Image Analysis and Object Recognition

Exercise 3

Summer Semester 2024

(Course materials for internal use only!)

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

Agenda

Topics:

Assignment 1.	Image enhancement, Binarization, Morphological operators
Assignment 2.	Gradient of Gaussian filtering, Förstner interest operator
Assignment 3.	**Shape detection based on Hough-voting**
Assignment 4.	Filtering in the frequency domain, Fourier descriptors for shape recognition
Assignment 5.	Image segmentation using clustering
Assignment 6.	Convolutional neural networks for image classification
Final Project.	**-** *Will be announced during the last exercise class* **-**

Agenda

Start date and submission deadlines:

Assignment 1.	~~18.04-24 – 01.05.24~~
Assignment 2.	~~02.05.24 – 15.05.24~~
Assignment 3.	**16.05.24 – 29.05.24**
Assignment 4.	30.05.24 – 12.06.24
Assignment 5.	13.06.24 – 26.06.24
Assignment 6.	27.06.24 – 10.07.24
Final Project.	11.07.24 – 22.09.24

Wednesday by 23:00
(Central European Time)

Assignment 2:
Sample Solution

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

```
function Assignment2
%         ===========
sigma = 0.5;                                              % standard deviation
wmin  = 0.004;                                            % minimum cornerness
qmin  = 0.5;                                               % minimum roundness

I = double(imread('ampelmaennchen.png')) / 255;    % convert uint8 to double

[Ix, Iy] = Gradient(mean(I, 3), sigma);     % grayvalue gradient in x and y
mag = sqrt(Ix.^2 + Iy.^2);                                % gradient magnitude
figure; subplot(1, 4, 1); imshow(mag, []); title('Gradient magnitude');

[W, Q] = Foerstner(Ix, Iy);                 % Förstner cornerness and roundness
subplot(1, 4, 2); imshow(W, []); title('Cornerness');
subplot(1, 4, 3); imshow(Q, []); title('Roundness');

W(Q <= qmin) = 0;                                     % remove non-circular points
[r, c] = find(FindMax(W, wmin));                % find row and column coordinates
subplot(1, 4, 4); imshow(I); hold on; plot(c, r, 'r+');
title('Förstner interest points');
```

helper functions

```
function [Ix, Iy] = Gradient(I, sigma)                               % task A
%         ==============================
r = round(3*sigma); i = -r:r;                                   % mask radius
g = exp(-i.^2 / (2*sigma^2)) / (sqrt(2*pi)*sigma);              % 1D-Gaussian
d = -i.*g / sigma^2;                                 % 1D-Gaussian derivative
Ix = conv2(conv2(I, g', 'same'), d , 'same');   % separated GoG convolution
Iy = conv2(conv2(I, g , 'same'), d', 'same');

function [W, Q] = Foerstner(Ix, Iy)                                  % task B
%         ===========================
g = ones(1, 5);                                    % 5x5 accumulation of values
Ix2 = conv2(conv2(Ix.^2,  g, 'same'), g', 'same');
Iy2 = conv2(conv2(Iy.^2,  g, 'same'), g', 'same');            % M = [Ix2 Ixy;
Ixy = conv2(conv2(Ix.*Iy, g, 'same'), g', 'same');            %      Ixy Iy2]
trace = Ix2 + Iy2;
det = Ix2.*Iy2 - Ixy.^2;
W = trace/2 - sqrt((trace/2).^2 - det + eps);                     % cornerness
Q = 4*det./(trace.^2 + eps);                                       % roundness

function R = FindMax(W, wmin)
%         ====================
m = ordfilt2(W, 9, ones(3,3));    % max in 3x3 is 9th element of sorted list
R = (W == m) & (W > wmin);               % find maxima larger than threshold
```

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma}} exp^{-\frac{x^2}{2\sigma^2}}$$

helper functions

```
function [Ix, Iy] = Gradient(I, sigma)                               % task A
%         ==============================
r = round(3*sigma); i = -r:r;                                      % mask radius
g = exp(-i.^2 / (2*sigma^2)) / (sqrt(2*pi)*sigma);                 % 1D-Gaussian
d = -i.*g / sigma^2;                                    % 1D-Gaussian derivative
Ix = conv2(conv2(I, g', 'same'), d , 'same');    % separated GoG convolution
Iy = conv2(conv2(I, g , 'same'), d', 'same');

function [W, Q] = Foerstner(Ix, Iy)                                  % task B
%         ===========================
g = ones(1, 5);                                % 5x5 accumulation of values
Ix2 = conv2(conv2(Ix.^2,  g, 'same'), g', 'same');
Iy2 = conv2(conv2(Iy.^2,  g, 'same'), g', 'same');          % M = [Ix2 Ixy;
Ixy = conv2(conv2(Ix.*Iy, g, 'same'), g', 'same');          %       Ixy Iy2]
trace = Ix2 + Iy2;
det = Ix2.*Iy2 - Ixy.^2;
W = trace/2 - sqrt((trace/2).^2 - det + eps);                     % cornerness
Q = 4*det./(trace.^2 + eps);                                       % roundness

function R = FindMax(W, wmin)
%         ====================
m = ordfilt2(W, 9, ones(3,3));    % max in 3x3 is 9th element of sorted list
R = (W == m) & (W > wmin);               % find maxima larger than threshold
```

Element-wise power operator

comparison between the operators ^ and .^

```
>> a = ones(3)
a =

   3    3    3
   3    3    3
   3    3    3


>> b = a^2
b =

   27   27   27
   27   27   27
   27   27   27


>> c = a.^2
c =

   9    9    9
   9    9    9
   9    9    9
```

helper functions

```
function [Ix, Iy] = Gradient(I, sigma)                               % task A
%        ==============================
r = round(3*sigma); i = -r:r;                                     % mask radius
g = exp(-i.^2 / (2*sigma^2)) / (sqrt(2*pi)*sigma);                % 1D-Gaussian
d = -i.*g / sigma^2;                                  % 1D-Gaussian derivative
Ix = conv2(conv2(I, g', 'same'), d , 'same');    % separated GoG convolution
Iy = conv2(conv2(I, g , 'same'), d', 'same');

function [W, Q] = Foerstner(Ix, Iy)                                  % task B
%        ===========================
g = ones(1, 5);                                  % 5x5 accumulation of values
Ix2 = conv2(conv2(Ix.^2,  g, 'same'), g', 'same');
Iy2 = conv2(conv2(Iy.^2,  g, 'same'), g', 'same');           % M = [Ix2 Ixy;
Ixy = conv2(conv2(Ix.*Iy, g, 'same'), g', 'same');           %       Ixy Iy2]
trace = Ix2 + Iy2;
det = Ix2.*Iy2 - Ixy.^2;
W = trace/2 - sqrt((trace/2).^2 - det + eps);                     % cornerness
Q = 4*det./(trace.^2 + eps);                                       % roundness

function R = FindMax(W, wmin)
%        ====================
m = ordfilt2(W, 9, ones(3,3));    % max in 3x3 is 9th element of sorted list
R = (W == m) & (W > wmin);               % find maxima larger than threshold
```

Implicit auto-correlation matrix

prevention of numerical instabilities due to rounding effects

Bauhaus-Universität Weimar

helper functions

```
function [Ix, Iy] = Gradient(I, sigma)                          % task A
%         ==============================
r = round(3*sigma); i = -r:r;                                   % mask radius
g = exp(-i.^2 / (2*sigma^2)) / (sqrt(2*pi)*sigma);              % 1D-Gaussian
d = -i.*g / sigma^2;                                            % 1D-Gaussian derivative
Ix = conv2(conv2(I, g', 'same'), d , 'same');    % separated GoG convolution
Iy = conv2(conv2(I, g , 'same'), d', 'same');

function [W, Q] = Foerstner(Ix, Iy)                              % task B
%         ===========================
g = ones(1, 5);                                  % 5x5 accumulation of values
Ix2 = conv2(conv2(Ix.^2,  g, 'same'), g', 'same');
Iy2 = conv2(conv2(Iy.^2,  g, 'same'), g', 'same');          % M = [Ix2 Ixy;
Ixy = conv2(conv2(Ix.*Iy, g, 'same'), g', 'same');          %      Ixy Iy2]
trace = Ix2 + Iy2;
det = Ix2.*Iy2 - Ixy.^2;
W = trace/2 - sqrt((trace/2).^2 - det + eps);                  % cornerness
Q = 4*det./(trace.^2 + eps);                                    % roundness

function R = FindMax(W, wmin)
%         ====================
m = ordfilt2(W, 9, ones(3,3));   % max in 3x3 is 9th element of sorted list
R = (W == m) & (W > wmin);                % find maxima larger than threshold
```

2D order-statistic filtering
here: maximum filter

also available in Octave
within the image package

small example with
maximum order-statistic filtering

```
A =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

>> B = ordfilt2(A,9,ones(3,3))

B =

    24    24    24    16    16
    24    24    24    22    22
    23    23    21    22    22
    18    25    25    25    22
    18    25    25    25    21
```

Assignment 2 – 2D order-statistic filtering

	function parameters	domain	sample data
median filter	`B = ordfilt2(A,5,ones(3,3))`	1 1 1 1 1 1 1 1 1	88 16 56 5 3 30 21 63 42
minimum filter	`B = ordfilt2(A,1,ones(3,3))`	1 1 1 1 1 1 1 1 1	88 16 56 5 3 30 21 63 42
maximum filter	`B = ordfilt2(A,9,ones(3,3))`	1 1 1 1 1 1 1 1 1	88 16 56 5 3 30 21 63 42
?	`B = ordfilt2(A,1, [0 1 0; 1 0 1; 0 1 0])`	0 1 0 1 0 1 0 1 0	88 16 56 5 3 30 21 63 42

source: https://uk.mathworks.com/help/images/ref/ordfilt2.html

Bauhaus-Universität Weimar

Assignment 2 – 2D order-statistic filtering

	function parameters	domain	sample data
median filter	`B = ordfilt2(A,5,ones(3,3))`	$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$	$\begin{bmatrix}88&16&56\\5&3&30\\21&63&42\end{bmatrix}$
minimum filter	`B = ordfilt2(A,1,ones(3,3))`	$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$	$\begin{bmatrix}88&16&56\\5&3&30\\21&63&42\end{bmatrix}$
maximum filter	`B = ordfilt2(A,9,ones(3,3))`	$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$	$\begin{bmatrix}88&16&56\\5&3&30\\21&63&42\end{bmatrix}$
minimum filter	`B = ordfilt2(A,1, [0 1 0; 1 0 1; 0 1 0])`	$\begin{bmatrix}0&1&0\\1&0&1\\0&1&0\end{bmatrix}$	$\begin{bmatrix}88&16&56\\5&3&30\\21&63&42\end{bmatrix}$

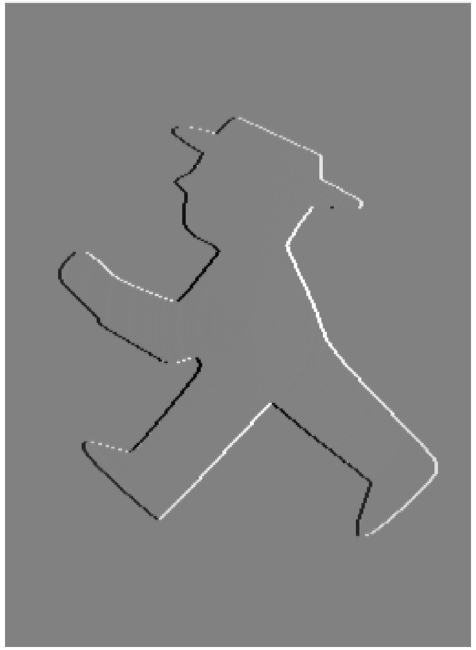
source: https://uk.mathworks.com/help/images/ref/ordfilt2.html

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

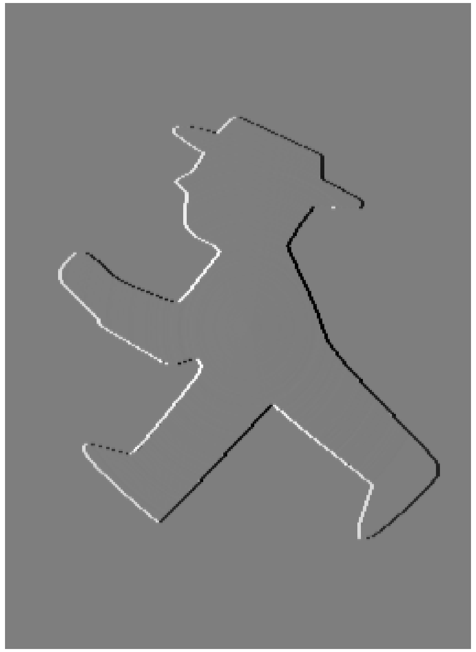
I – grayscale input image

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

$\otimes$ Cross-Correlation



$\star$ Convolution

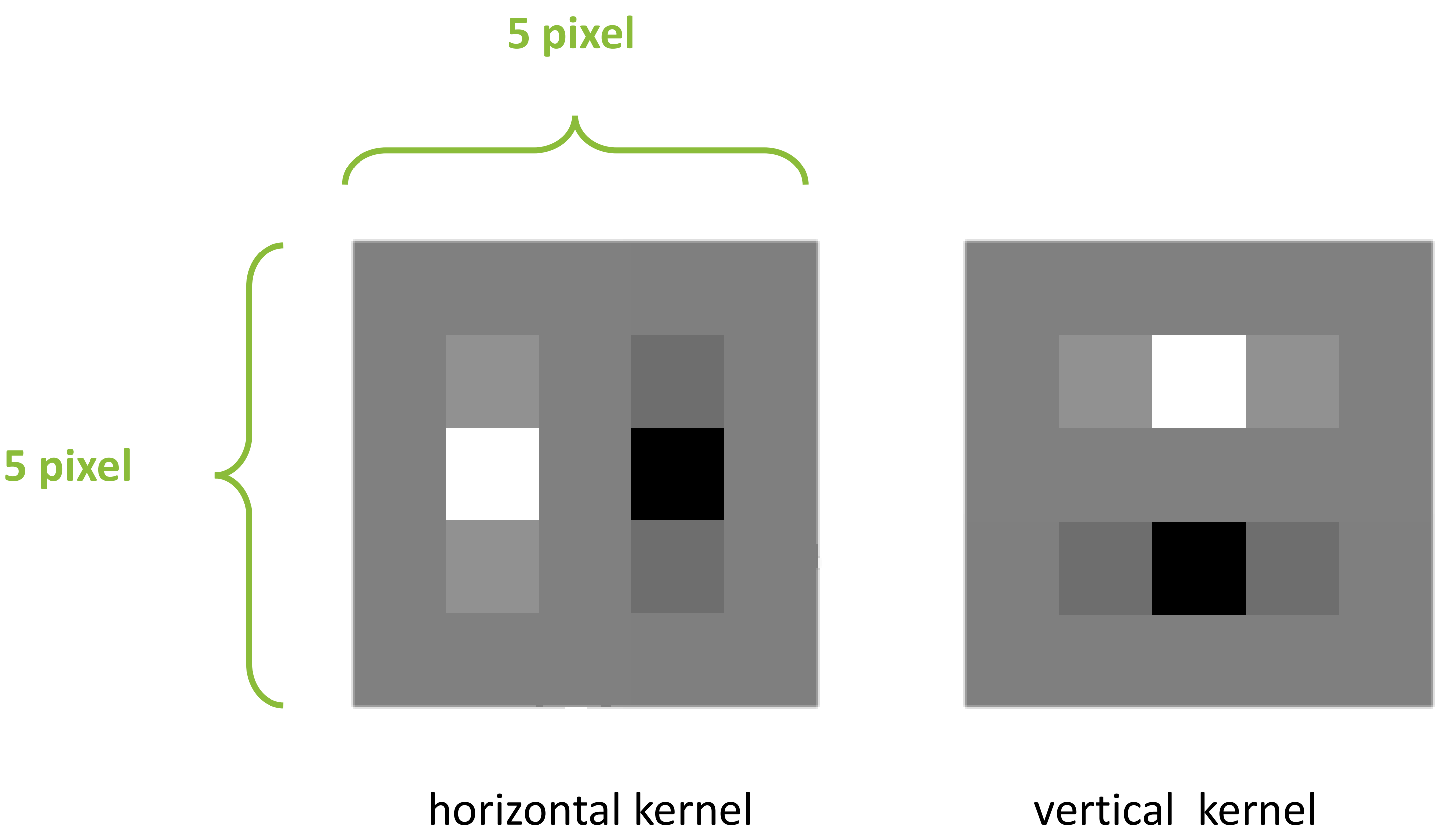


Horizontal gradient images

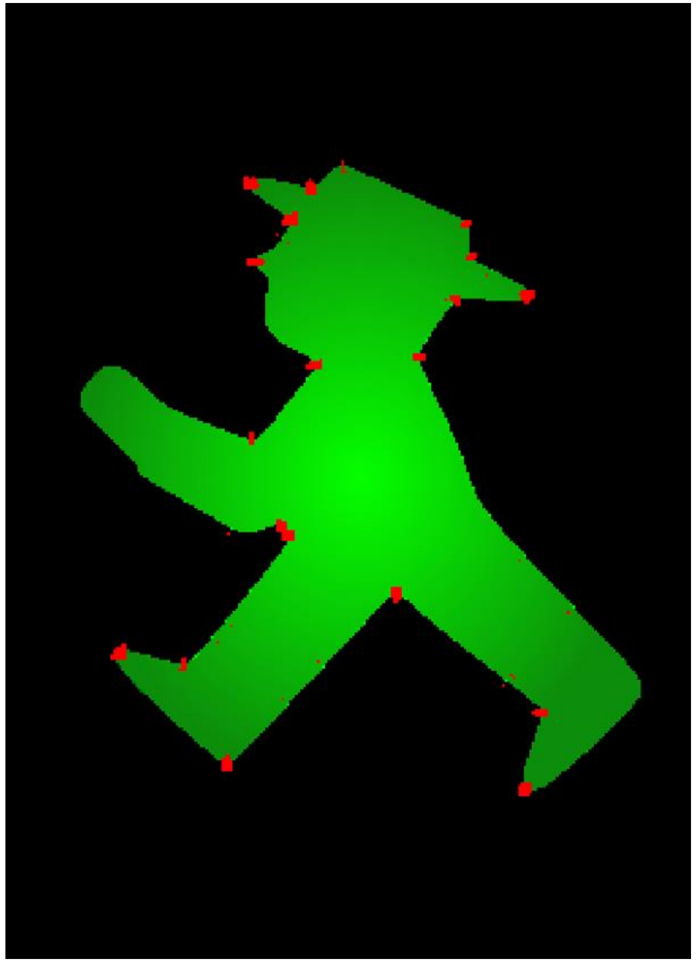
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

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

Assignment 2 – convolution kernel visualization



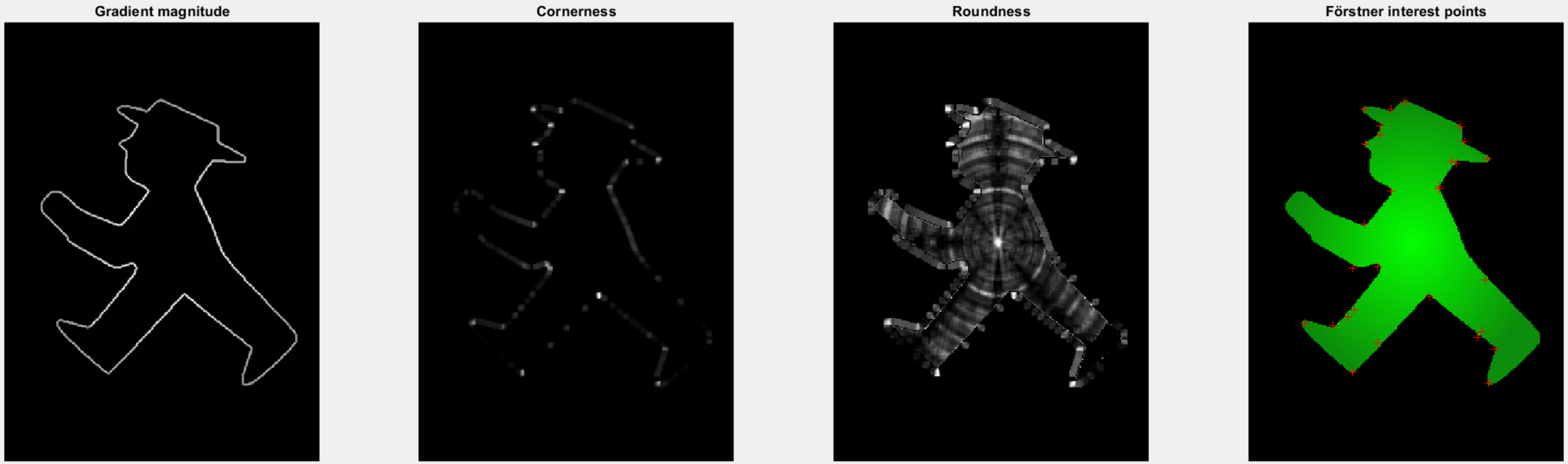
Assignment 2 – sample results



Choose the
max response

for each 3-by-3
neighbourhood

Assignment 2 – sample results



Assignment 3

Assignment 3: Overview

Topics:

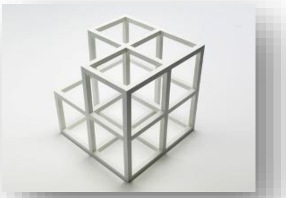
- Hough line detection

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



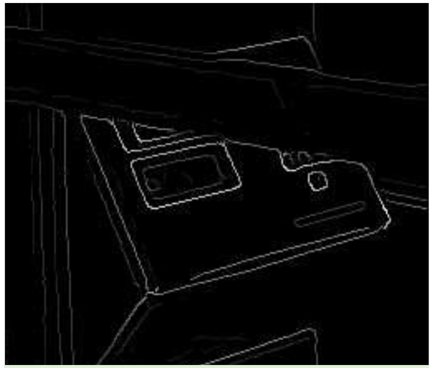
Assignment 3: Workflow

Hough line detection:

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

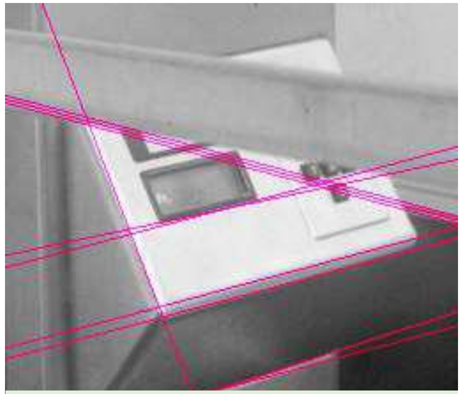
  → binary edge mask
- Use this edge mask to compute a Hough-voting table
  - Polar coordinates
  - Use edge directions
- Find local maxima in table
- Identify and plot the lines

Grayscale image



Gradient magnitude

Voting space



Result overlay

Bauhaus-Universität Weimar

Polar Line Representation

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

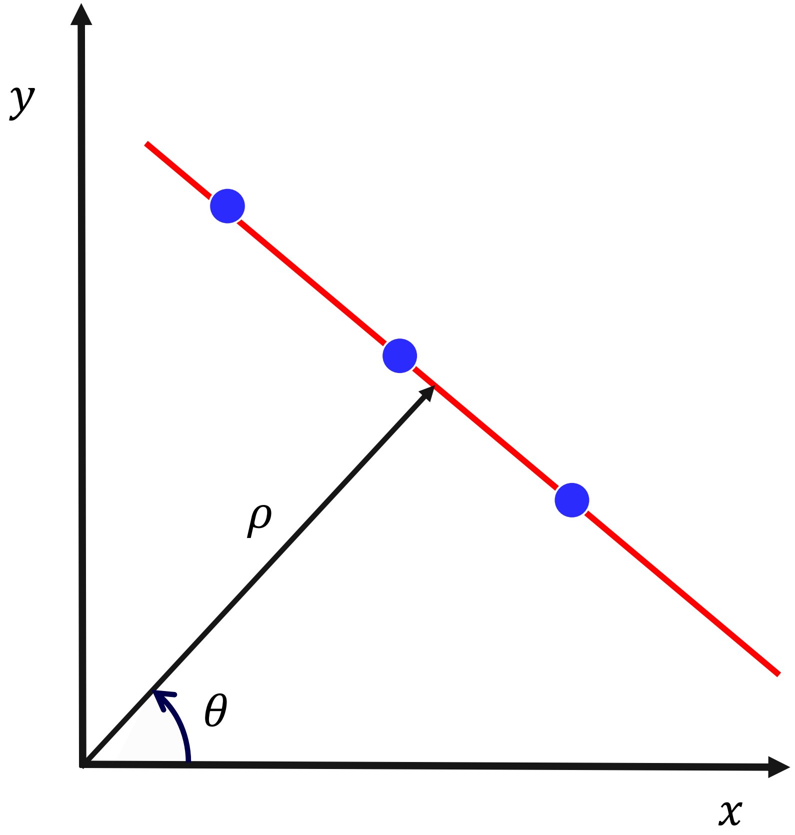
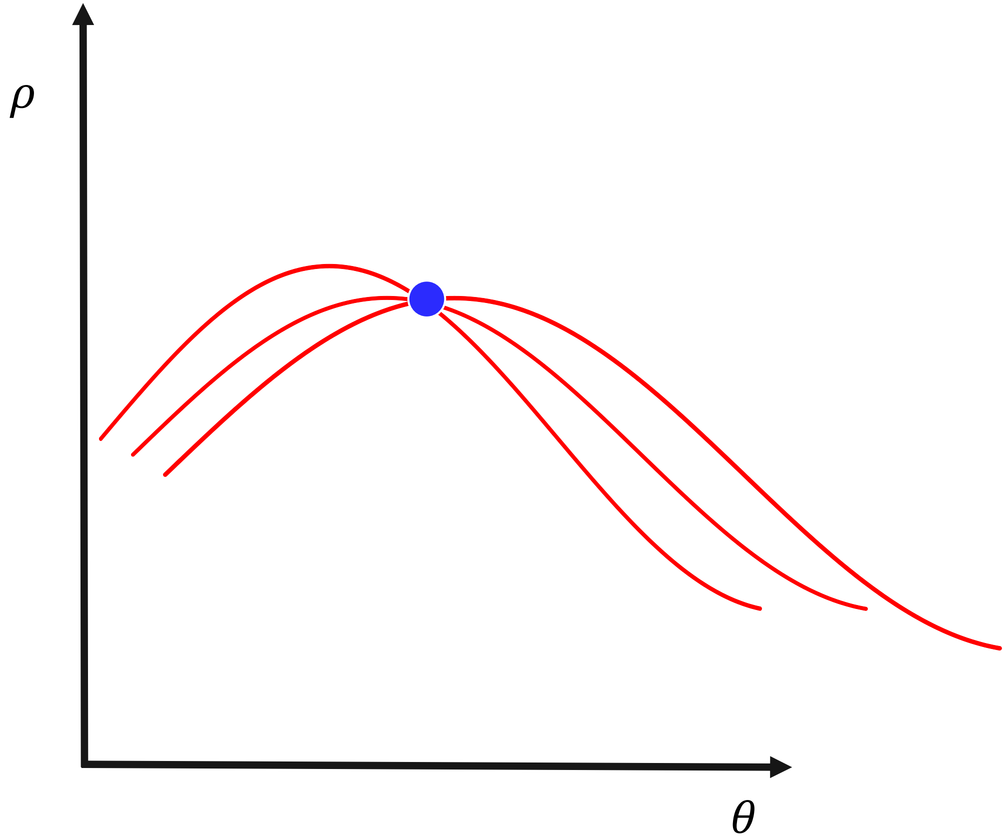


Image space



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

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

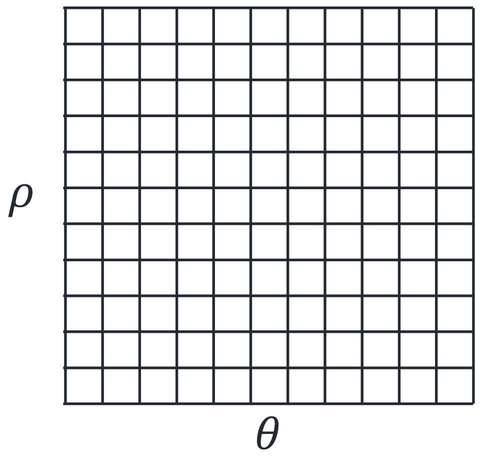
    $H(\ \rho_i, \theta_i,) = H(\ \rho_i, \theta_i) + 1$

  end

end

Find the local maxima of H

Voring array H

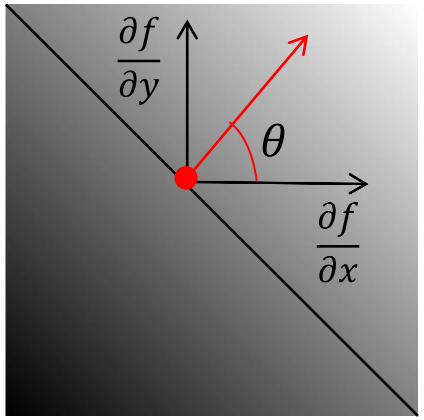


Algorithm Extension

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

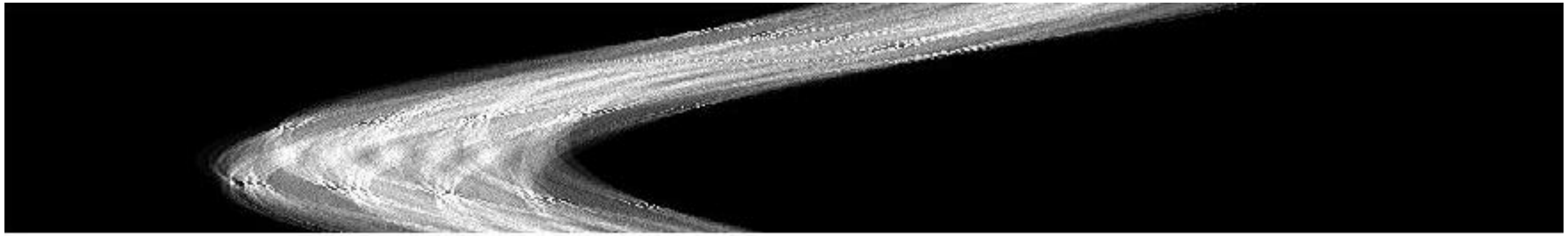


Modified algorithm:

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

Bauhaus-Universität Weimar

Assignment 3: Tasks

Implement a function that detects lines in an image based on **Hough-voting**. Do **not** use the built-in function *hough* (you may use it for comparison only).

a. Read the input image and convert it to a grayscale image with a value range $[0, \ldots, 1]$. Plot the result image.

b. Apply a GoG filter (from assignment 2) in order 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 derive 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.

f. Find local maxima of H. You may use the built-in function ***houghpeaks***.

g. Plot the found extrema on top of your figure in step f.

h. Use the built-in function ***houghlines*** to derive the corresponding line segments.

i. Plot the lines on the figure of step a.

<u>Note</u>: When working with Octave, make sure that you have loaded the **image package** before using the functions ***houghpeaks*** and ***houghlines.***

Assignment 3: Tasks and expected results

Note: Do **not** use the MATLAB/Octave function *hough*!