Image Analysis and Object Recognition

Exercise 4

Summer Semester 2025

(Course materials for internal use only!)

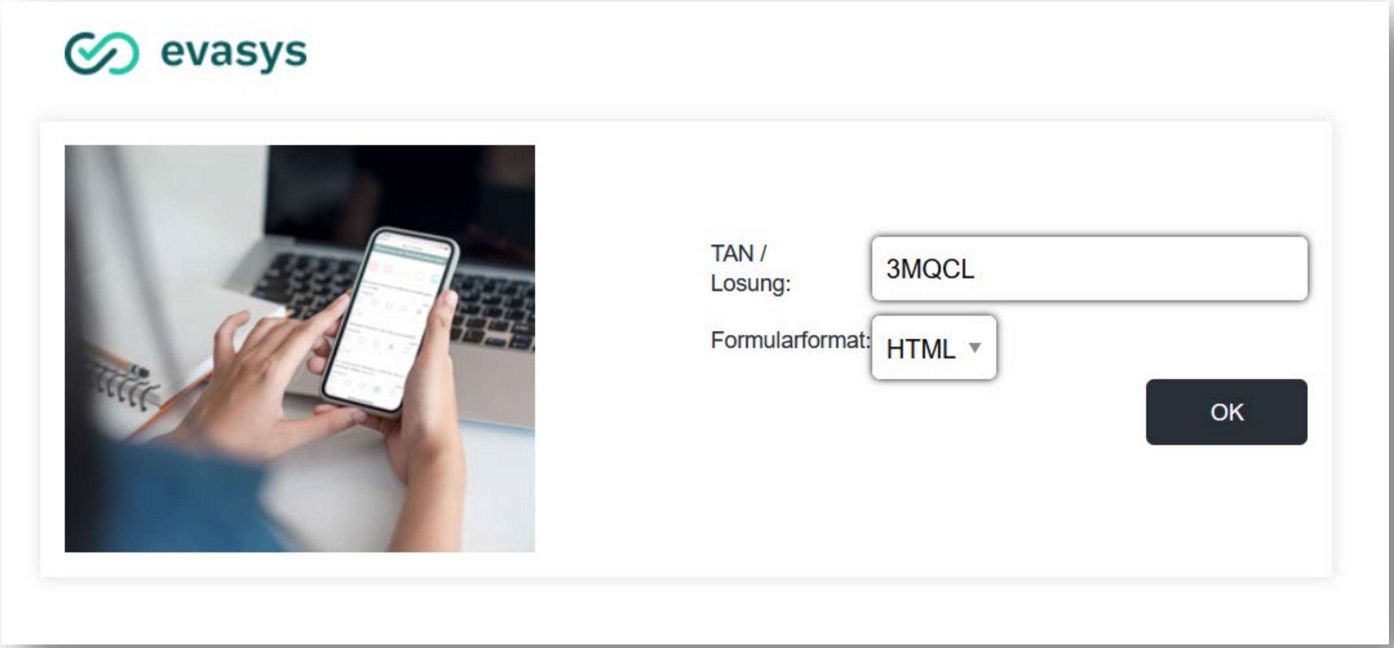
Computer Vision in Engineering – Prof. Dr. Rodehorst
M.Sc. Mariya Kaisheva
mariya.kaisheva@uni-weimar.de

Online Course Evaluation

Teaching Evaluation:

URL: ***https://cloud14.evasys.de/uniweimar/online/***

Code: ***3MQCL***



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 3:
Sample Solution

Bauhaus-
Universität
Weimar

Assignment 3: Overview

Topics:

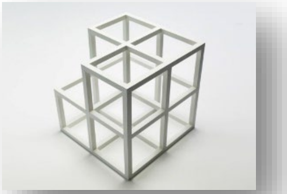
- Hough line detection

Goal:

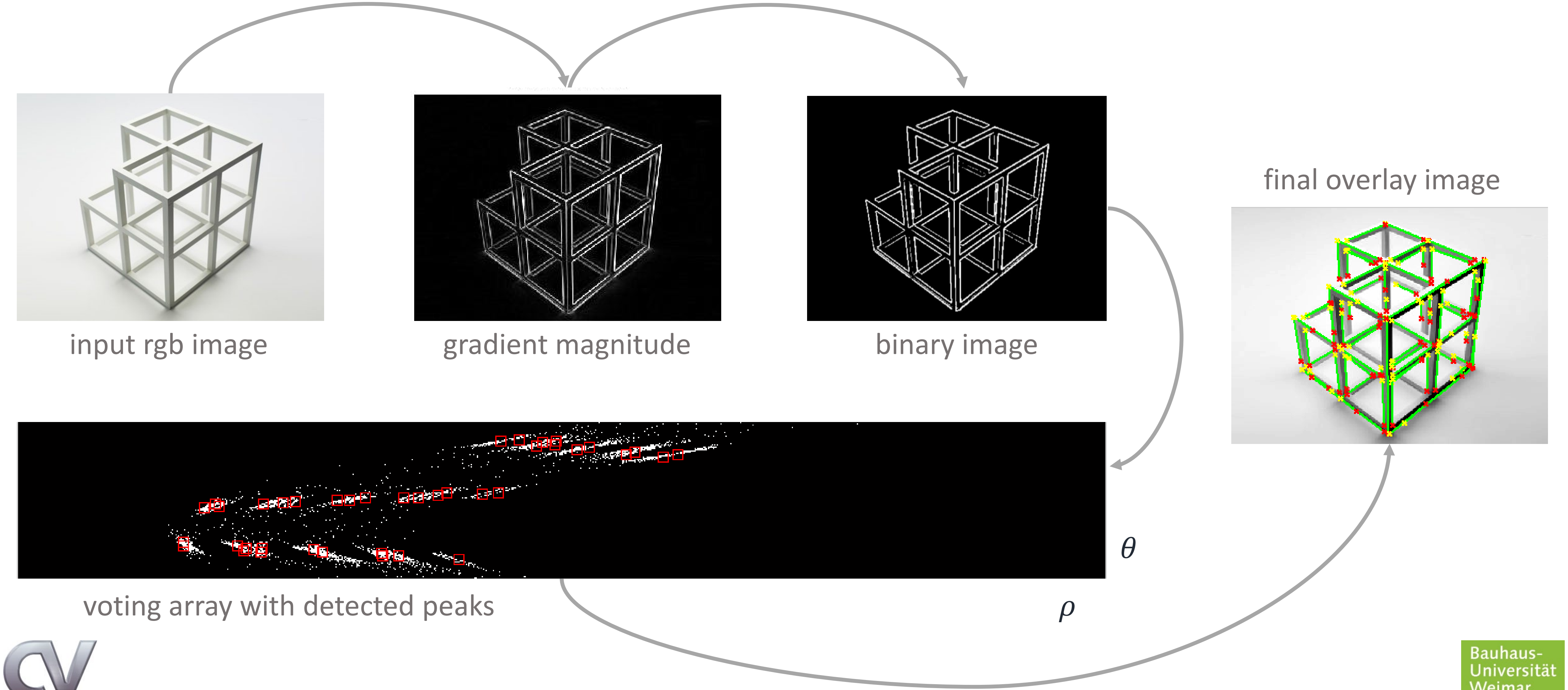
- Understanding the concept of Hough-voting
- Practice detection and parameterization of lines in images

Input:

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



Assignment 3: workflow



Algorithm outline

Input: binary edge image (from GoG-filtering)

Initialize index vectors

$$\boldsymbol{\rho_{ind}} = [-\rho_{max}, \dots, \rho_{max}],\ \rho_{max} = \sqrt{n_{rows}^2 + n_{columns}^2}$$

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

Initialize voting array H

$$H = zeros(2 \cdot \rho_{max} + 1,\ 180)$$

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

$\boxed{\theta}$ = gradient orientation at (x, y)

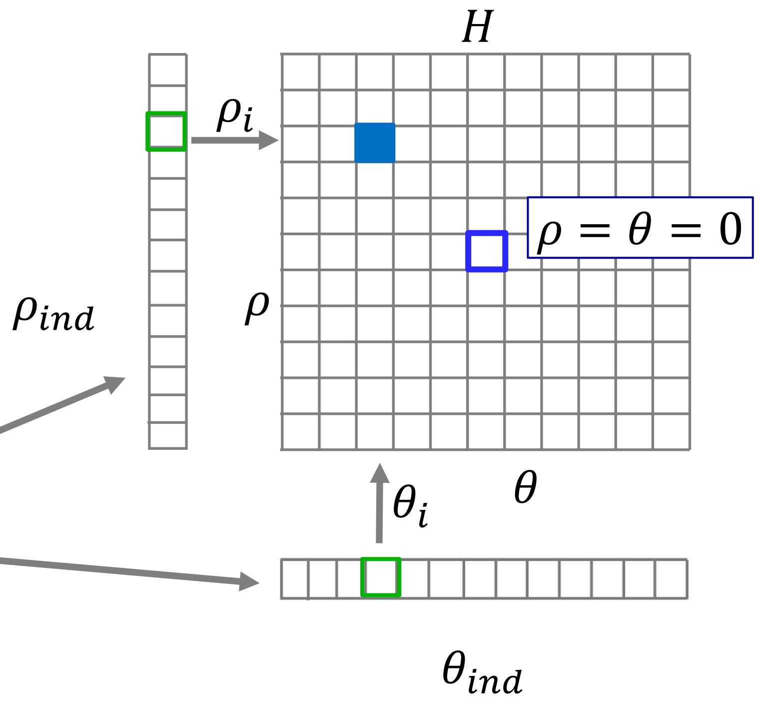
$\boxed{\rho} = x \cdot cos\theta + y \cdot sin\theta$

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

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

$\boxed{H(\rho_i, \theta_i)} = H(\rho_i, \theta_i) + 1$

end



main function

```
def assignment3():
    sigma = 0.5                                       # standard deviation for smoothing
    thres = 0.07                                      # binarization threshold

    img = np.array(Image.open('input_ex3.jpg')).astype(np.float64) / 255.0
    I_gray = np.mean(img, axis=2)                     # Convert to grayscale

    Ix, Iy = gradient(I_gray, sigma)                  # Calculate image gradients
    M = np.sqrt(Ix**2 + Iy**2)                        # Calculate gradient magnitude
    BW = M > thres                                    # Compute binary edge mask
    H, t, r = my_hough(BW, Ix, Iy)                    # Apply Hough transform

    # Apply contrast enhancement
    if np.max(H) > 0:                                 # Avoid division by zero
        H_adjusted = imadjust(H.astype(np.float64))
        print("At least one element was non zero")
    else:
        H_adjusted = H
        print("No element was non zero")

    peaks = houghpeaks(H, 40, threshold=10) # Find peaks in Hough space
    lines = houghlines(BW, t, r, peaks, fill_gap=5, min_length=10) # Compute lines

    display_results(img,M,BW,H_adjusted,t,r,peaks,lines)
```

my_hough

modified algorithm

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

degree → radian conversion

```python
def my_hough(BW, Ix, Iy):
        rows, cols = BW.shape             # Calculate image diagonal to determine rho range
        d = int(np.round(np.sqrt(rows**2 + cols**2)))

        t = np.arange(-90, 90)            # theta values: -90 to 89
        r = np.arange(-d, d + 1)          # rho values: -d to d

        H = np.zeros((len(r), len(t)))    # Initialize Hough voting space
        y, x = np.nonzero(BW)             # Find edge pixels

        # Vote for each edge pixel
        for i in range(len(x)):
            # Calculate gradient direction at this pixel
            theta = np.round(np.arctan2(Iy[y[i], x[i]], Ix[y[i], x[i]]) * 180 / np.pi).astype(int)

            # Ensure theta is within our range
            while theta < -90:
                theta += 180
            while theta >= 90:
                theta -= 180

            # Calculate rho for this pixel and theta
            rho = int(np.round(x[i] * np.cos(theta * np.pi / 180) + y[i] * np.sin(theta * np.pi / 180)))

            # Find indices in the Hough array
            ind_r = np.where(r == rho)[0]
            ind_t = np.where(t == theta)[0]

            if len(ind_r) > 0 and len(ind_t) > 0:
                H[ind_r[0], ind_t[0]] += 1 # Vote for this (theta, rho) combination

        return H, t, r
```

display_results

```
def display_results(img,M,BW,H_adjusted,t,r,peaks,lines):

    plt.figure(figsize=(20, 5)) # Create figure for visualization

    plt.subplot(1, 5, 1); plt.imshow(img); plt.title('Original image‘); plt.axis('off')
    plt.subplot(1, 5, 2); plt.imshow(M, cmap='gray'); plt.title('Gradient magnitude‘); plt.axis('off')
    plt.subplot(1, 5, 3); plt.imshow(BW, cmap='gray'); plt.title('Binarized gradient‘); plt.axis('off')
    plt.subplot(1, 5, 4)
    plt.imshow(H_adjusted, extent=[t[0]-0.5, t[-1]+0.5, r[-1]+0.5, r[0]-0.5],
               aspect='auto', cmap='gray', interpolation='nearest')
    plt.title('Voting space'); plt.xlabel('θ'); plt.ylabel('ρ'); plt.axis('on')

    # Plot peaks with minimal 1-pixel square outline
    # Note: We need to adjust for the flipped Y-axis in our visualization
    plt.subplot(1, 5, 4)
    plt.plot(t[peaks[:, 1]], r[peaks[:, 0]], 's', mfc='none', mec='red', markersize=5, markeredgewidth=1)

    plt.subplot(1, 5, 5)
    plt.imshow(img); plt.title('Original image with detected lines’); plt.axis('off')

    for line in lines:
        p1 = line['point1']
        p2 = line['point2']
        plt.plot([p1[0], p2[0]], [p1[1], p2[1]], 'g-', linewidth=2)
        plt.plot(p1[0], p1[1], 'yx', markersize=10)
        plt.plot(p2[0], p2[1], 'rx', markersize=10)

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

Assignment 4

Assignment 4: Overview

Topics:

- Filtering in the frequency domain
- Shape recognition using Fourier descriptors

Goal:

- Practice noise removal in the frequency domain (Task A)
- Practice automatic shape detection using Fourier descriptors (Task B)

Input:

- All images provided for this assignment can be found on Moodle course page

Assignment 4:Image filtering in frequency domain

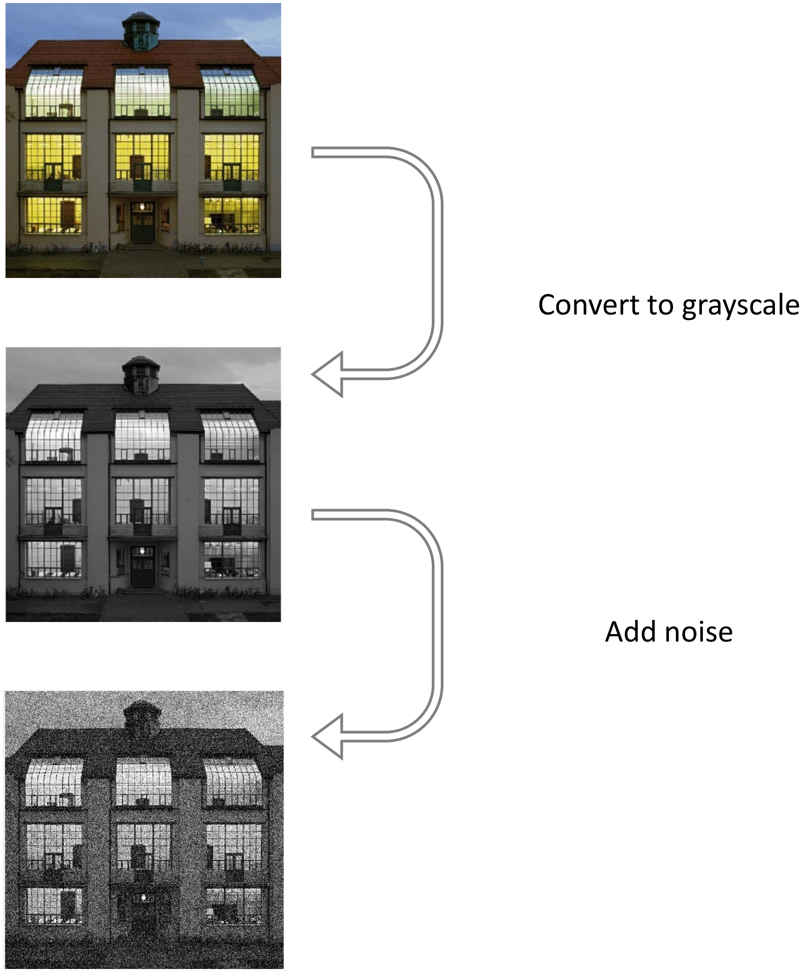
Task A: Image filtering

a. Read the input image *taskA.png* and convert it to a grayscale image (double values between 0.0 and 1.0)

b. Add Gaussian noise to the image (parameters e.g. M=0, V=0.01) and plot the result

c. Filter the noisy image with a self-made 2D Gaussian filter in the frequency-domain (`fft2, ifft2`). Which σ is suitable to remove the noise? Plot the result

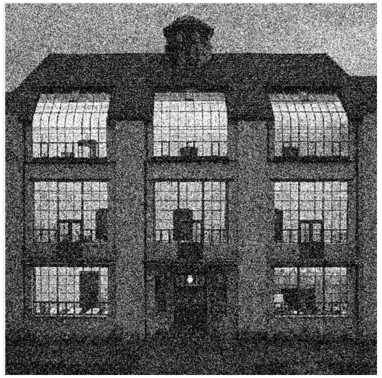
d. Plot the logarithmic centered image spectra of the noisy image, the (padded) Gaussian filter and the filtered image (`log`, `abs` and `fftshift`)



Bauhaus-Universität Weimar

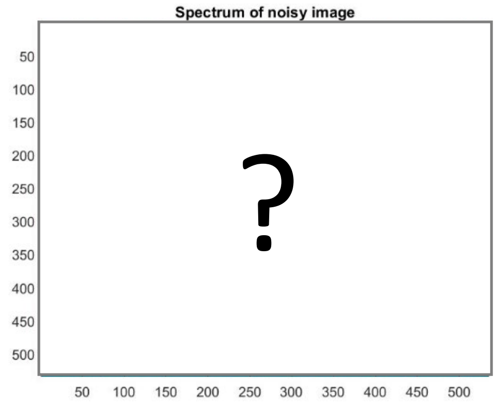
Task A

$f(x, y)$



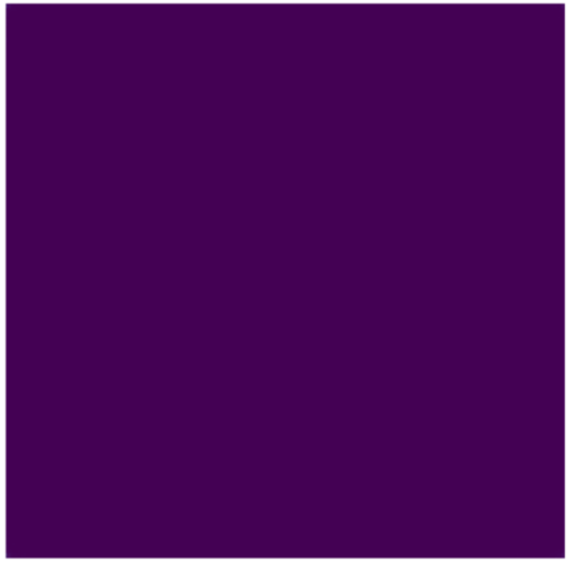
FFT

$\Rightarrow$



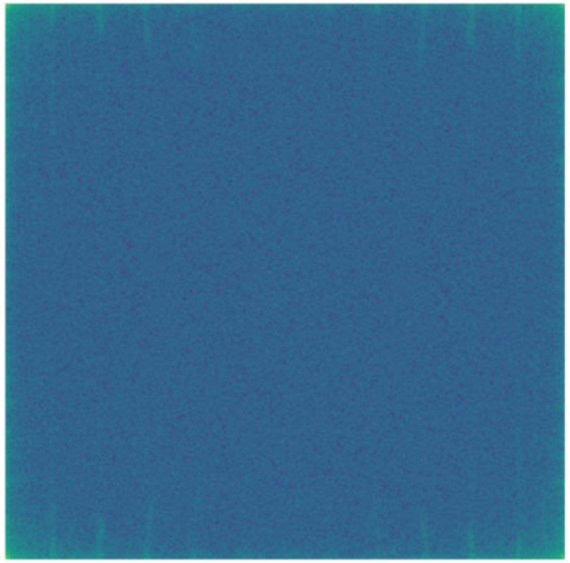
$F(u, v)$

image spectrum



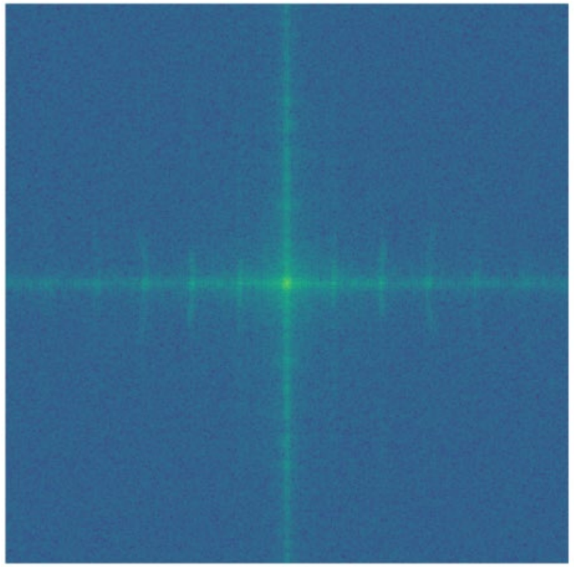
abs(fft_image)

logarithmic **scaled** image spectrum



log*(abs(fft_image))*

centered scaled image spectrum

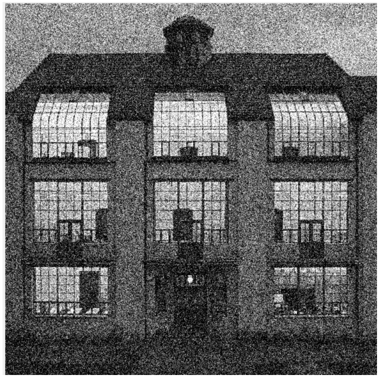


*log(abs(****fftshift****(fft_image)))*

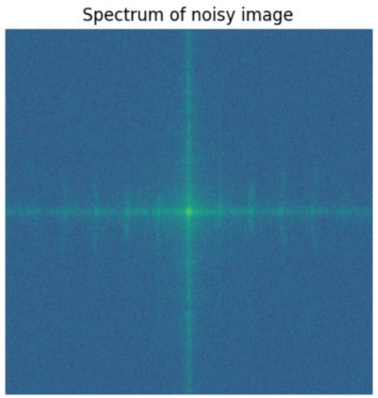
Bauhaus-Universität Weimar

Task A

$f(x, y)$



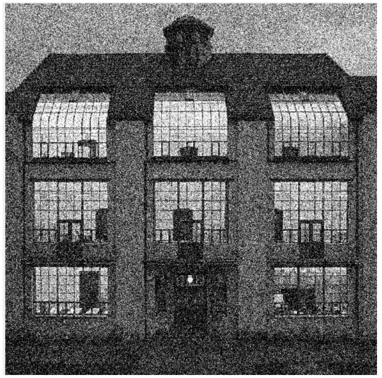
FFT
$\Rightarrow$



$F(u, v)$

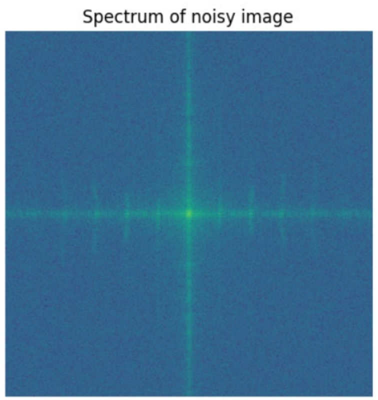
Task A

$f(x, y)$



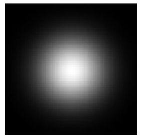
FFT

$\Rightarrow$



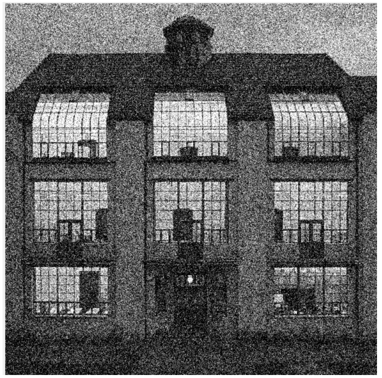
$F(u, v)$

$h(x, y)$



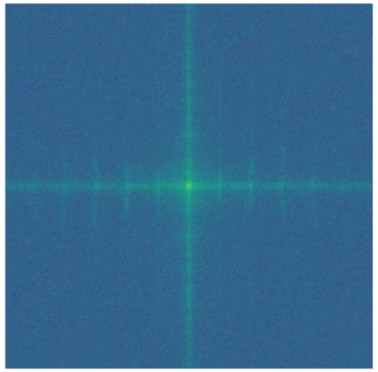
Task A

$f(x, y)$



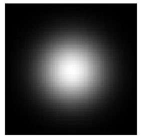
FFT

$\Rightarrow$



$F(u, v)$

$h(x, y)$

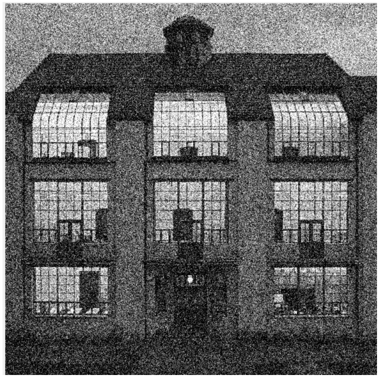


Padding

is necessary

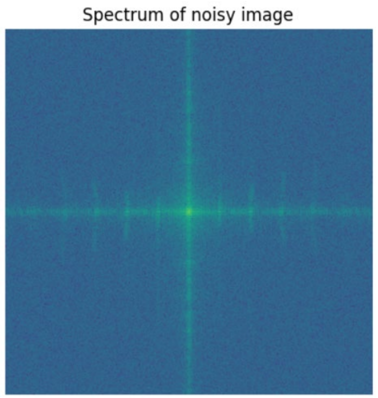
Task A

$f(x, y)$



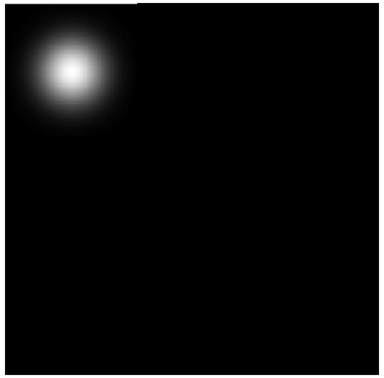
FFT

$\Rightarrow$



$F(u, v)$

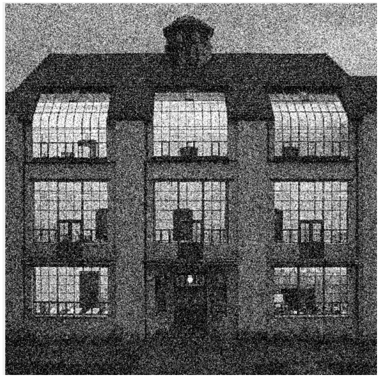
$h(x, y)$



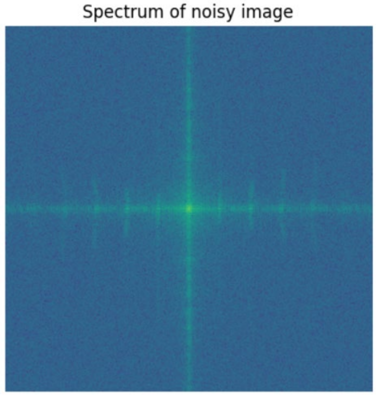
Filter after padding

Task A

$f(x, y)$

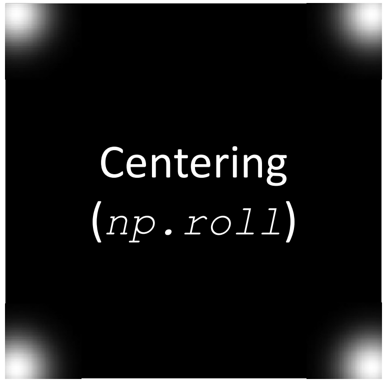


FFT $\Rightarrow$



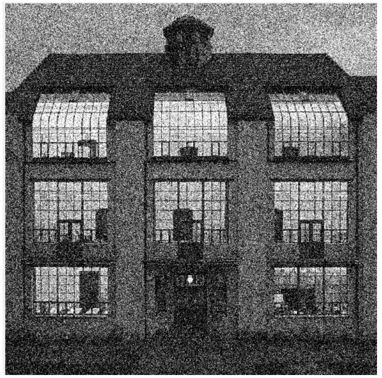
$F(u, v)$

$h(x, y)$



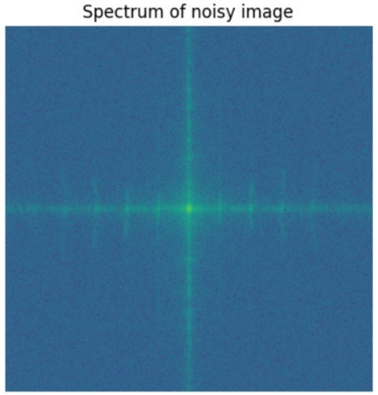
Task A

$f(x, y)$



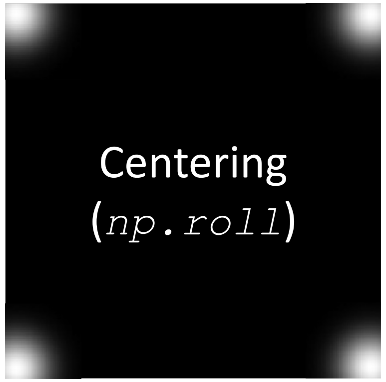
FFT

$\Rightarrow$



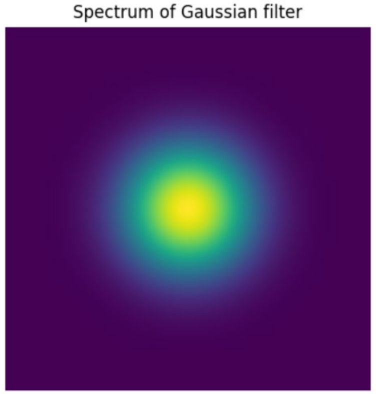
$F(u, v)$

$h(x, y)$



FFT

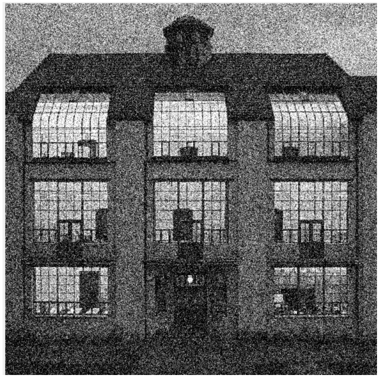
$\Rightarrow$



$H(u, v)$

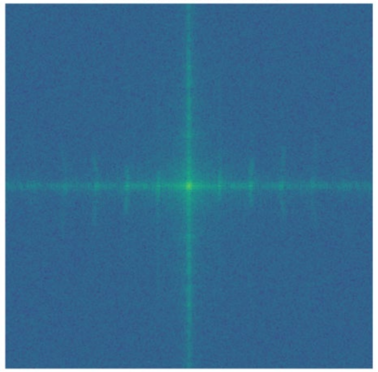
Task A

$f(x,y)$



FFT $\Rightarrow$

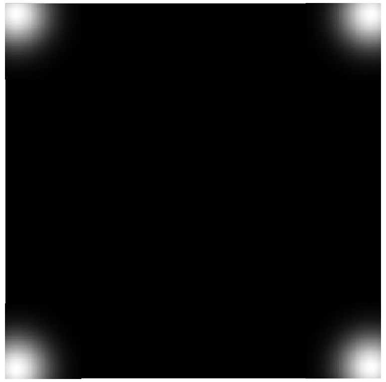
Spectrum of noisy image



$F(u,v)$

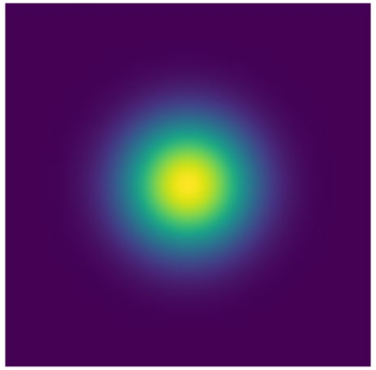
$.*$

$h(x,y)$



FFT $\Rightarrow$

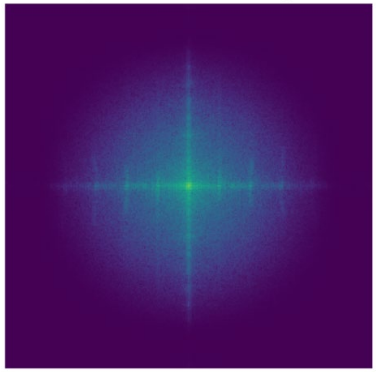
Spectrum of Gaussian filter



$H(u,v)$

$\Downarrow$

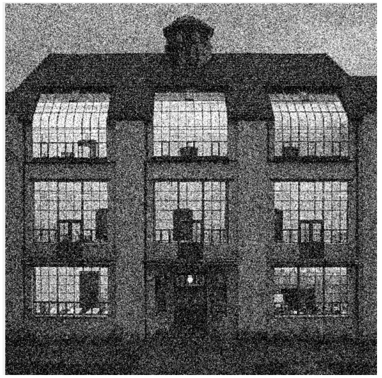
Spectrum of filtered image



$G(u,v)$

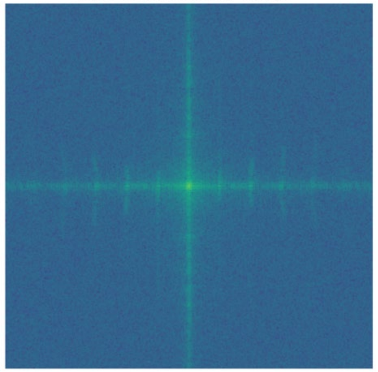
Task A

$f(x,y)$



FFT $\Rightarrow$

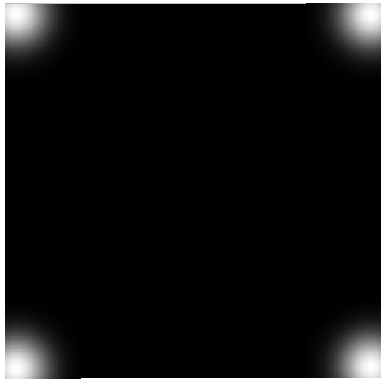
Spectrum of noisy image



$F(u,v)$

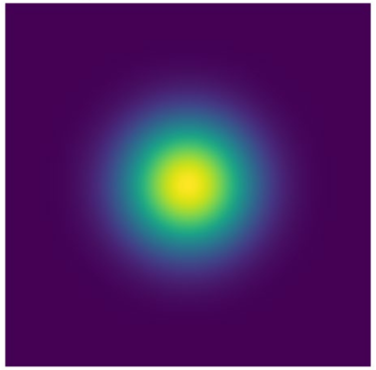
$.*$

$h(x,y)$



FFT $\Rightarrow$

Spectrum of Gaussian filter



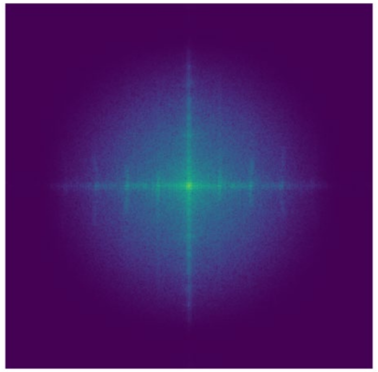
$H(u,v)$

$\Downarrow$

$g(x,y)$

FFT^{-1} $\Leftarrow$

Spectrum of filtered image



$G(u,v)$

Assignment 4:Shape recognition

Task B: Image filtering

a. Read the image trainB.png and convert it to a grayscale image (double values between 0.0 and 1.0)

b. Derive a binary mask of the image where 1 represents the object of interest and 0 is background

c. Build a Fourier-descriptor D_f based on the binary mask of b.
    - Extraction of boundaries of the binary mask
    - Use n=24 elements for the descriptor
    - Make it invariant against translation, orientation and scale

d. Apply steps a.-c. on the images test1B.jpg, test2B.jpg and test3B.jpg in order to identify all potential object boundaries in the images. Note that here more than one boundaries will be identified in the binary mask.

e. Identify the searched object by comparison of the trained Fourier-descriptor (result of task c) with all identified descriptors of the two test images (result of task d). Use the Euclidean distance of the Fourier-descriptors for identification, i.e.

$$\text{norm}\left(D_{f,train} - D_{f,test}\right) < 0.09$$

f. Plot the identified boundaries on your mask (result of task b.) in order to validate the results

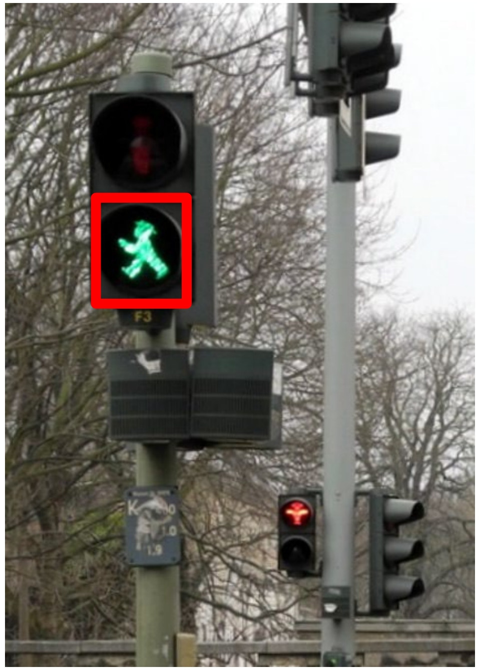
Input data

Task B

training image

test image 1

test image 2

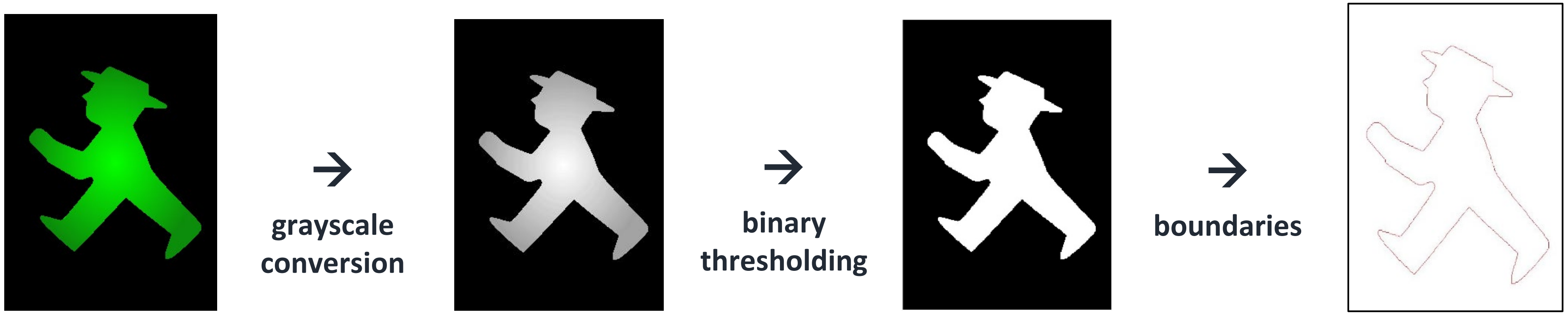


test image 3

Bauhaus-Universität Weimar

Boundary extraction

Task B



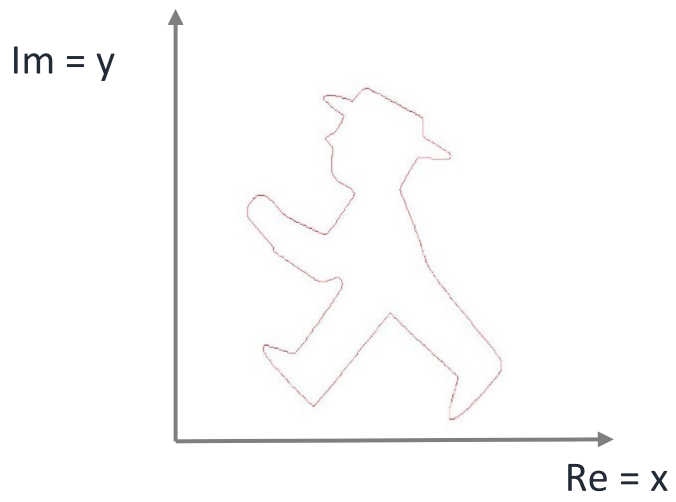
Fourier descriptor

Task B

- **Given**: m points representing the boundary of a **closed** region in the image
- Interpret the boundary coordinates (x, y) as complex numbers
- $b = \begin{bmatrix} (y_1, x_1) \\ \vdots \\ (y_m, x_m) \end{bmatrix}$ ($m \times 2$ array: output of contour/boundary detection)
- Build the ***complex vector*** $\boldsymbol{D}$:

$$D = b(:,2) + \boldsymbol{i} * b(:,1);$$

where $\boldsymbol{i}^2 = -1$



Note: In Python, the built-in **imaginary unit is** denoted by ***j***

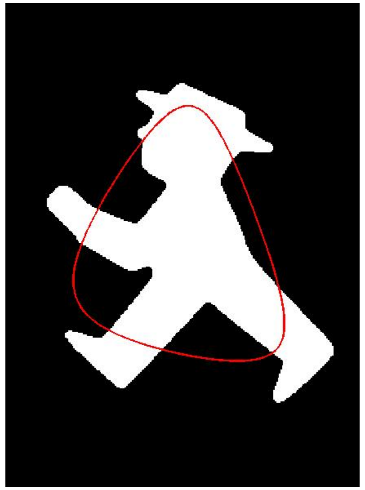
Reducing the number of elements n in D_f → shape generalization

Task B

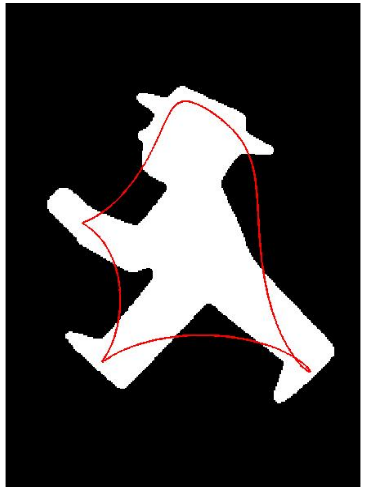
$$D_f = \underbrace{[\;\square\;\square\;\cdots\;\square\;]}_{1 \;\ldots\; m}$$

low frequencies — highest frequency (center) — low frequencies

$n = 2$



$n = 4$



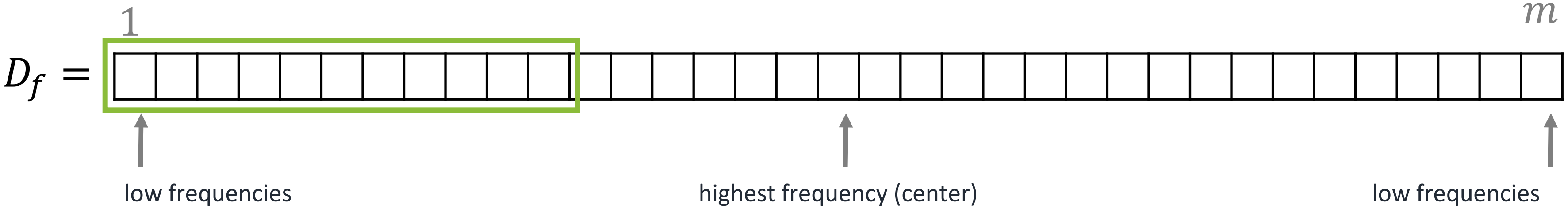
$n = 8$

$n = 16$

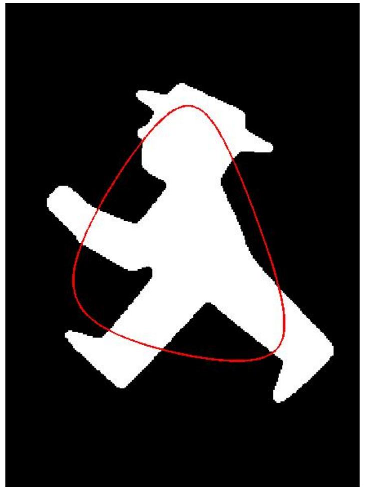
$n = 24$

Reducing the number of elements n in D_f → shape generalization

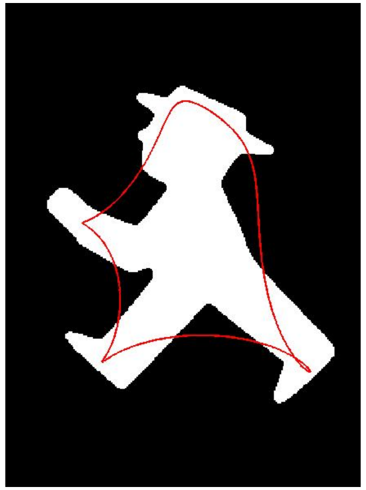
Task B



$n = 2$



$n = 4$



$n = 8$

$n = 16$

$n = 24$

Extract only the **first n** elements (low frequency values) of the **Fourier-descriptor D_f** and ignore the rest

Bauhaus-Universität Weimar

Expected results

Task B

training image