Image Analysis and Object Recognition

Exercise 3

Summer Semester 2025

(Course materials for internal use only!)

Computer Vision in Engineering – Prof. Dr. Rodehorst
M.Sc. Mariya Kaisheva
mariya.kaisheva@uni-weimar.de

Agenda

	Topics:	Submission Dates:
Assignment 1.	Image enhancement, Binarization, Morphological operators	30.04.25
Assignment 2.	Gradient of Gaussian filtering, Förstner interest operator	21.05.25
Assignment 3.	**Shape detection based on Hough-voting**	**04.06.25**
Assignment 4.	Filtering in the frequency domain, Fourier descriptors	18.06.25
Assignment 5.	Image segmentation using clustering	02.07.25
Final Project.	- *Will be announced during the last exercise class* -	10.08.25

Assignment 2:
Sample Solution

Bauhaus-
Universität
Weimar

Assignment 2: Overview

Topics:

- Image filtering with Gradient of Gaussian (GoG)
- Interest points

Goal:

- Learn how to perform image filtering
- Practice reducing noise and **simultaneously** deriving image gradients (intensity changes)
- Practice identifying points of interest with the help of image gradients

Input:

- Provided image → *ampelmaennchen.png*
- Or a different image of your own choice

main function

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from scipy.ndimage import maximum_filter
from scipy.signal import convolve2d
```

convince functions for
2D convolution and **max filtering**

```python
def assignment2():
    sigma = 0.5   # standard deviation
    wmin = 0.004  # minimum cornerness
    qmin = 0.5    # minimum roundness

    I = np.array(Image.open('ampelmaennchen.png')).astype(float) / 255.0
    I_gray = np.mean(I, axis=2)
```

Task A

```python
    Ix, Iy = gradient(I_gray, sigma) # Compute gradient in x and y directions
    plt.figure(); plt.imshow(Ix, cmap='gray'); plt.title('convolution filtering')

    mag = np.sqrt(Ix**2 + Iy**2)      # Calculate gradient magnitude

    plt.figure(figsize=(15, 4))       # Create figure with subplots
    plt.subplot(1, 4, 1); plt.imshow(mag, cmap='gray'); plt.title('Gradient magnitude')
```

helper function

```
def gradient(I, sigma):
    r = int(round(3 * sigma))
    i = np.arange(-r, r + 1)

    # 1D Gaussian
    g = np.exp(-i**2 / (2 * sigma**2)) / (np.sqrt(2 * np.pi) * sigma)

    # Derivative of Gaussian
    d = -i * g / sigma**2

    # Apply separable convolution: GoG
    Ix = convolve2d(convolve2d(I, g[:,np.newaxis], mode='same', boundary="symm"),
                    d[np.newaxis], mode='same', boundary="symm")
    Iy = convolve2d(convolve2d(I, g[np.newaxis], mode='same', boundary="symm"),
                    d[:,np.newaxis], mode='same', boundary="symm")

    return Ix, Iy
```

explicit conversion of **1D arrays** g & d to a **row / column vector** using **`np.newaxis`**

the **mode** parameter of **`convolve2d()`** determines the size of the output, while **boundary** controls the type of padding

helper function

Alternative implementation using **1D convolution** and **explicit padding**

```
def gradient(I, sigma):
    # Define filter radius based on sigma
    r = round(3 * sigma)
    x = np.arange(-r, r + 1)

    # Create 1D Gaussian filter
    g = np.exp(-x**2 / (2 * sigma**2)) / (np.sqrt(2 * np.pi) * sigma)

    # Create 1D Gaussian derivative filter
    d = -x * g / sigma**2

    # Pad the image
    pad_width = r  # Use filter radius as padding width
    I_padded = np.pad(I, pad_width, mode='reflect')  # Other modes: 'constant', 'edge', 'symmetric'

    # Apply separable convolution to padded image
    temp_x = np.apply_along_axis(lambda x: np.convolve(x, g, mode='same'), 0, I_padded)
    temp_y = np.apply_along_axis(lambda x: np.convolve(x, g, mode='same'), 1, I_padded)

    Ix = np.apply_along_axis(lambda x: np.convolve(x, d, mode='same'), 1, temp_x)
    Iy = np.apply_along_axis(lambda x: np.convolve(x, d, mode='same'), 0, temp_y)

    # Crop to original size
    Ix = Ix[pad_width:-pad_width, pad_width:-pad_width]
    Iy = Iy[pad_width:-pad_width, pad_width:-pad_width]

    return Ix, Iy
```

`apply_along_axis()` applies a function to 1-D slices along the given axis main function

```python
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from scipy.ndimage import maximum_filter
from scipy.signal import convolve2d

def assignment2():
    sigma = 0.5   # standard deviation
    wmin = 0.004  # minimum cornerness
    qmin = 0.5    # minimum roundness

    I = np.array(Image.open('ampelmaennchen.png')).astype(float) / 255.0
    I_gray = np.mean(I, axis=2)
```

Task A

```python
    Ix, Iy = gradient(I_gray, sigma) # Compute gradient in x and y directions
    plt.figure(); plt.imshow(Ix, cmap='gray'); plt.title('convolution filtering')

    mag = np.sqrt(Ix**2 + Iy**2)      # Calculate gradient magnitude

    plt.figure(figsize=(15, 4))       # Create figure with subplots
    plt.subplot(1, 4, 1); plt.imshow(mag, cmap='gray'); plt.title('Gradient magnitude')
```

Task B

```python
    W, Q = foerstner(Ix, Iy)          # Compute Förstner cornerness and roundness

    plt.subplot(1, 4, 2); plt.imshow(W, cmap='gray'); plt.title('Cornerness')
    plt.subplot(1, 4, 3); plt.imshow(Q, cmap='gray'); plt.title('Roundness')


    W[Q <= qmin] = 0                  # Remove non-circular points
    R = find_max(W, wmin)             # Find interest points
    r, c = np.where(R)

    plt.subplot(1, 4, 4); plt.imshow(I); plt.plot(c, r, 'r+')
    plt.title('Förstner interest points')
```

```python
    plt.tight_layout()
    plt.show()
```

helper function

```
def foerstner(Ix, Iy):
    # Define accumulation kernel (5x5 box filter)
    g = np.ones((1,5))

    Ix2 = convolve2d(convolve2d(Ix**2, g, mode='same'), g.T, mode='same')
    Iy2 = convolve2d(convolve2d(Iy**2, g, mode='same'), g.T, mode='same')
    Ixy = convolve2d(convolve2d(Ix*Iy, g, mode='same'), g.T, mode='same')

    # Compute trace and determinant
    trace = Ix2 + Iy2
    det = Ix2 * Iy2 - Ixy**2

    # Avoid division by zero
    eps = np.finfo(float).eps

    # Compute cornerness and roundness
    W = trace/2 - np.sqrt((trace/2)**2 - det + eps)
    Q = 4 * det / (trace**2 + eps)

    return W, Q

def find_max(W, wmin):
    m = maximum_filter(W, size=3, mode='constant')
    R = (W == m) & (W > wmin)
    return R
```

prevention of numerical instabilities due to rounding effects

results in 0 padding at the image borders

Assignment 2 – **convolution** vs **correlation**

I – grayscale input image

$$G_x = \begin{bmatrix} 0.0000 & 0.0001 & 0.0 & -0.0001 & -0.0000 \\ 0.0002 & 0.0466 & 0.0 & -0.0466 & -0.0002 \\ 0.0017 & 0.3446 & 0.0 & -0.3446 & -0.0017 \\ 0.0002 & 0.0466 & 0.0 & -0.0466 & -0.0002 \\ 0.0000 & 0.0001 & 0.0 & -0.0001 & -0.0000 \end{bmatrix};$$

$\otimes$ Cross-Correlation

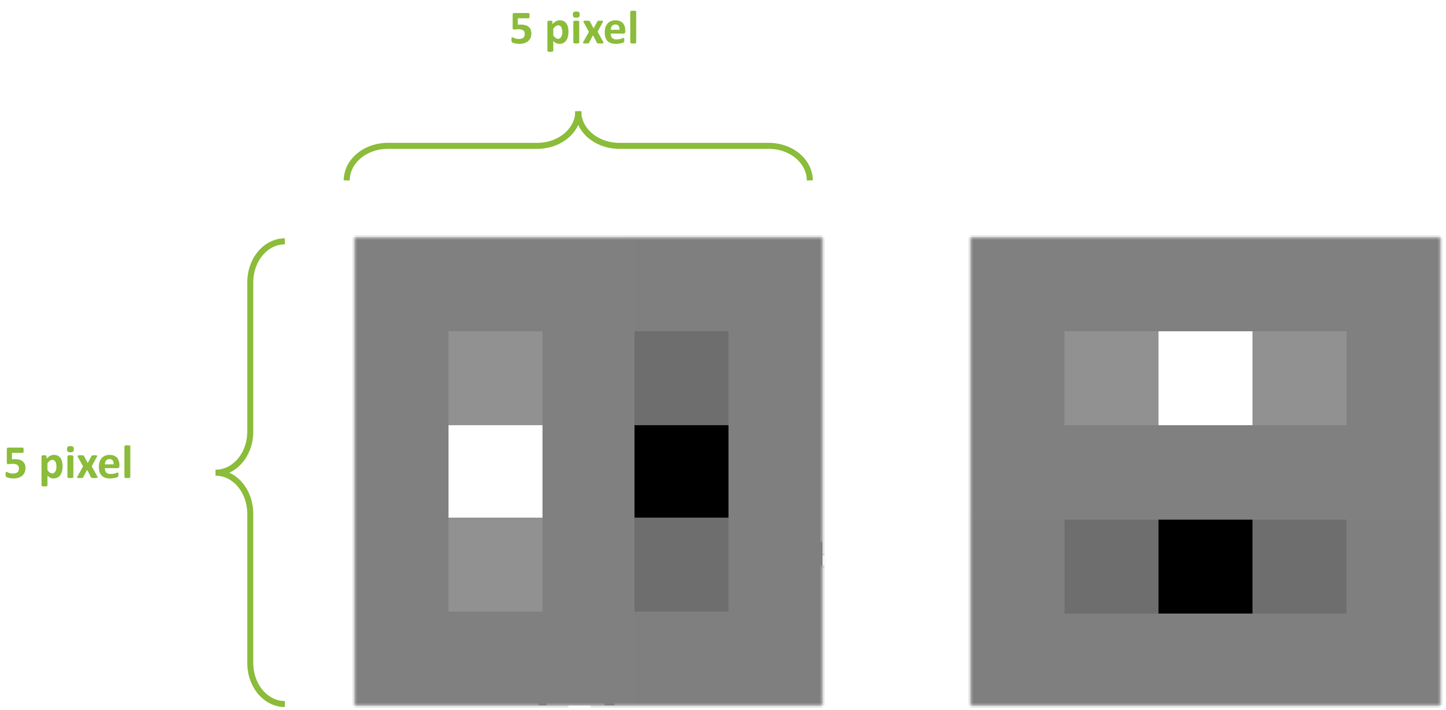
$\star$ Convolution

Horizontal gradient images

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \star \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

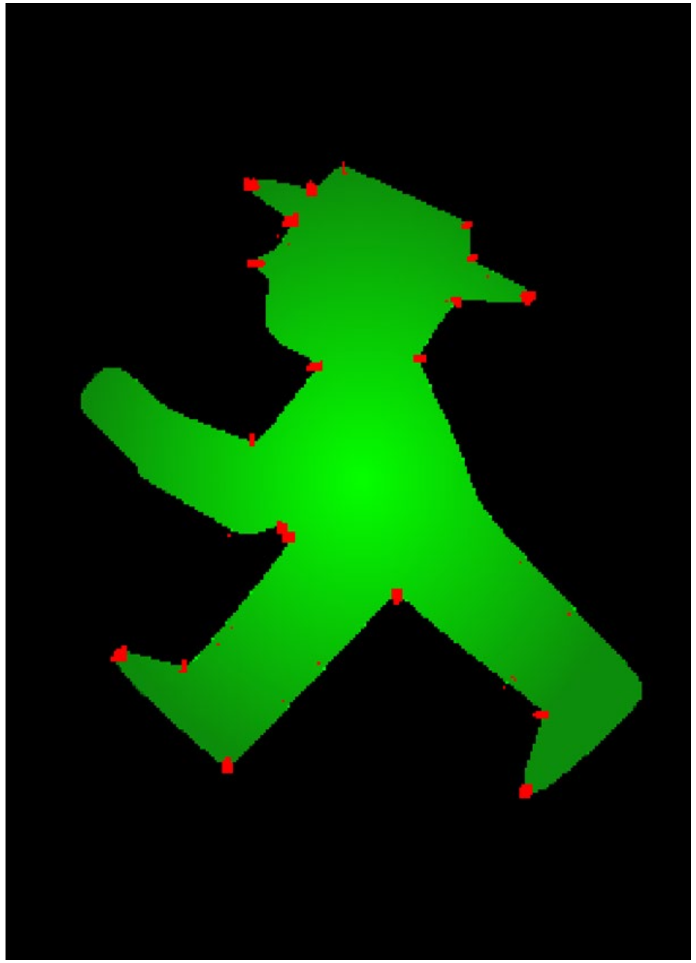
Assignment 2 – convolution kernel visualization



horizontal kernel

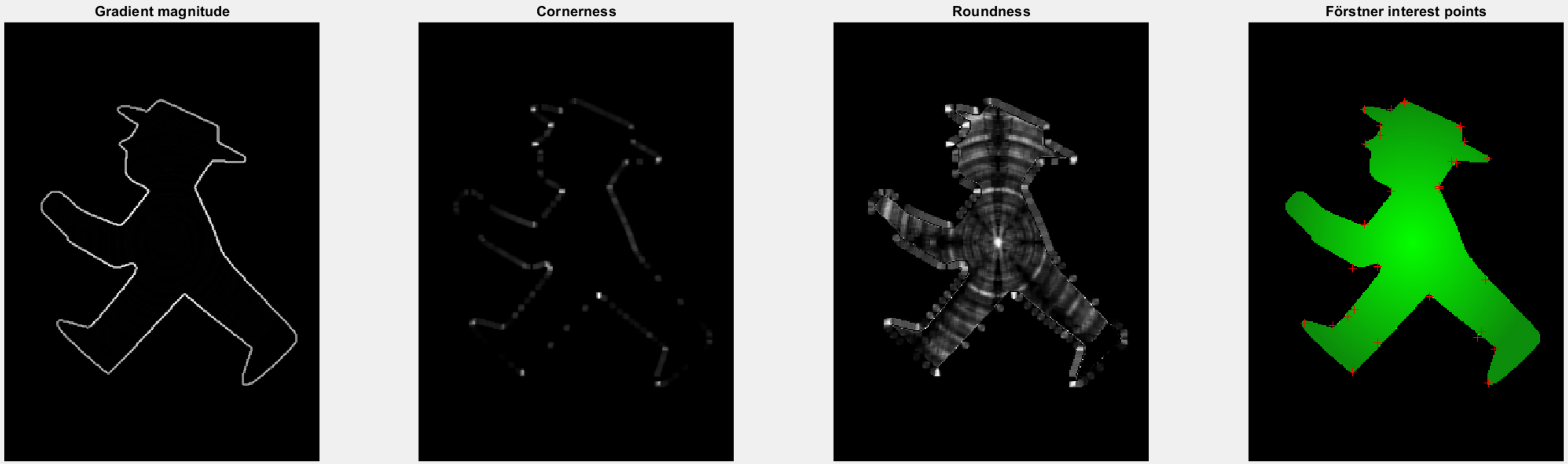
vertical kernel

Assignment 2 – sample results



Choose the
max response
for each 3-by-3
neighbourhood

Assignment 2 – sample results



Assignment 3

Assignment 3: Overview

Topics:

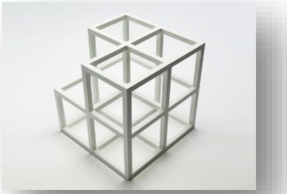
- Hough line detection

Goal:

- Understanding the concept of Hough-voting
- Practice detection and parameterization lines in images

Input:

- Provided image → *input_ex3.jpg*
- Or a different image of your own choice



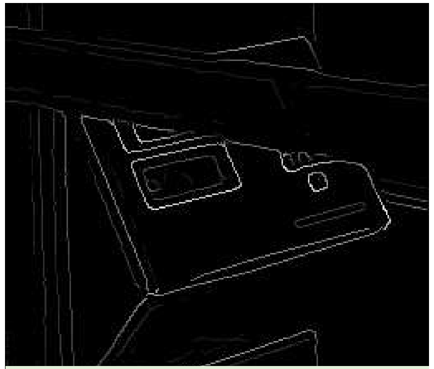
Assignment 3: Workflow

Hough line detection:

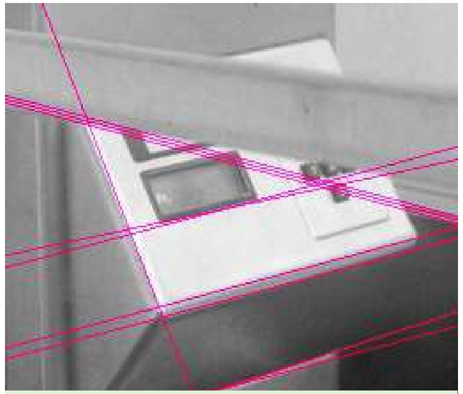
- Grayscale conversion
- Computation of gradient images
- Apply threshold on gradient magnitudes

  → binary edge mask
- Use this edge mask to compute a Hough-voting table
  - Polar coordinates
  - Use edge directions
- Find local maxima in table
- Identify and plot the lines


Grayscale image


Gradient magnitude


Voting space


Result overlay

Polar Line Representation

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

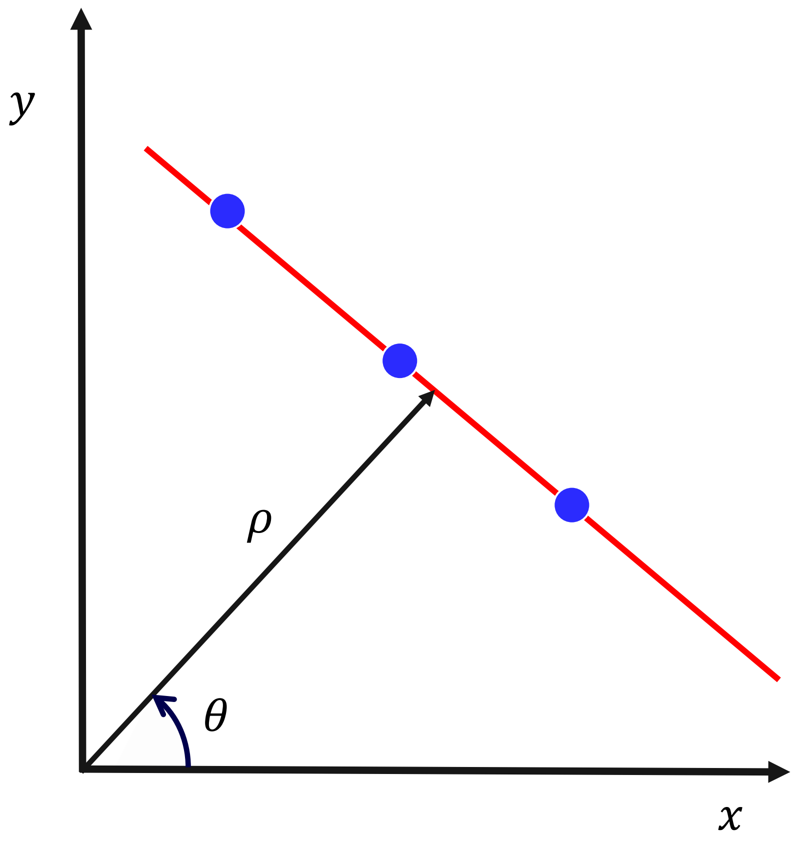
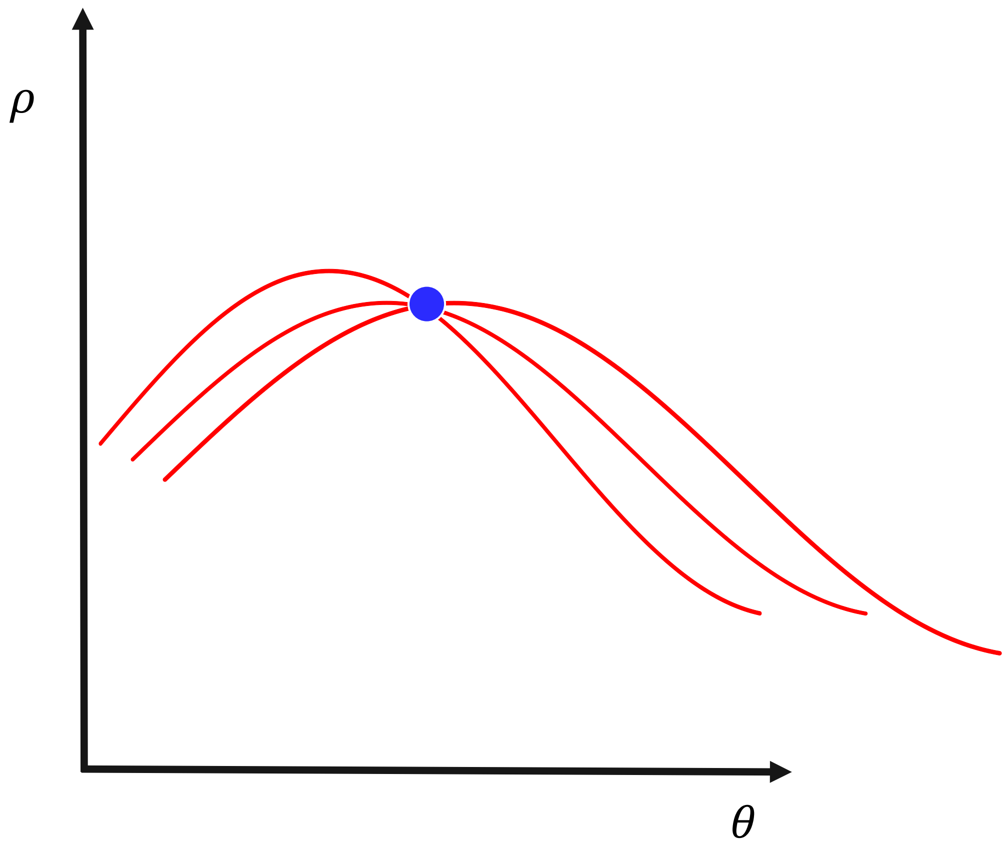


Image space



Hough parameter space

Algorithm Outline

Input: binary edge image (from GoG-filtering + gradient magn. + thresholding)

Initialize index vectors

$$\rho_{ind} = [-\rho_{max}, \dots, \rho_{max}],\ \rho_{max} = \sqrt{n_{row}^2 + n_{col}^2}$$

$$\theta_{ind} = [-90, \dots, 89]$$

Initialize voting array H (integer)

$$H = zeros(num\_rows, num\_cols\ );$$

where $num\_rows = 2 \cdot \rho_{max} + 1$ **and** $num\_cols = 180$

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

for θ = -90 to 89

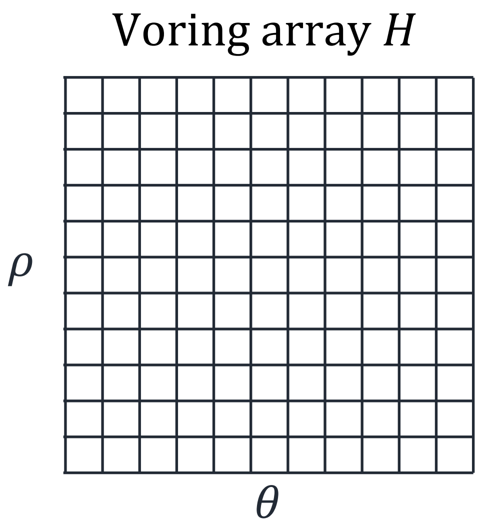
$$\rho = x \cdot cos\theta + y \cdot sin\theta$$

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

end

end

Find the local maxima of H

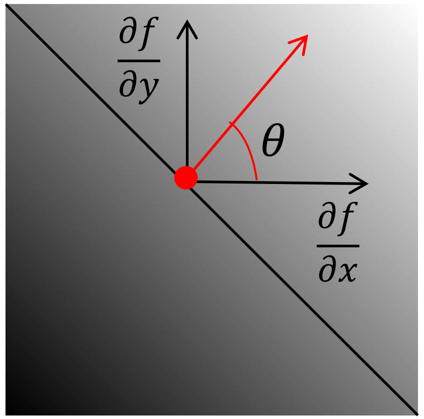


Algorithm Extension

Use the **gradient direction** of detected edges

GoG-filtering → first image derivatives in *x*- and *y*-direction: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$

Gradient direction: $\theta = tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$

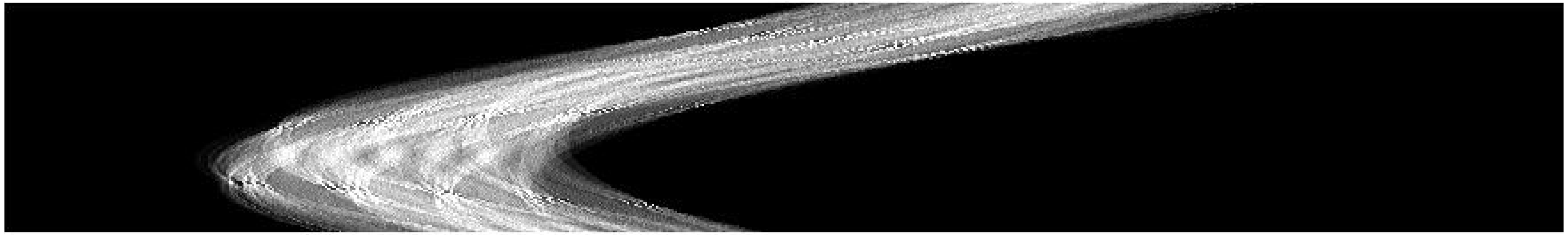


Modified algorithm:

for each edge point (x, y) in the image
$\theta =$ 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:



θ

ρ

Modified algorithm:

θ

ρ

Assignment 3: Tasks

Implement a function that detects lines in an image based on Hough-voting.

Do not use the built-in OpenCV function *cv2.HoughLines()* (you may use it for comparison only).
You are free to use the provided image (*input_ex3.jpg*) or your own photos containing visible straight lines.

a. Read the input image and convert it to a grayscale image with a value range $[0, 1]$. Plot the result image.
b. Apply a GoG filter (from assignment 2) to derive gradient images in *x*- and *y*-direction and compute the gradient magnitude.
c. Find and apply an appropriate threshold on the gradient magnitude to extract representative edge pixels. Plot the binary edge mask.
d. Implement a function for Hough line detection:
    i. Input: Binary edge mask (from c) and gradient images (from b)
    ii. Output: Hough voting array H, index arrays for the ranges of θ and ρ
    iii. Hints:
        1. Use the polar line representation.
        2. Incorporate information about the gradient direction to speedup processing.
e. Plot the resulting Hough voting array H. For better visibility, use the provided **imadjust** function.
f. Find local maxima of H. You may use the provided utility function ***houghpeaks***.
g. Plot the found extrema on top of your figure in step f.
h. Use the provided utility function ***houghlines*** to derive the corresponding line segments.
i. Plot the lines on the figure of step a.

Assignment 3: Tasks and expected results

Note: Do **not** use the OpenCV function *HoughLines()*