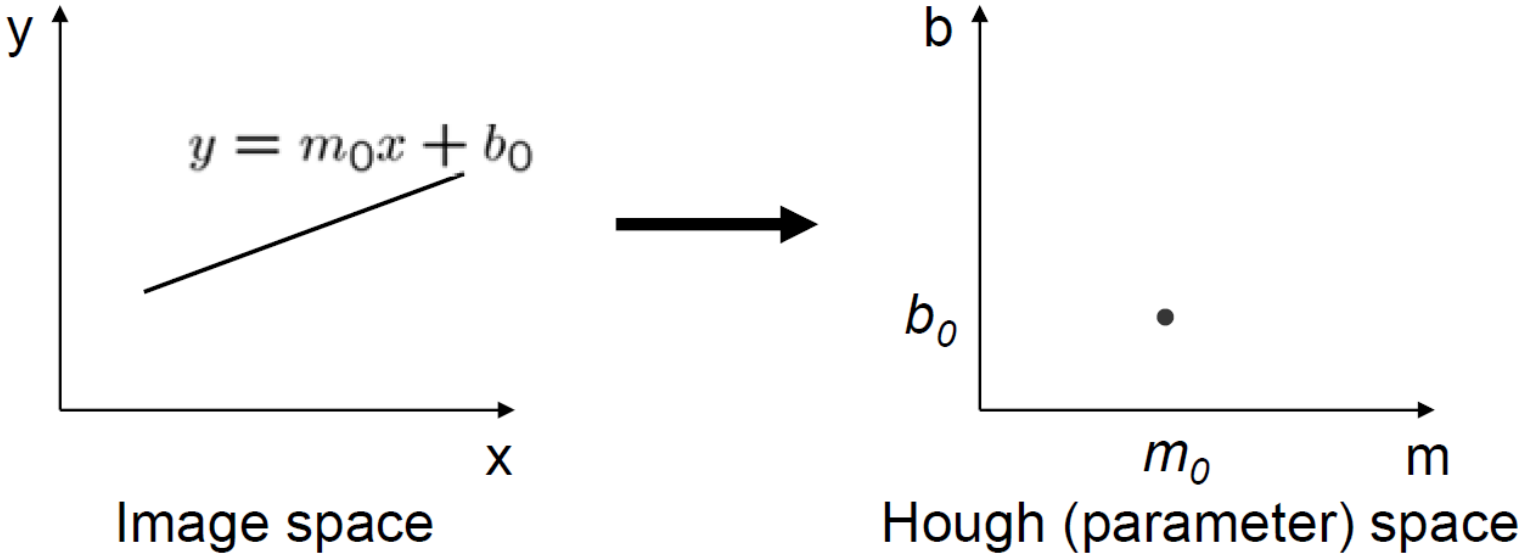
Hough line detection

- Line equation: $y = xm + b$



Hough line detection

- Line equation: $y = xm + b$

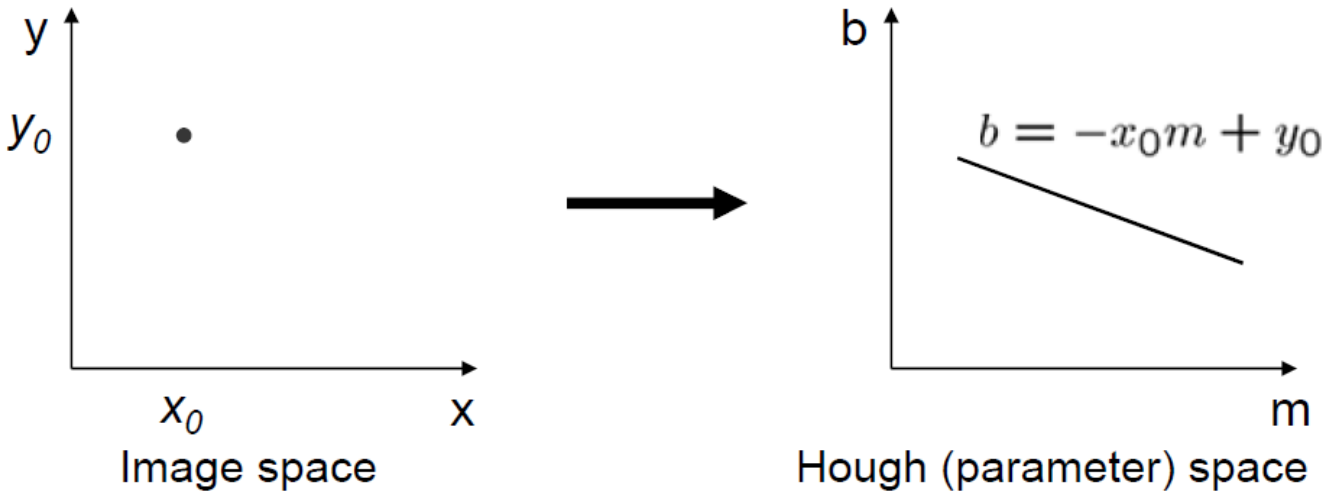


Hough line detection

- Problem: We don't know the line in the image yet

  → Check **all edge pixels** and let them vote!

- A point (x_0, y_0) in the image maps to a line in the hough space according to $b = -x_0 m + y_0$

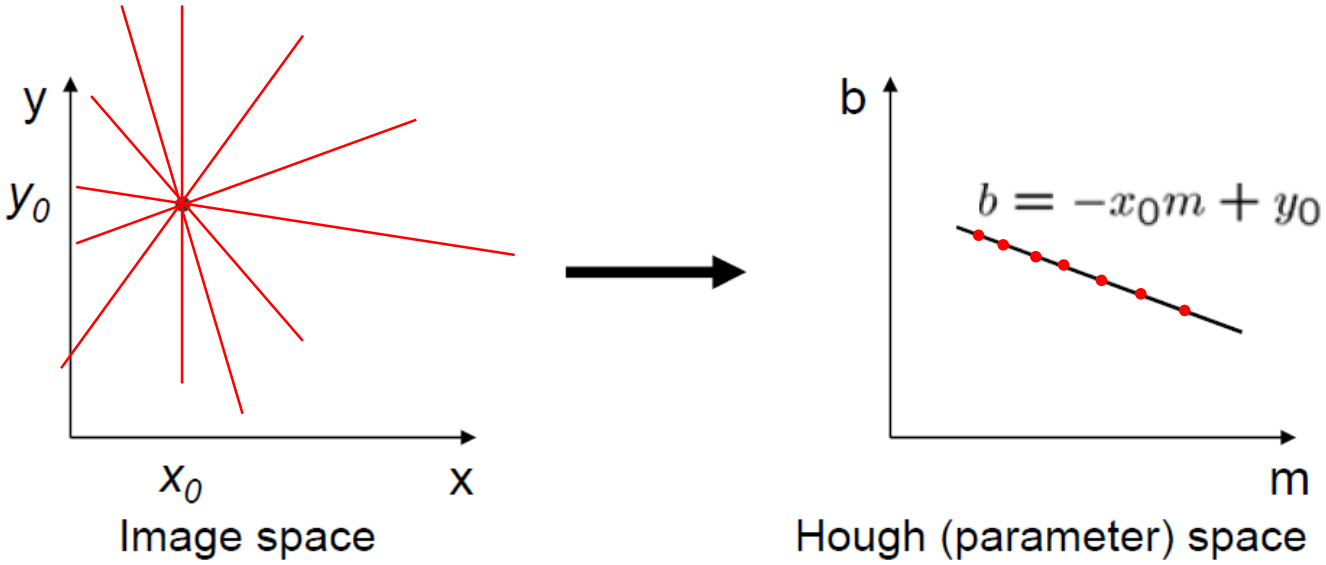


Hough line detection

- Problem: We don't know the line in the image yet

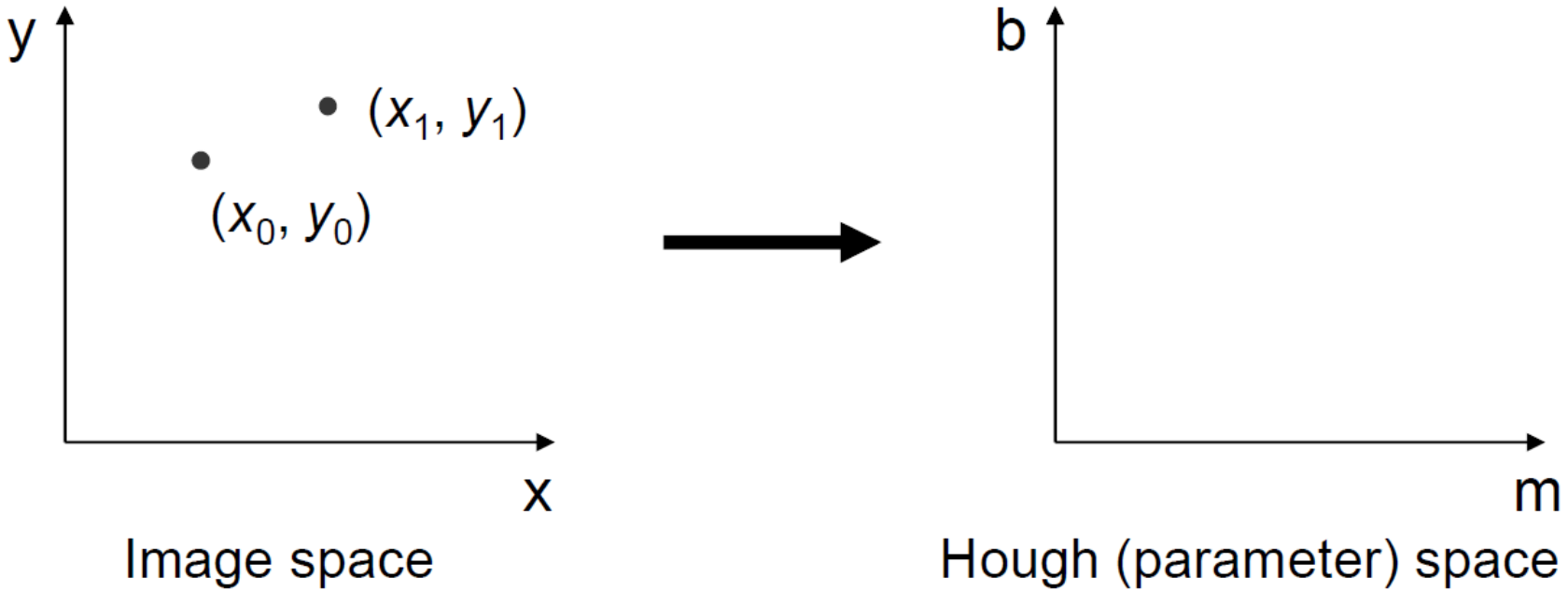
  → Check **all edge pixels** and let them vote!

- A point (x_0, y_0) in the image maps to a line in the hough space according to $b = -x_0 m + y_0$



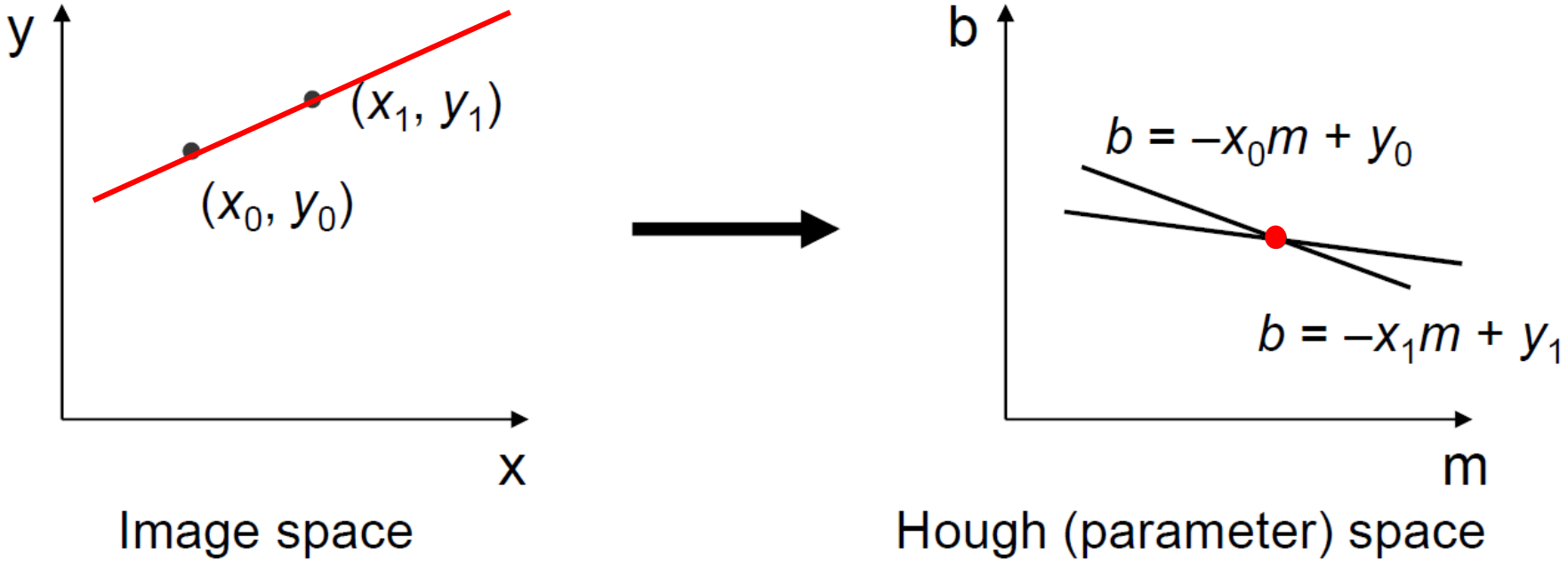
Hough line detection

- A line that contains two points (x_0, y_0) and (x_1, y_1)

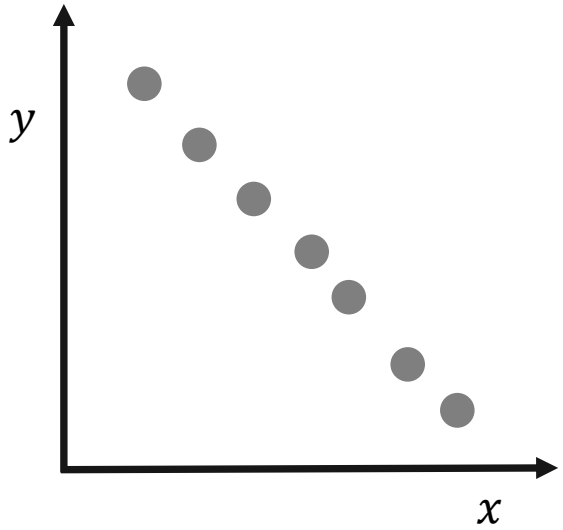


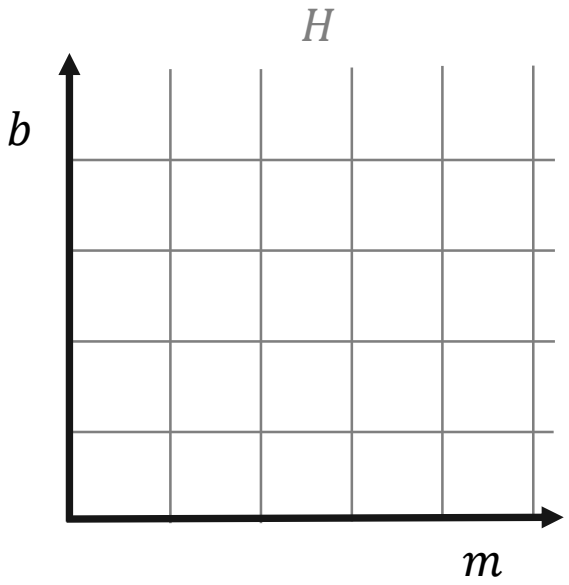
Hough line detection

- A line that contains two points (x_0, y_0) and (x_1, y_1)



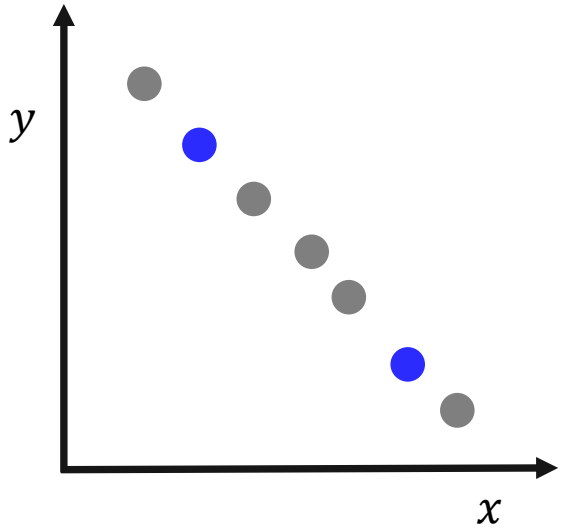
Voting Example

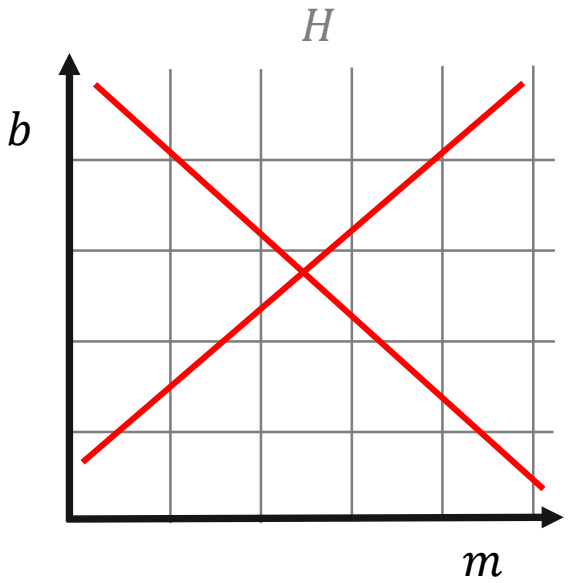




- Voting
  - Define an array with discrete values for m and b

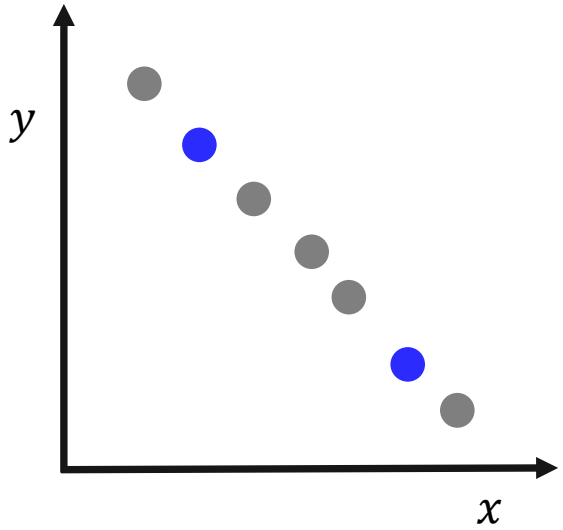
Voting Example

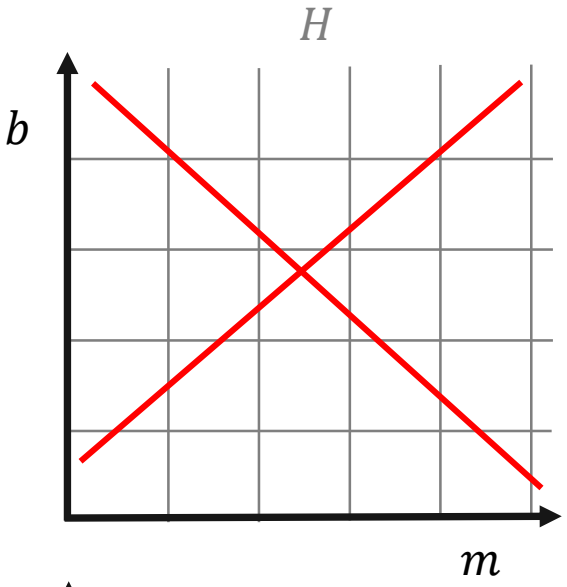


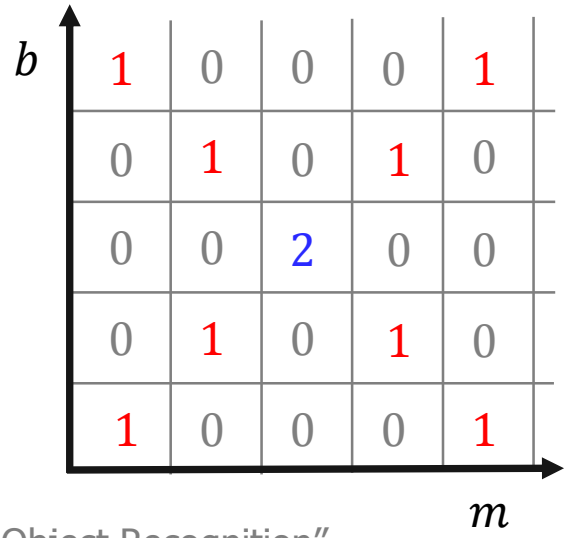


- Voting
  - Define an array H with discrete values for m and b
  - Find all value pairs (m, b) for a line using $b = -xm + y$
  - Increase each pixel (m, b) of voting array H by 1

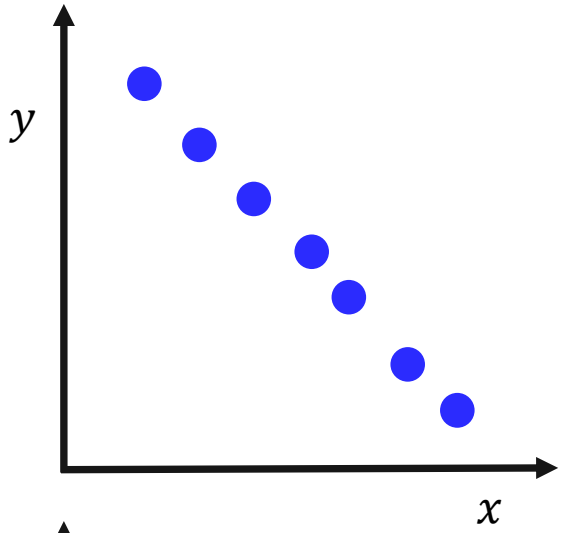
Voting Example

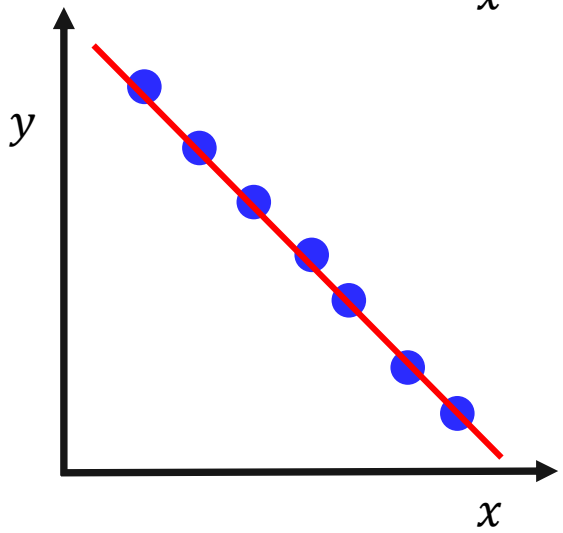


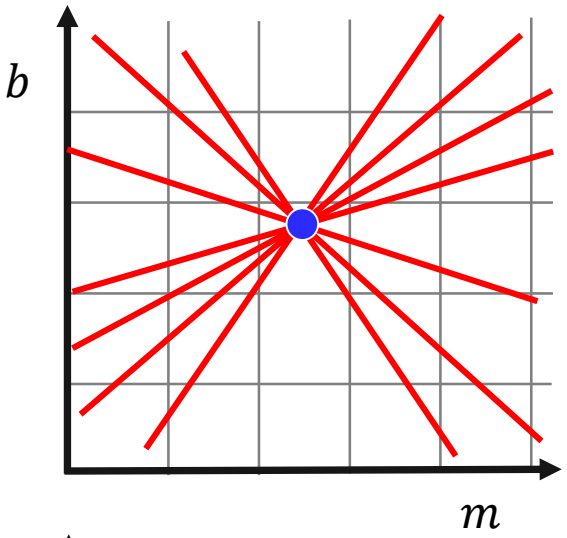


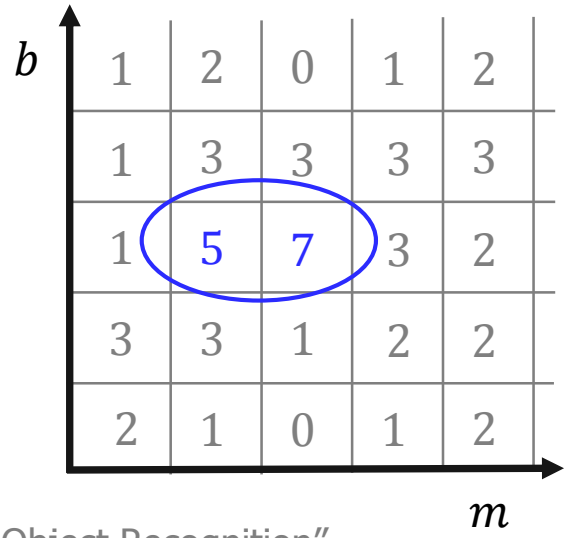


Voting Example



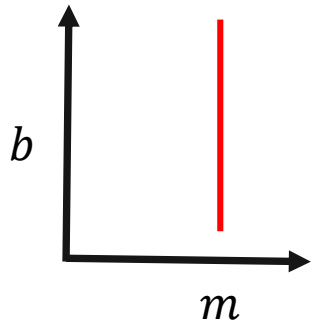






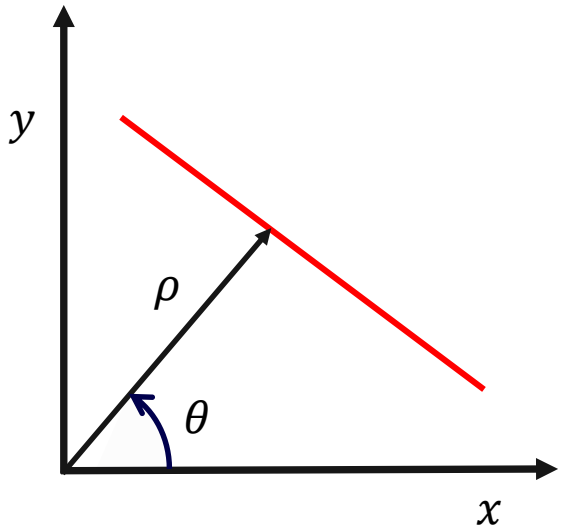
Polar Line Representation

- Problem with line equation $b = -x_0 m + y_0$
  → Unbounded parameter domain



- Alternative: polar representation of lines in image domain

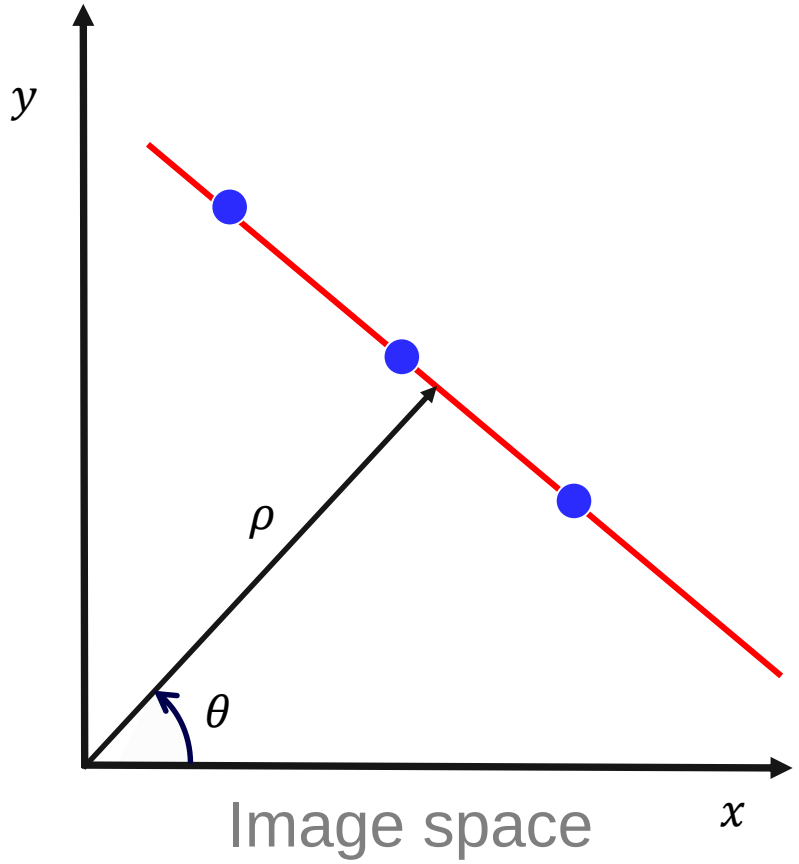
$$x \cdot cos\theta + y \cdot sin\theta = \rho$$

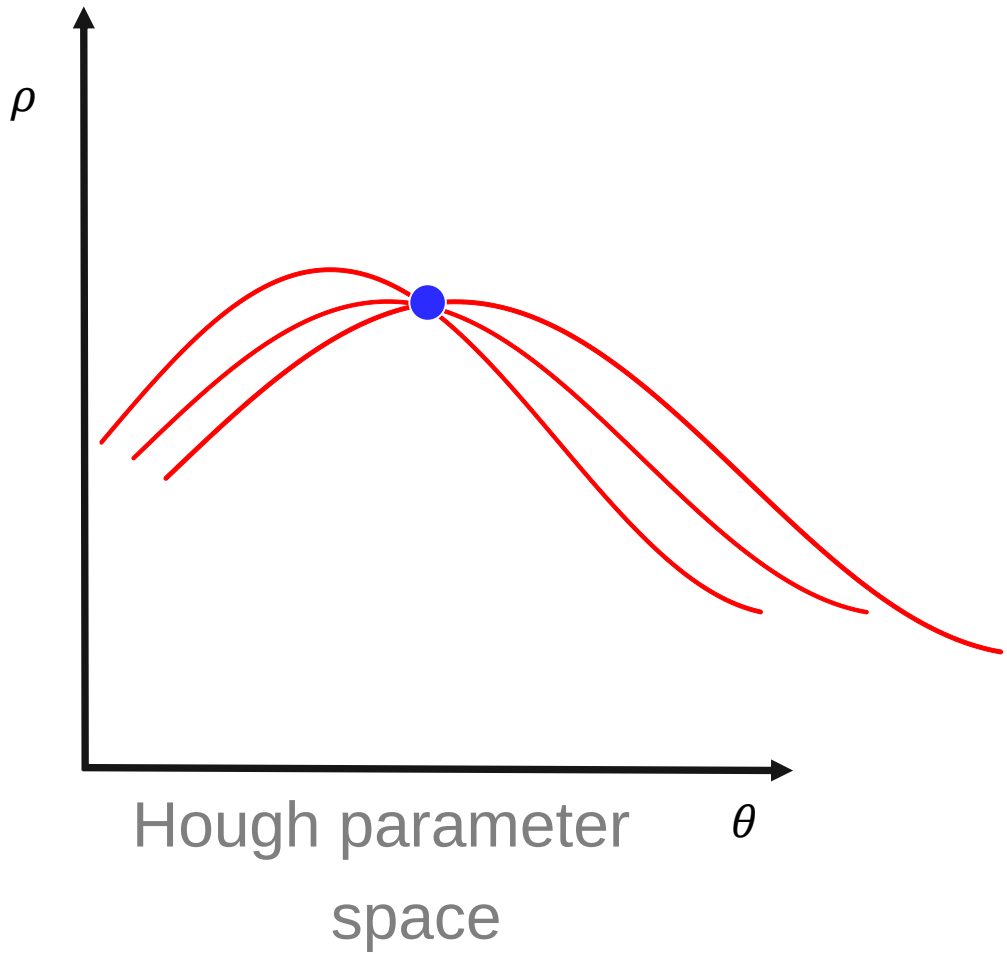


- ρ: shortest distance from origin to line
- θ: angle between x-axis and the perpendicular

Polar Line Representation

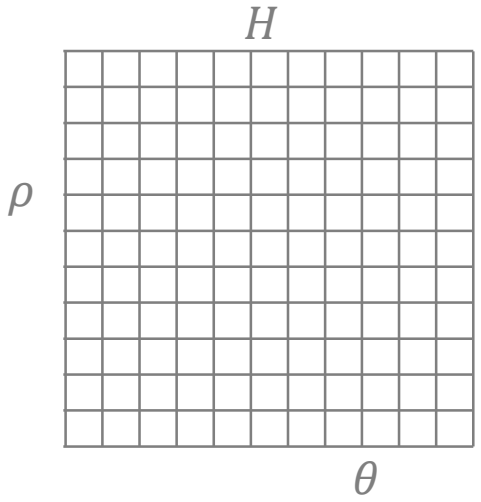
- Each point in image domain is a sinusoid in (θ, ρ)-space





Algorithm Outline

- **Input:** binary edge image (from GoG-filtering+gradient magn.+thresholding)
- **Initialize index vectors**
  - $\rho_{ind} = [-\rho_{max}, \dots, \rho_{max}]$, $\rho_{max} = \sqrt{n_{rows}^2 + n_{columns}^2}$
  - $\theta_{ind} = [-90, \dots, 89]$
- **Initialize** voting array H (integer)
  - $H = zeros(2 \cdot \rho_{max} + 1{,}180);$



- **for** each **edge point** (x, y) in the image

  for θ = -90 to 89

  $\rho = x \cdot cos\theta + y \cdot sin\theta$

  $H(\rho_i, \theta_i,) = H(\rho_i, \theta_i) + 1$

  end

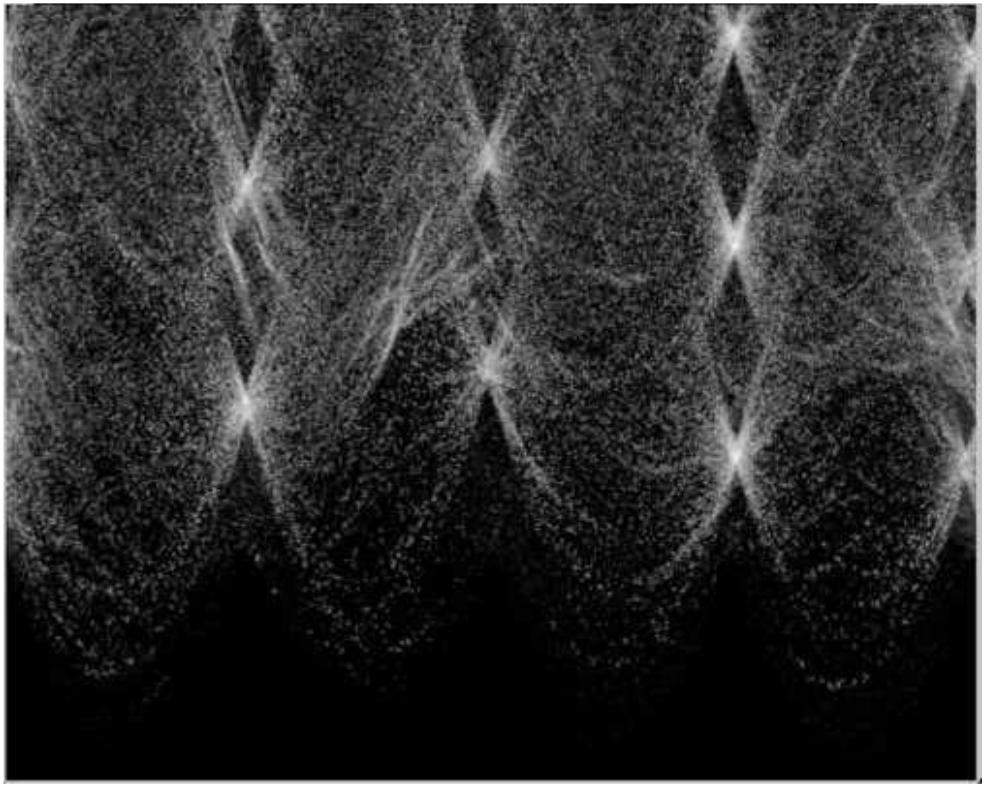
end

$\theta_i = find(\theta_{ind} == \theta)$

$\rho_i = find(\rho_{ind} == \rho)$

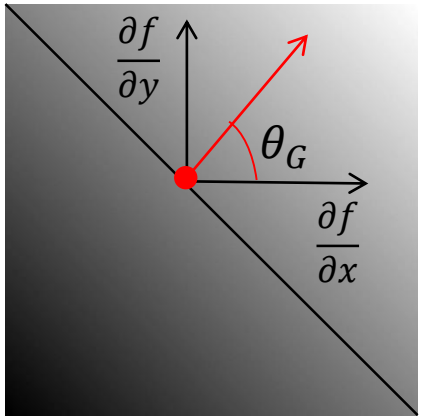
- Find the local maxima of H

Example



Algorithm extension

- Use the **gradient direction** of detected edges
  - GoG-filtering → first derivatives in x- and y-direction: $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$
  - Gradient direction: $\theta_G = tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$



- **Modified algorithm:**

for each edge point (x, y) in the image

θ = gradient orientation at (x, y)

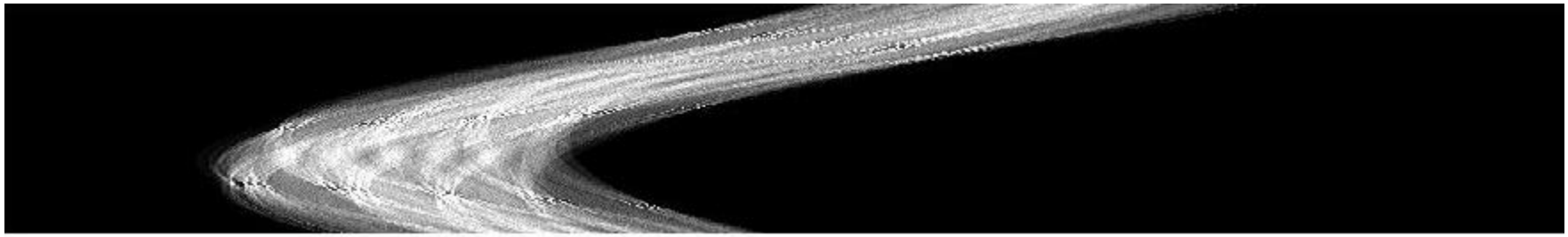
$\rho = x \cdot cos\theta + y \cdot sin\theta$

$H(\rho, \theta) = H(\rho, \theta) + 1$

end

Algorithm extension

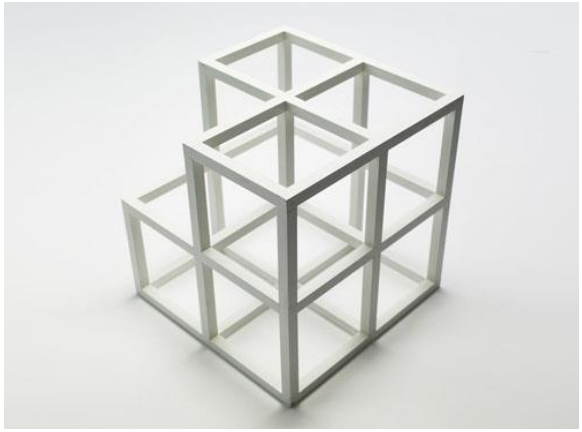
- Original Algorithm:



- Modified Version:

Tasks: Hough line detection 1/4

Input image:



Compute grayscale image (type double, [0,…,1]).

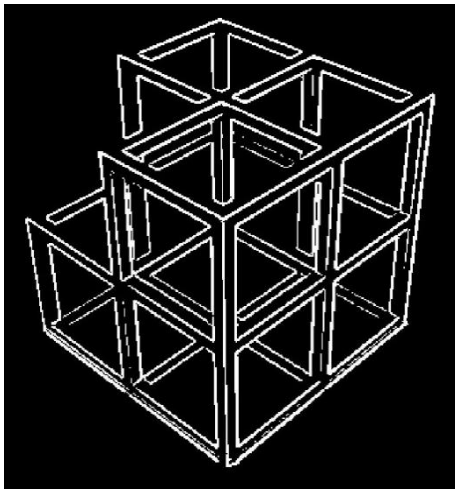
a. Read the input image and compute a grayscale image, if it has more than one channel. Plot the resulting image.

b. Apply a GoG-filtering (A2) to the grayscale image in order to derive a gradient image in x- and y-direction. **Don't use the function `conv2`!**

c. Compute the gradient magnitude image (A2)

$$G = \sqrt{(I_x)^2 + (I_y)^2}$$

Tasks: Hough line detection 2/4



d. Find and apply an appropriate threshold on the gradient magnitudes in order to derive representative edge pixels. Plot the binary edge image.

e. **Implement** a function for Hough line detection.

→**Inputs**: Binary edge image (d), GoG-filter outputs (b)

→**Outputs**: Hough voting array H, index vectors for the parameters θ and ρ (see matlab help: *hough* and algorithm outline)

→ Hints:

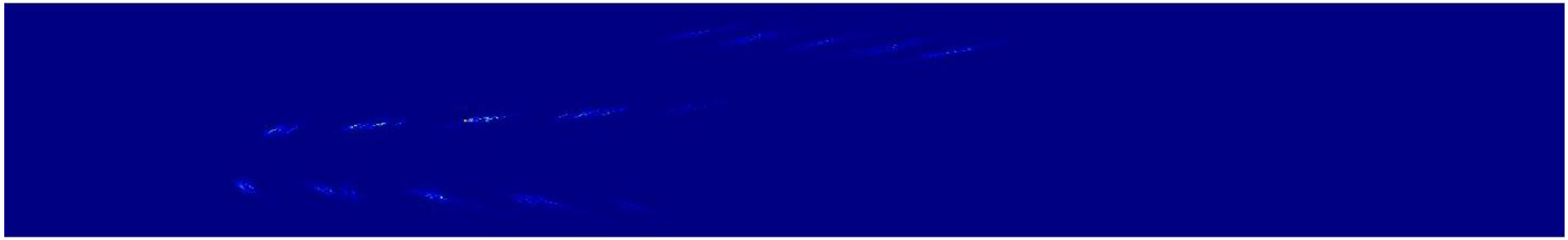
1. Use polar line representation
2. Incorporate available information about the gradient direction to speed up processing
3. **Do not use MATLAB function *hough*!**

Tasks: Hough line detection 3/4

f. Plot the resulting Hough voting array H

Either grayscale representation…

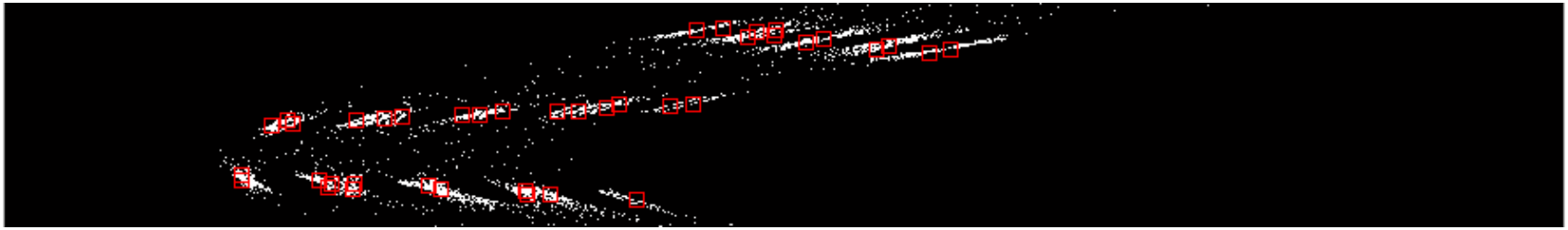
…or other colormaps



Hough line detection 4/4

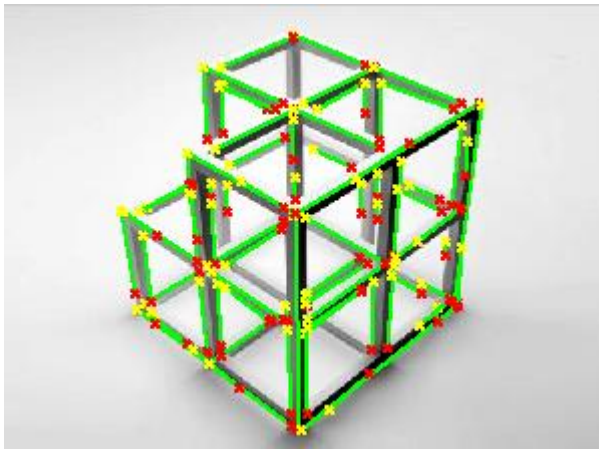
g. Use the MATLAB function *houghpeaks* to find local maxima of H

h. Plot the found extrema on your figure of step g



i. Use the MATLAB function *houghlines* to derive the corresponding lines

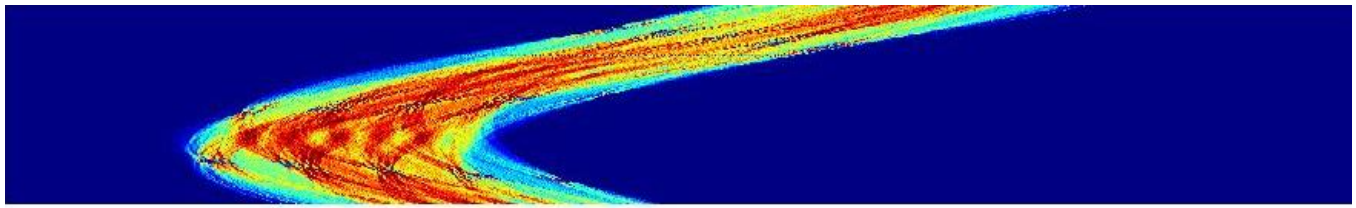
j. Plot them on the figure of step a.



Practical Hints

1) Initially you may use MATLAB function *hough* (see help)

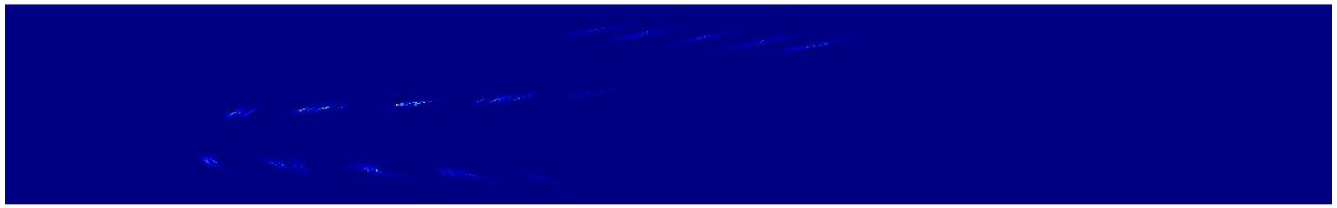
- *figure;*
- *imshow(imadjust(mat2gray(H)), 'XData', t, 'YData', r, 'InitialMagnification' , 'fit');*
- *colormap(jet);*



2) Compare it with **your** version

→Implement your function with same inputs and outputs as *hough*

→ Since we use the gradient directions, we will not obtain full sinusoids!



Practical Hints

- Use functions
  - `houghpeaks` *(g)*, `houghlines` *(i)*
  - *type conversion (double, [0,…,1]:* `mat2gray`
  - colormapping: `colormap('jet')`
- Don't use these functions in your final submission
  - `conv2`, `hough`, `imfilter`
- Be aware of different angle units
  - Conversion from $[0, \dots, 180]$ to $[0, \dots, \pi]$:

$$a_{[0,\dots,\pi]} = a_{[0,\dots,180]} \cdot \frac{\pi}{180}$$

Thank you!