Image Analysis and Object Recognition

Exercise 4

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.	**Frequency domain filtering, Shape recognition via Fourier descriptors**
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 3:
Sample Solution

Assignment 3: Overview

Topics:

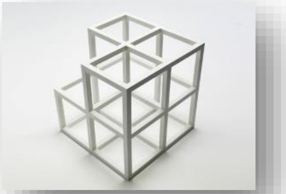
- Hough line detection

Goal:

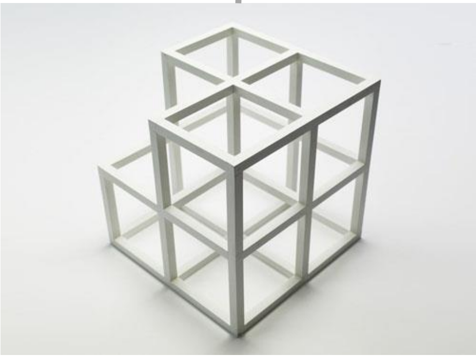
- Understanding the concept of Hough-voting
- Practice detection and parameterization of lines in images

Input:

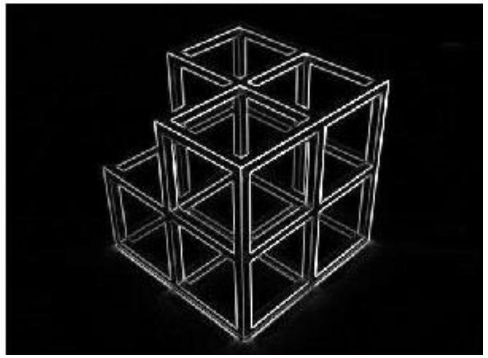
- Provided image → *input_ex3.jpg*
- Or a different image of your own choice



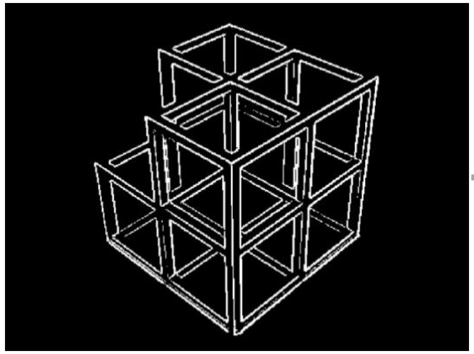
Assignment 3: workflow



input rgb image

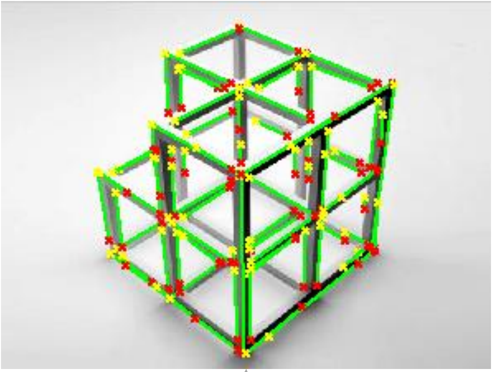


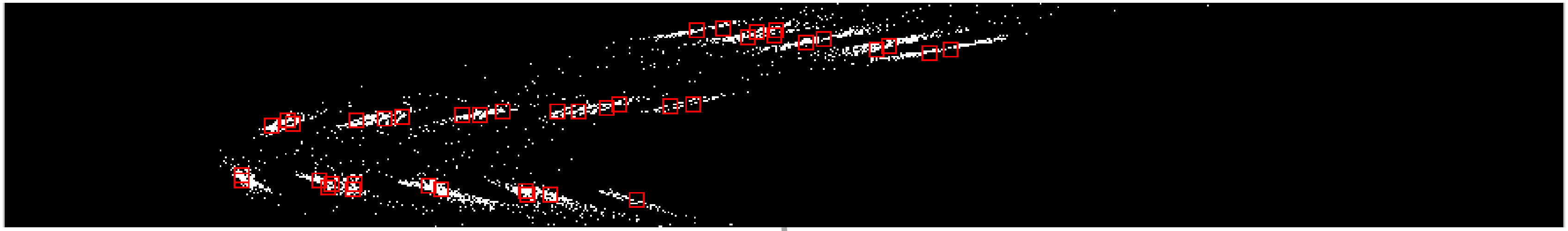
gradient magnitude



binary image

final overlay image





voting array with detected peaks

Algorithm outline

Input: binary edge image (from GoG-filtering)

Initialize index vectors

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

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

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)

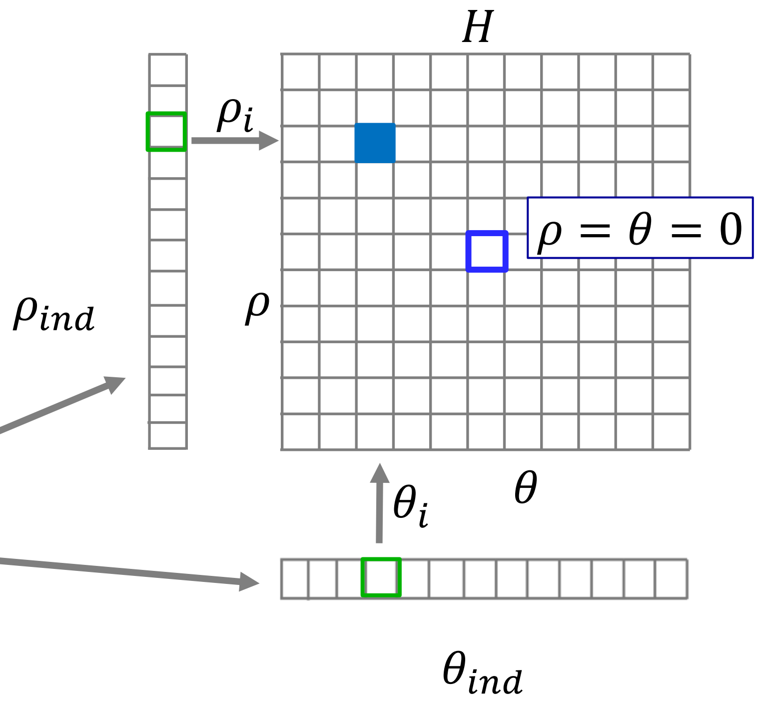
$\boxed{\rho} = x \cdot cos\theta + y \cdot sin\theta$

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

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

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

end



main function

```
function Assignment3
%        ===========
sigma = 0.5;                                   % standard deviation for smoothing
thres = 0.07;                                          % binarization threshold
I = double(imread('input_ex3.jpg')) / 255;
figure; subplot(1, 4, 1); imshow(I); title('Original image');

[Ix, Iy] = Gradient(mean(I, 3), sigma);   % image gradient from assignment 2
M = sqrt(Ix.^2 + Iy.^2);                                   % gradient magnitude
subplot(1, 4, 2); imshow(M, []); title('Gradient magnitude');
BW = M > thres;                                                   % binary mask
subplot(1, 4, 3); imshow(BW); title('Binarized gradient');

[H, t, r] = my_hough(BW, Ix, Iy);                         % own Hough transform
subplot(1, 4, 4); imshow(imadjust(H), 'XData', t, 'YData', r);
title('Voting space'); ylabel('\rho'); xlabel('\theta');

peaks = houghpeaks(H, 40, 'threshold', 10);                   % find max values
hold on; plot(t(peaks(:,2)), r(peaks(:,1)), 's', 'color', 'red');

lines = houghlines(BW, t, r, peaks, 'FillGap', 5, 'MinLength', 10);
subplot(1, 4, 1); hold on;
for i = 1 : length(lines)                                  % draw line segments
    xy = [lines(i).point1; lines(i).point2];
    plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'green');
    plot(xy(1,1), xy(1,2), 'x', 'LineWidth', 2, 'Color', 'yellow');
    plot(xy(2,1), xy(2,2), 'x', 'LineWidth', 2, 'Color', 'red');
end
```

`imshow(I,[])` displays the grayscale image `I`, scaling the display based on the range of pixel values in `I`.

`imshow(I)` displays the grayscale image `I`, using the default display range for the image data type.

helper function

```
function [H, t, r] = my_hough(BW, Ix, Iy)
%                    ====================
d = round(sqrt(size(BW, 1)^2 + size(BW, 2)^2));          % image diagonal
t = -90:1:89; r = -d:1:d;                          % ranges for theta and rho
H = zeros(length(r), length(t));              % initialize Hough voting space
[y, x] = find(BW > 0);                             % relevant pixel positions
for i = 1 : length(x)
    theta = round(atan2(Iy(y(i), x(i)), Ix(y(i), x(i))) * 180/pi);
    rho = round(x(i) * cos(theta * pi/180) + y(i) * sin(theta * pi/180));
    ind_r = find(r == rho);                            % index lookup for rho
    ind_t = find(t == theta);                        % index lookup for theta
    H(ind_r, ind_t) = H(ind_r, ind_t) + 1;           % vote for theta and rho
end
```

modified algorithm

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

$atan2(y, x)$ computes $\mathrm{atan}(y/x)$ for corresponding elements of y and x

helper function

```
function [H, t, r] = my_hough(BW, Ix, Iy)
%                    ====================
d = round(sqrt(size(BW, 1)^2 + size(BW, 2)^2));              % image diagonal
t = -90:1:89; r = -d:1:d;                            % ranges for theta and rho
H = zeros(length(r), length(t));                % initialize Hough voting space
[y, x] = find(BW > 0);                               % relevant pixel positions
for i = 1 : length(x)
    theta = round(atan2(Iy(y(i), x(i)), Ix(y(i), x(i))) * 180/pi);
    rho = round(x(i) * cos(theta * pi/180) + y(i) * sin(theta * pi/180));
    ind_r = find(r == rho);                              % index lookup for rho
    ind_t = find(t == theta);                          % index lookup for theta
    H(ind_r, ind_t) = H(ind_r, ind_t) + 1;             % vote for theta and rho
end
```

round up to the next integer value

helper function

```
function [H, t, r] = my_hough(BW, Ix, Iy)
%                    ====================
d = round(sqrt(size(BW, 1)^2 + size(BW, 2)^2));          % image diagonal
t = -90:1:89; r = -d:1:d;                          % ranges for theta and rho
H = zeros(length(r), length(t));              % initialize Hough voting space
[y, x] = find(BW > 0);                             % relevant pixel positions
for i = 1 : length(x)
    theta = round(atan2(Iy(y(i), x(i)), Ix(y(i), x(i))) * 180/pi);
    rho = round(x(i) * cos(theta * pi/180) + y(i) * sin(theta * pi/180));
    ind_r = find(r == rho);                             % index lookup for rho
    ind_t = find(t == theta);                         % index lookup for theta
    H(ind_r, ind_t) = H(ind_r, ind_t) + 1;            % vote for theta and rho
end
```

degree → radian conversion

cos( ) and ***sin( )*** require input in **radians**

alternative functions which work with input in **degrees** are ***cosd( )*** and ***sind( )***

main function

```
function Assignment3
%        ===========
sigma = 0.5;                        % standard deviation for smoothing
thres = 0.07;                                 % binarization threshold
I = double(imread('input_ex3.jpg')) / 255;
figure; subplot(1, 4, 1); imshow(I); title('Original image');

[Ix, Iy] = Gradient(mean(I, 3), sigma);  % image gradient from assignment 2
M = sqrt(Ix.^2 + Iy.^2);                          % gradient magnitude
subplot(1, 4, 2); imshow(M, []); title('Gradient magnitude');
BW = M > thres;                                          % binary mask
subplot(1, 4, 3); imshow(BW); title('Binarized gradient');

[H, t, r] = my_hough(BW, Ix, Iy);                 % own Hough transform
subplot(1, 4, 4); imshow(imadjust(H), 'XData', t, 'YData', r);
title('Voting space'); ylabel('\rho'); xlabel('\theta');

peaks = houghpeaks(H, 40, 'threshold', 10);            % find max values
hold on; plot(t(peaks(:,2)), r(peaks(:,1)), 's', 'color', 'red');

lines = houghlines(BW, t, r, peaks, 'FillGap', 5, 'MinLength', 10);
subplot(1, 4, 1); hold on;
for i = 1 : length(lines)                         % draw line segments
    xy = [lines(i).point1; lines(i).point2];
    plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'green');
    plot(xy(1,1), xy(1,2), 'x', 'LineWidth', 2, 'Color', 'yellow');
    plot(xy(2,1), xy(2,2), 'x', 'LineWidth', 2, 'Color', 'red');
end
```

increases the contrast

main function

```matlab
function Assignment3
%         ===========
sigma = 0.5;                            % standard deviation for smoothing
thres = 0.07;                                     % binarization threshold
I = double(imread('input_ex3.jpg')) / 255;
figure; subplot(1, 4, 1); imshow(I); title('Original image');

[Ix, Iy] = Gradient(mean(I, 3), sigma);  % image gradient from assignment 2
M = sqrt(Ix.^2 + Iy.^2);                              % gradient magnitude
subplot(1, 4, 2); imshow(M, []); title('Gradient magnitude');
BW = M > thres;                                              % binary mask
subplot(1, 4, 3); imshow(BW); title('Binarized gradient');

[H, t, r] = my_hough(BW, Ix, Iy);                    % own Hough transform
subplot(1, 4, 4); imshow(imadjust(H), 'XData', t, 'YData', r);
title('Voting space'); ylabel('\rho'); xlabel('\theta');

peaks = houghpeaks(H, 40, 'threshold', 10);               % find max values
hold on; plot(t(peaks(:,2)), r(peaks(:,1)), 's', 'color', 'red');

lines = houghlines(BW, t, r, peaks, 'FillGap', 5, 'MinLength', 10);
subplot(1, 4, 1); hold on;
for i = 1 : length(lines)                             % draw line segments
    xy = [lines(i).point1; lines(i).point2];
    plot(xy(:,1), xy(:,2), 'LineWidth', 2, 'Color', 'green');
    plot(xy(1,1), xy(1,2), 'x', 'LineWidth', 2, 'Color', 'yellow');
    plot(xy(2,1), xy(2,2), 'x', 'LineWidth', 2, 'Color', 'red');
end
```

maximum **number of peaks**

maximum value to be considered a peak

Bauhaus-Universität Weimar

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

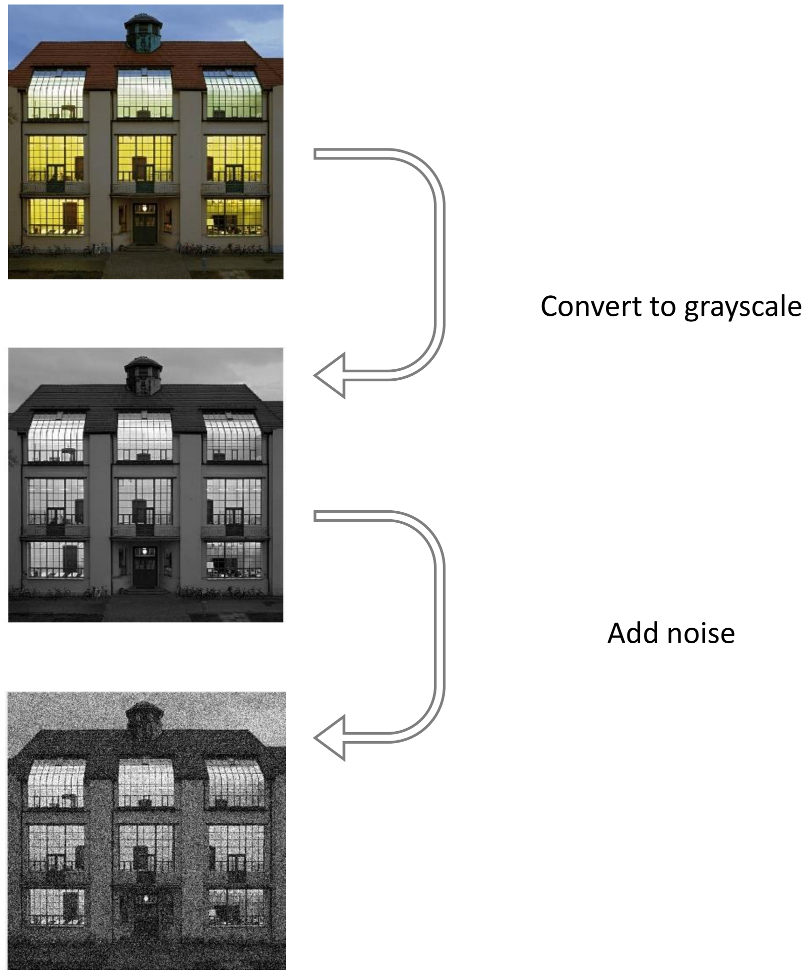
Task A: Image filtering

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 (`imnoise`, 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

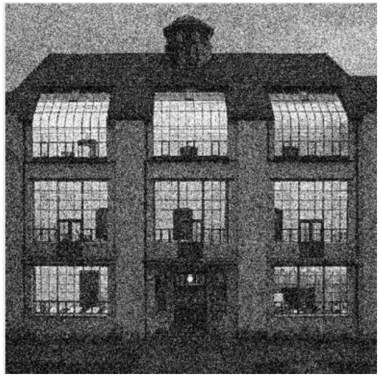
d. Plot the logarithmic centered image spectra of the noisy image, the (padded) Gaussian filter and the filtered image (`imagesc`, `log`, `abs` and `fftshift`)



Bauhaus-Universität Weimar

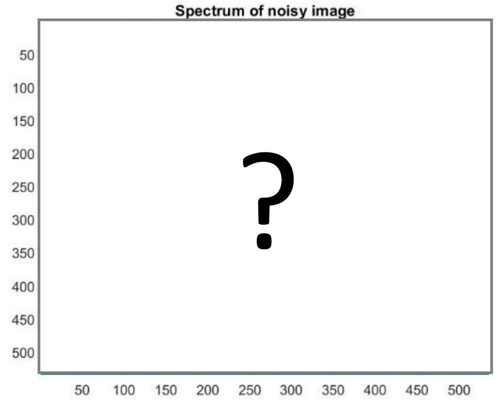
Task A

$f(x, y)$



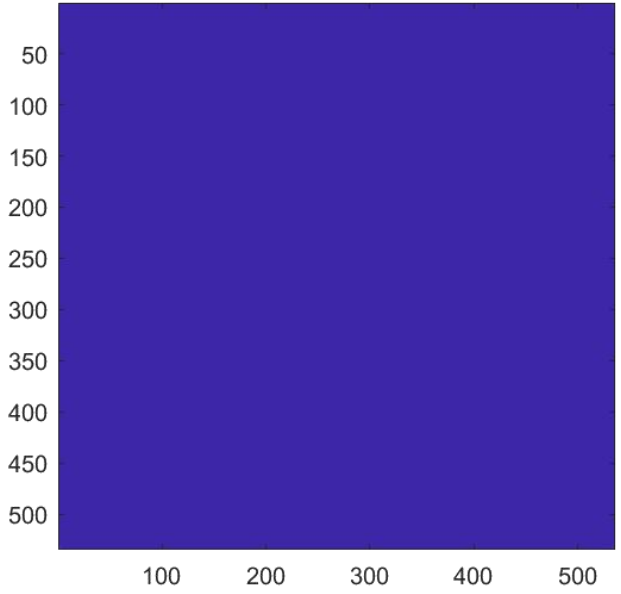
FFT

$\Rightarrow$



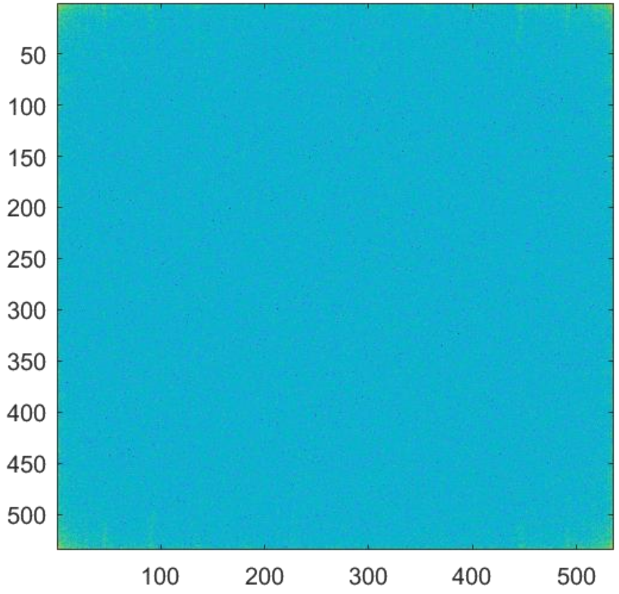
$F(u, v)$

image spectrum



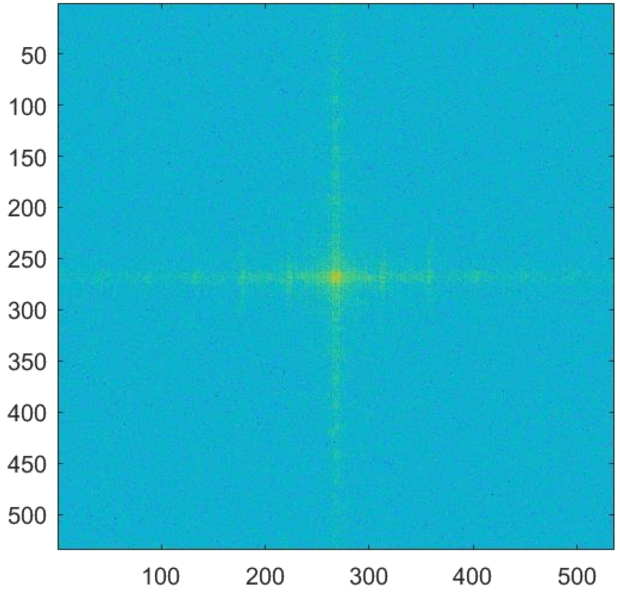
imagesc(abs(fft_image))

logarithmic scaled image spectrum



*imagesc(**log**(abs(fft_image)))*

centered scaled image spectrum

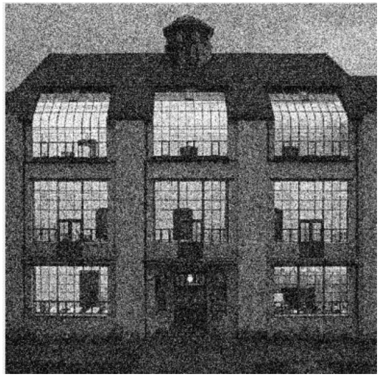


*imagesc(log(abs(**fftshift**(fft_image))))*

Bauhaus-
Universität
Weimar

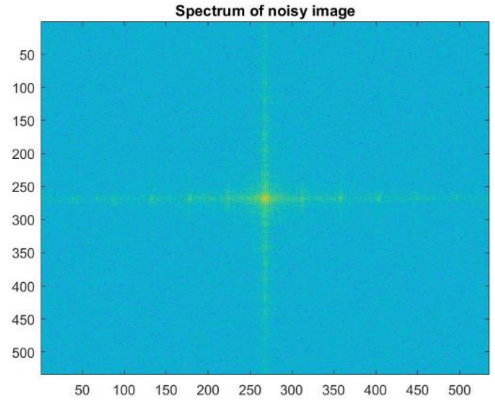
Task A

$f(x, y)$



FFT

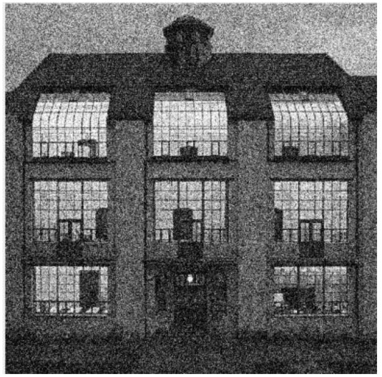
$\Rightarrow$



$F(u, v)$

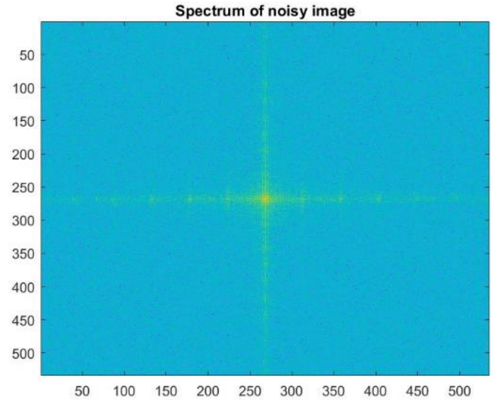
Task A

$f(x, y)$



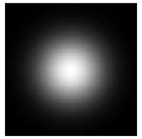
FFT

$\Rightarrow$



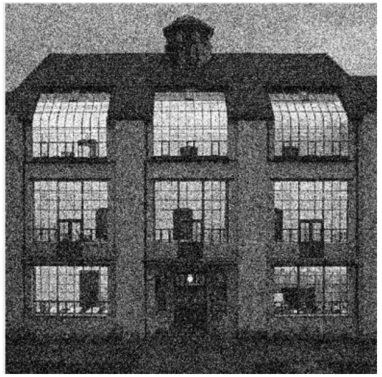
$F(u, v)$

$h(x, y)$

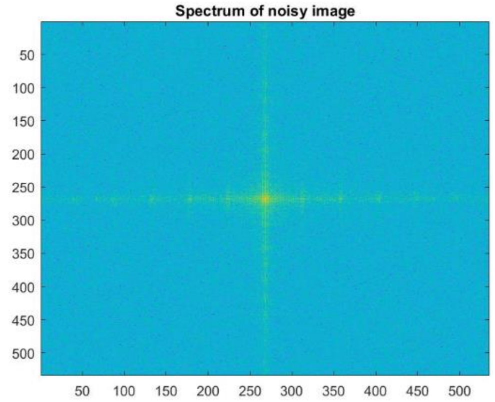


Task A

$f(x,y)$

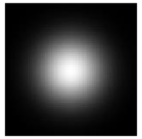


FFT $\Rightarrow$



$F(u,v)$

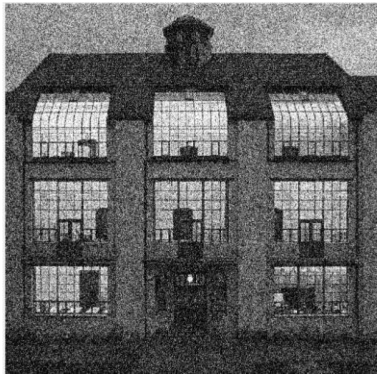
$h(x,y)$



Padding
is necessary

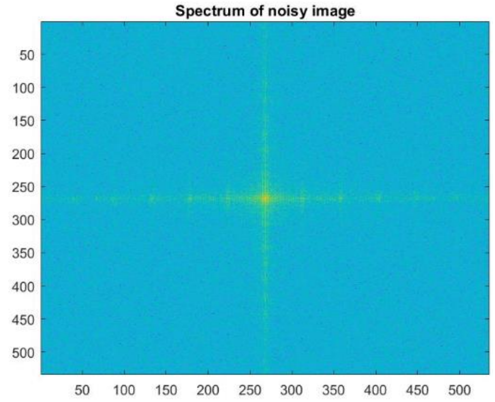
Task A

$f(x, y)$



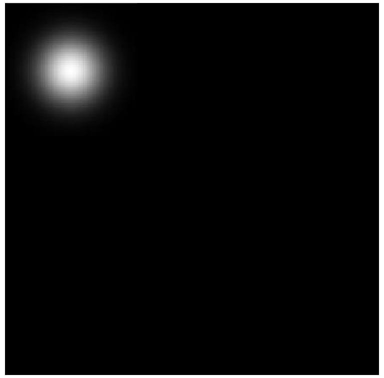
FFT

$\Rightarrow$



$F(u, v)$

$h(x, y)$

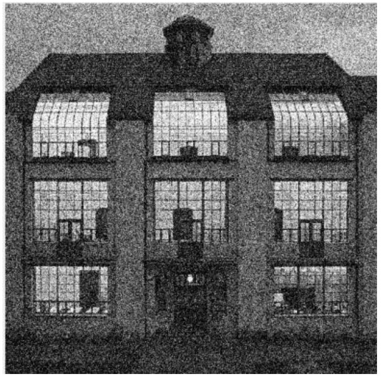


Filter after padding

Bauhaus-Universität Weimar

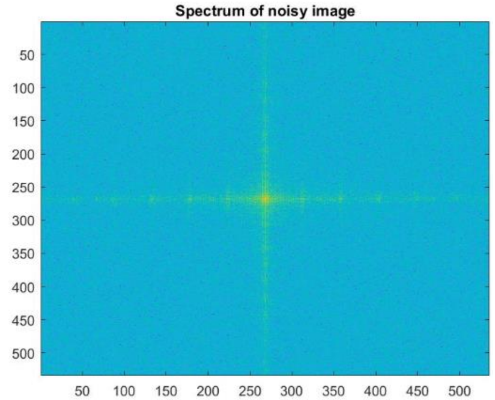
Task A

$f(x,y)$



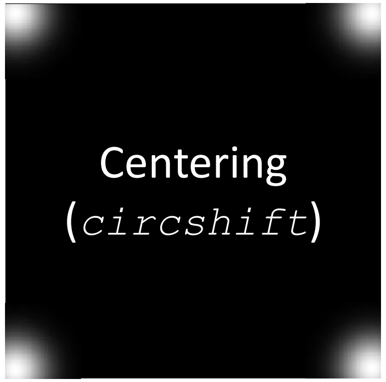
FFT

$\Rightarrow$



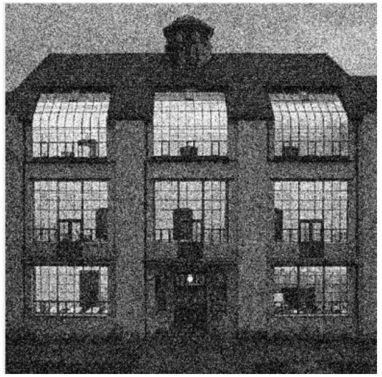
$F(u,v)$

$h(x,y)$

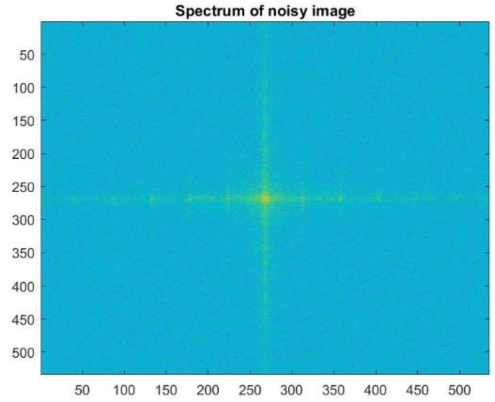


Task A

$f(x, y)$

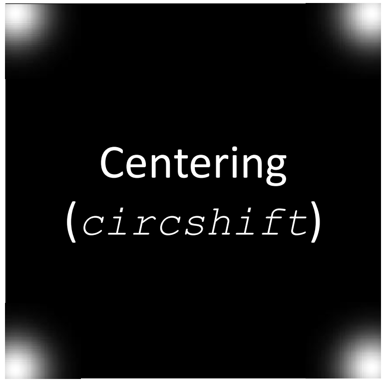


FFT $\Rightarrow$

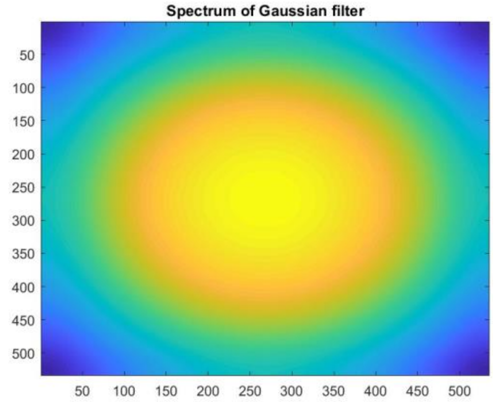


$F(u, v)$

$h(x, y)$



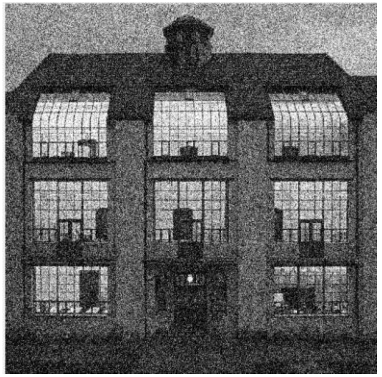
FFT $\Rightarrow$



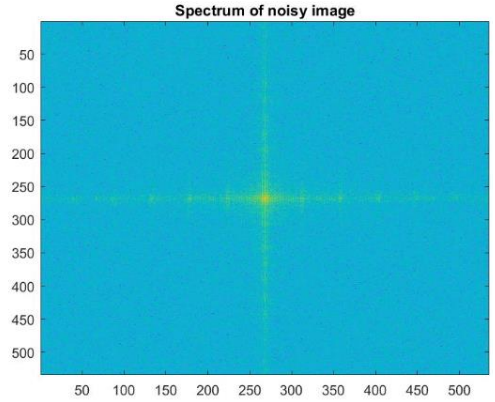
$H(u, v)$

Task A

$f(x,y)$



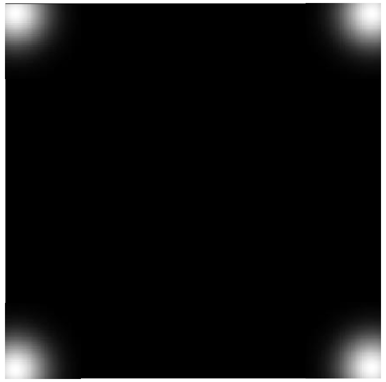
FFT $\Rightarrow$



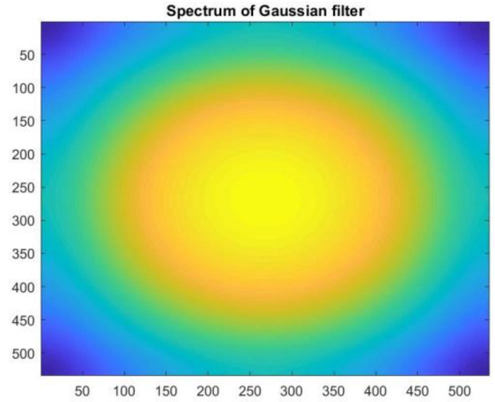
$F(u,v)$

$.*$

$h(x,y)$

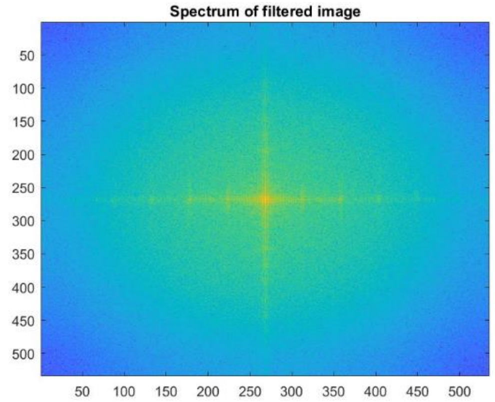


FFT $\Rightarrow$



$H(u,v)$

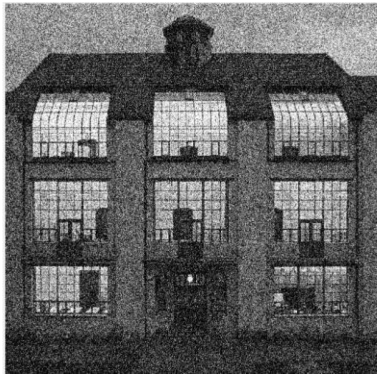
$\Downarrow$



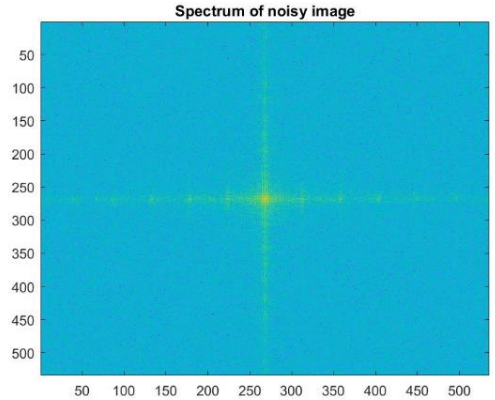
$G(u,v)$

Task A

$f(x,y)$



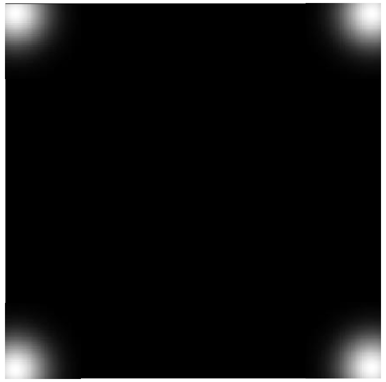
FFT $\Rightarrow$



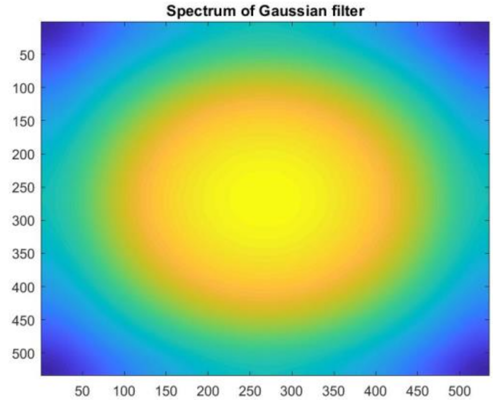
$F(u,v)$

$.*$

$h(x,y)$



FFT $\Rightarrow$

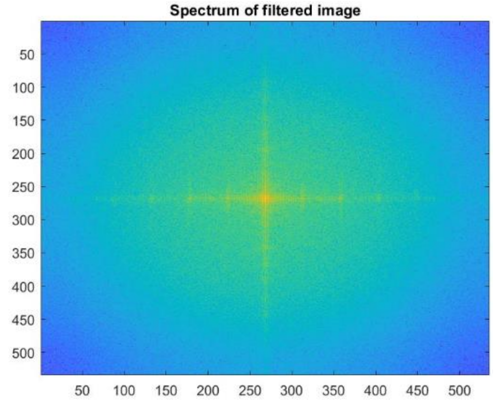


$H(u,v)$

$\Downarrow$

$g(x,y)$

FFT^{-1} $\Leftarrow$



$G(u,v)$

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 (data type `logical`) of the image where 1 represents the object of interest and 0 is background (`graythresh` and `im2bw`)

c. Build a Fourier-descriptor D_f based on the binary mask of b.

- Extraction of boundaries of the binary mask: `bwboundaries`
- 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 by `bwboundaries`

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 (`imshow, hold on` and `plot`)

Task B

Input data

training image

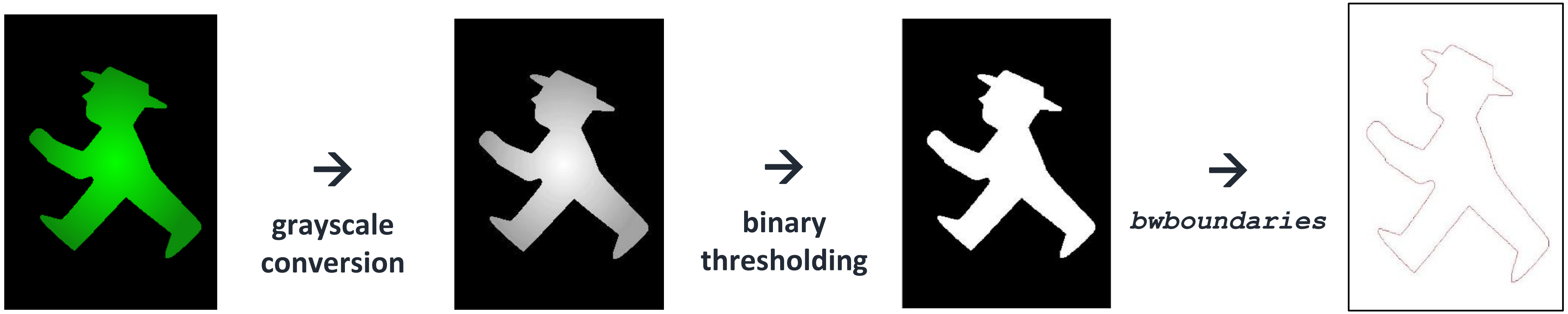
test image 1

test image 2

test image 3

Boundary extraction

Task B



Fourier descriptor

Task B

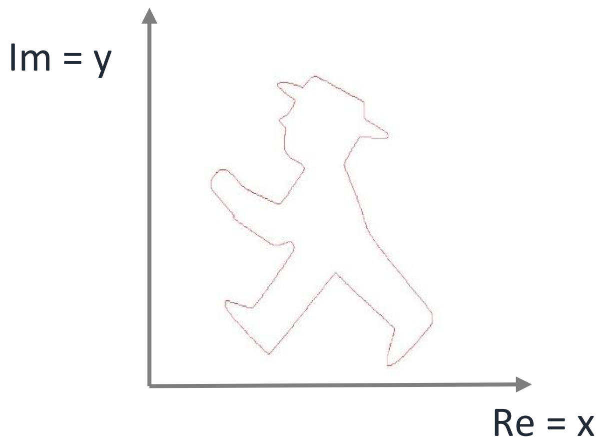
- **Given**: m points representing the boundary of a **closed** region in the image
- Interpret the boundary coordinates (x, y) as complex numbers
- $b = \begin{bmatrix} (y_1, x_1) \\ \vdots \\ (y_m, x_m) \end{bmatrix}$ ($m \times 2$ array: output of `bwboundaries`)
- Build the ***complex vector D***:

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

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

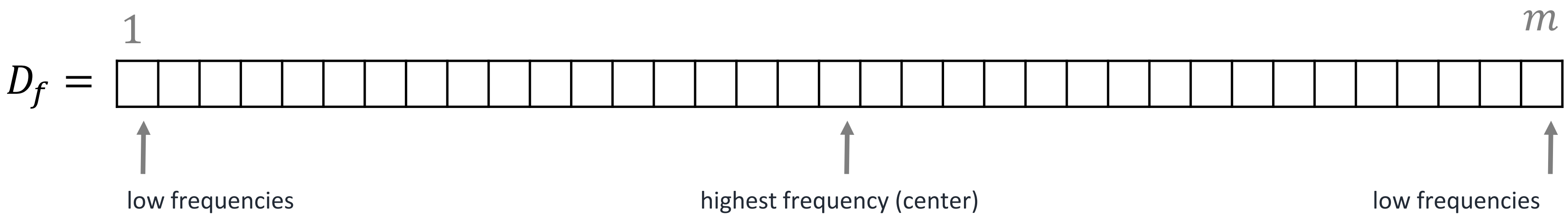
<u>Note:</u> In MATLAB, the built-in **imaginary unit** can be found denoted both as $\boldsymbol{i}$ and as $\boldsymbol{j}$

Don't use $\boldsymbol{j}$ / $\boldsymbol{i}$ as a variable in your code!

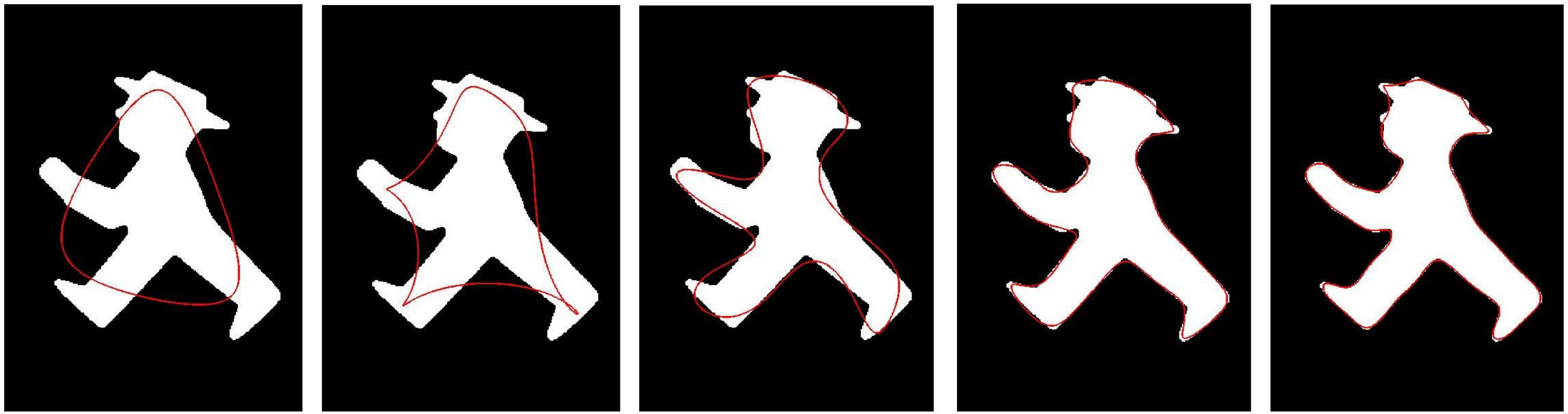


Reducing the number of elements n in D_f → shape generalization

Task B

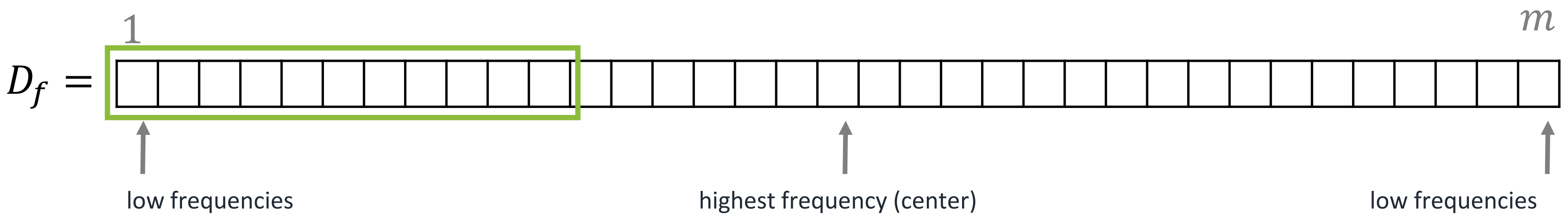


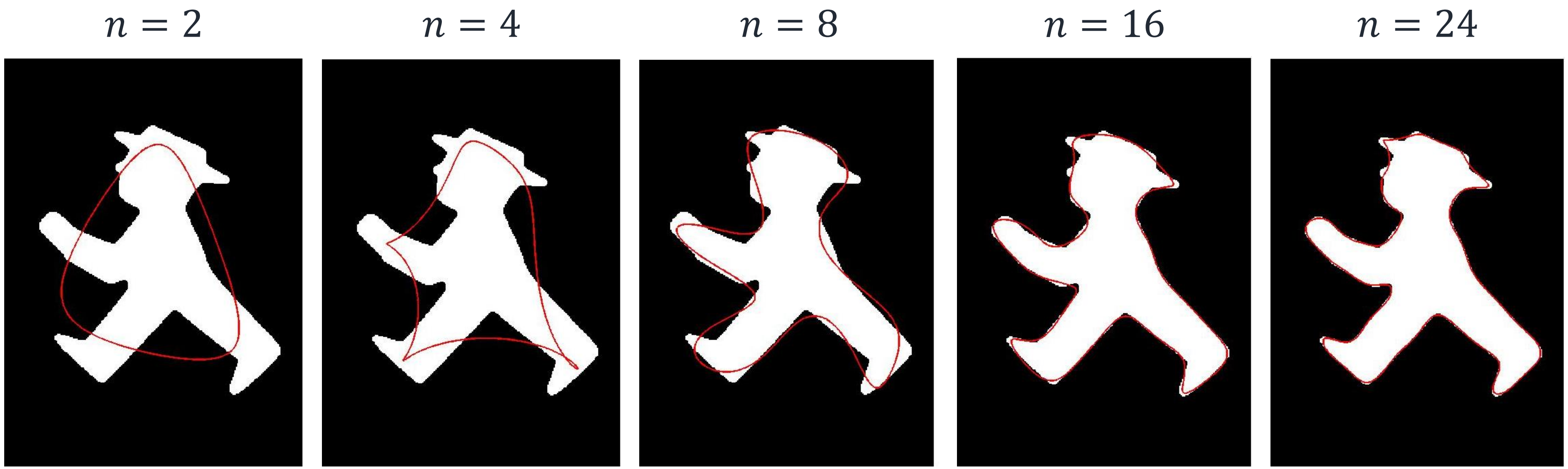
$n = 2$ $n = 4$ $n = 8$ $n = 16$ $n = 24$



Reducing the number of elements n in D_f → shape generalization

Task B





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

Implementation details

Task B

c. Build a Fourier-descriptor based on the binary image of b.

  i. Extraction of boundaries of the binary mask: `bwboundaries`

  ii. Use $n = 24$ elements for the descriptor

  iii. Make it invariant against translation, orientation and scale

Results:

→ the final descriptor $D_{f,train}$ is a $\mathbf{1 \times n}$ **vector**

→ a $\mathbf{1 \times 1}$ **cell** (matlab data type) containing an $m \times 2$ array which represent the m corresponding border pixel coordinates of the found shape (output of `bwboundaries`)

Implementation details

Task B

Output of `bwboundaries`:
$(\mathbf{k} \times \mathbf{1})$ **cell**,

where k is the number of identified closed boundaries

```
My_Cell =

    [682x2 double]
    [686x2 double]
    [654x2 double]
    [685x2 double]
    [154x2 double]
    [168x2 double]
    [328x2 double]
    [335x2 double]
    [377x2 double]
    [332x2 double]
    [ 52x2 double]
    [333x2 double]
    [350x2 double]
    [288x2 double]
    [ 98x2 double]
    [196x2 double]
    [ 57x2 double]
    [ 41x2 double]
    [ 44x2 double]
    [189x2 double]
    [458x2 double]
    [326x2 double]
    [253x2 double]
    [ 84x2 double]
    [ 74x2 double]
    [244x2 double]
    [289x2 double]
    [209x2 double]
    [239x2 double]
    [ 87x2 double]
    [238x2 double]
    [ 84x2 double]
    [ 58x2 double]
    [ 12x2 double]
    [  3x2 double]
    [216x2 double]
```

Access the $\mathbf{34}^{th}$ **array** of boundary coordinates:

```
K>> boundary_points = My_Cell{34}

boundary_points =

    51   886
    50   887
    49   888
    50   888
    51   888
    52   888
    53   888
    54   888
    54   887
    53   887
    52   887
    51   886
```

Implementation details

Task B

d. Apply steps a.-c. on images *test1B.jpg, test2B.jpg* and *test3B.jpg* in order to identify all potential objects

Results **for each image**:

→ **Descriptors**: $\mathrm{k} \times n$ array, where k is the number of identified boundaries

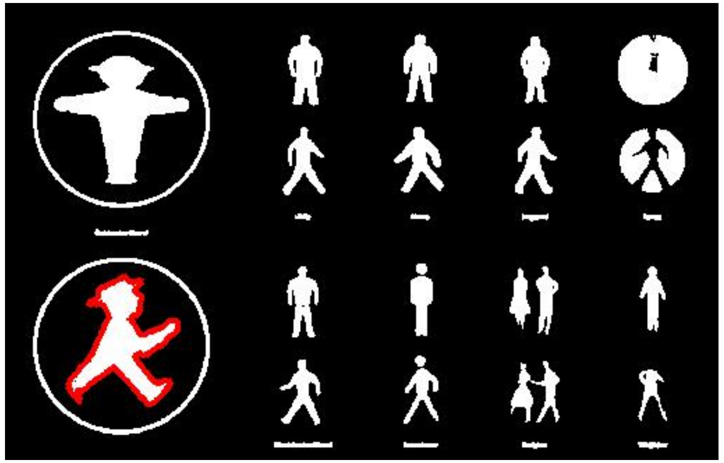
→ **Boundaries**: $\mathrm{k} \times 1$ cell containing k $(m \times 2)$ arrays which represent the corresponding border **pixel coordinates** of the k found shapes

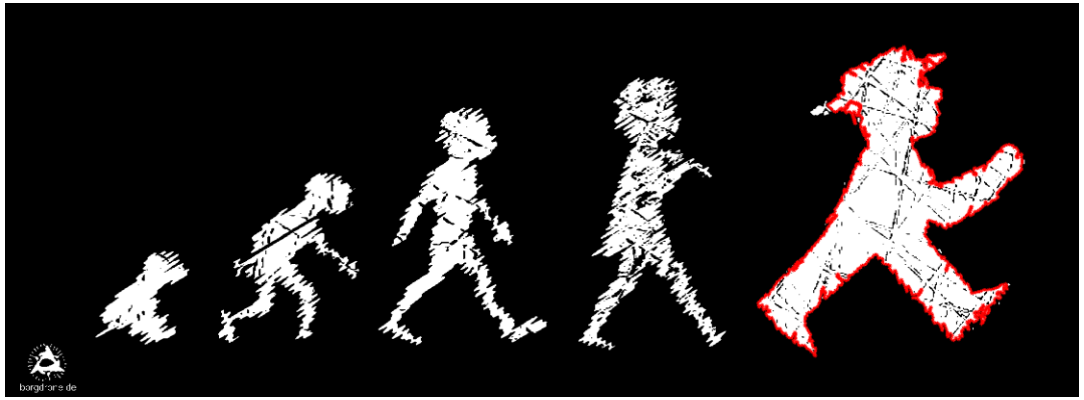
Bauhaus-Universität Weimar

Expected results

Task B

training image





Bauhaus-
Universität
Weimar