Bauhaus-Universität Weimar

Image Analysis and Object Recognition

Assignment 4

Image filtering in frequency domain
&
Shape recognition using Fourier descriptors

SS 2018

(Course notes for internal use only!)

Assignment 3 – sample solution 1/2

```
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
```

Assignment 3 – sample solution 2/2

```
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);                            % ind_r = rho + d + 1
    ind_t = find(t == theta);                           % ind_t = theta + 91
    H(ind_r, ind_t) = H(ind_r, ind_t) + 1;          % vote for theta and rho
end
```

Bauhaus-Universität Weimar

Assignment 4

A: Image filtering in frequency domain

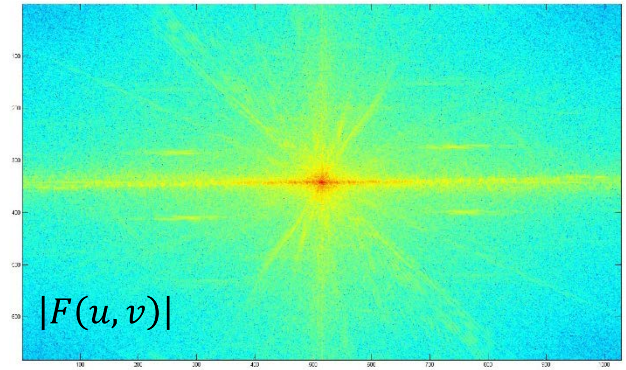
&

B: Shape recognition using Fourier descriptors

Smoothing in frequency domain

$f(x,y)$

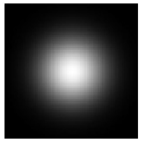
FFT $\Rightarrow$



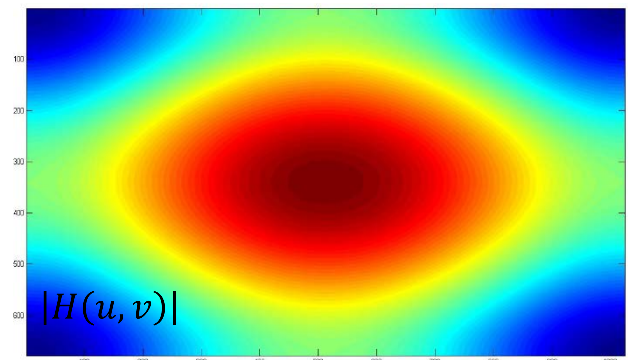
$F(u,v)$

$.*$

$h(x,y)$



Padding + FFT $\Rightarrow$

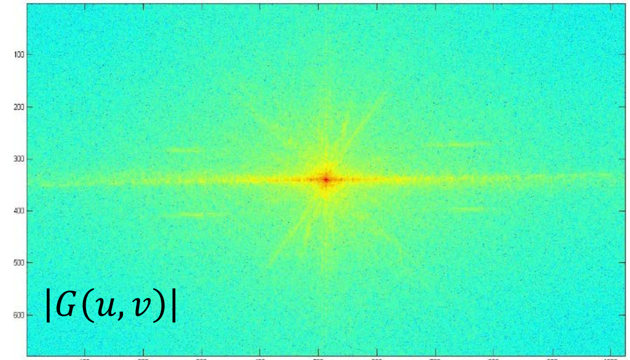


$H(u,v)$

$\Downarrow$

$g(x,y)$

FFT^{-1} $\Leftarrow$



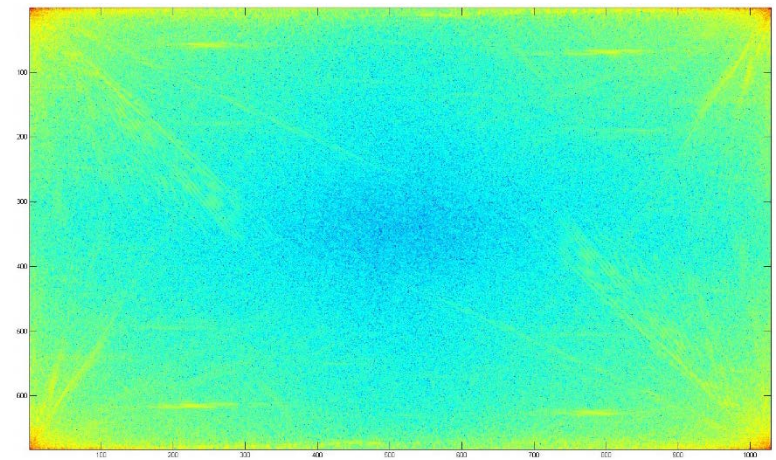
$G(u,v)$

Fast Fourier transform FFT

Input image

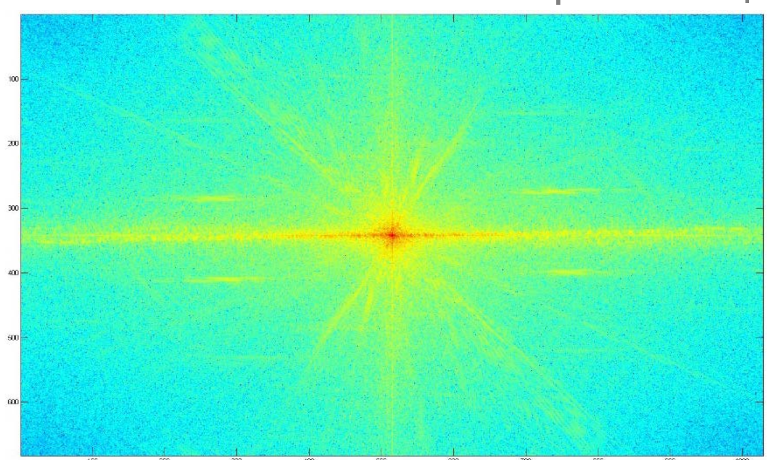
→ *fft2*

Logarithmic scaled spectrum $|F(u,v)|$



imagesc(log(abs(fft_image)))

Logarithmic scaled centered spectrum $|F(u,v)|$



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

Task A: Noise removal

a. Read the input image *taskA.png* and convert it to a grayscale image from data type double with 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. Convolve 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

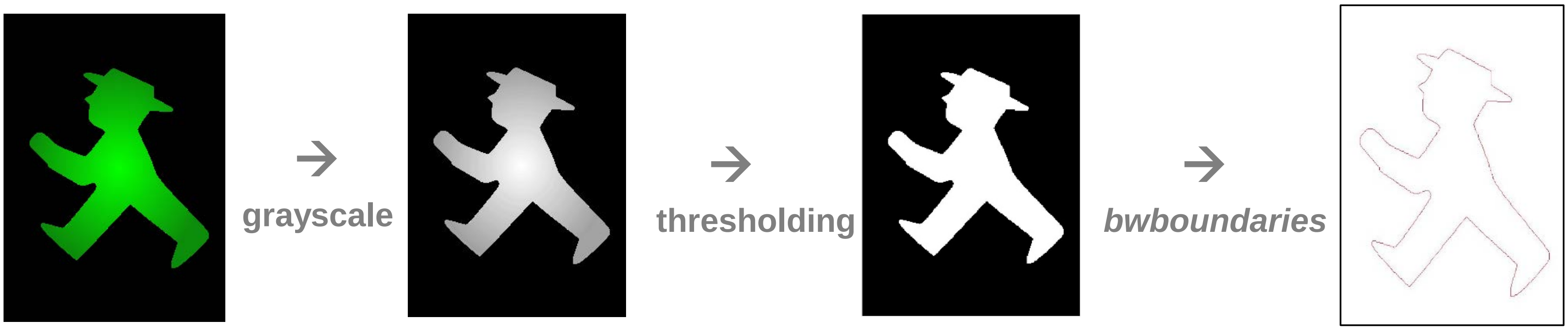
d. Plot the logarithmic centered image spectra of the noisy image, of the (padded) Gaussian filter and of the filtered image

Fourier descriptors

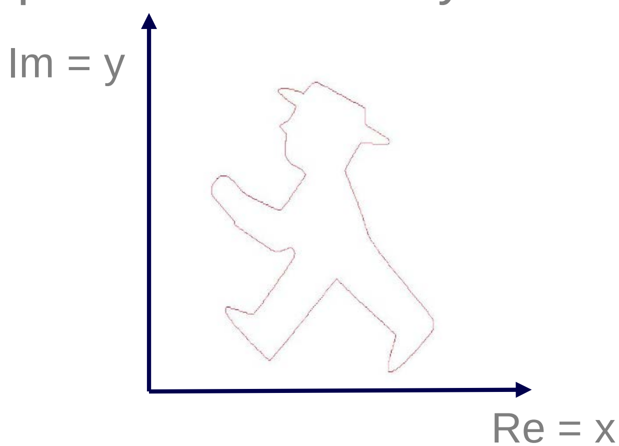
- **Given:** Image which represents a shape of interest
- **Task:** Find this shape in other images automatically

Fourier descriptors

- **Given**: m points representing the boundary of a **closed** region in the image



- Interprete the boundary coordinates as complex numbers



Fourier descriptors

Hint: Building the complex vector in Matlab

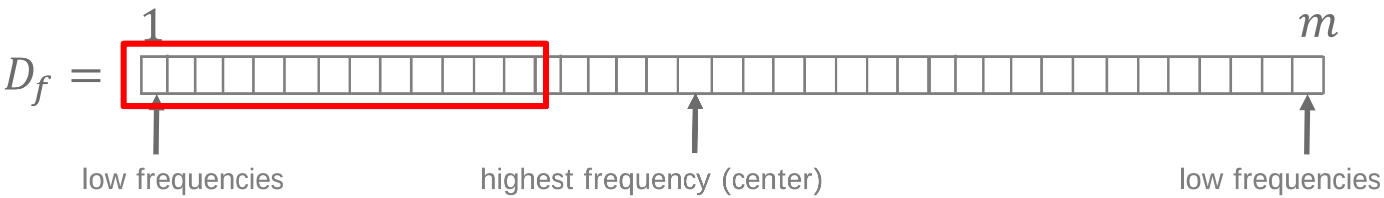
- Interprete the boundary coordinates (x, y) as complex numbers
- $b = \begin{bmatrix} (y_1, x_1) \\ \vdots \\ (y_m, x_m) \end{bmatrix}$ ($m \times 2$ array: output of *bwboundaries*)
- Building the complex vector D:

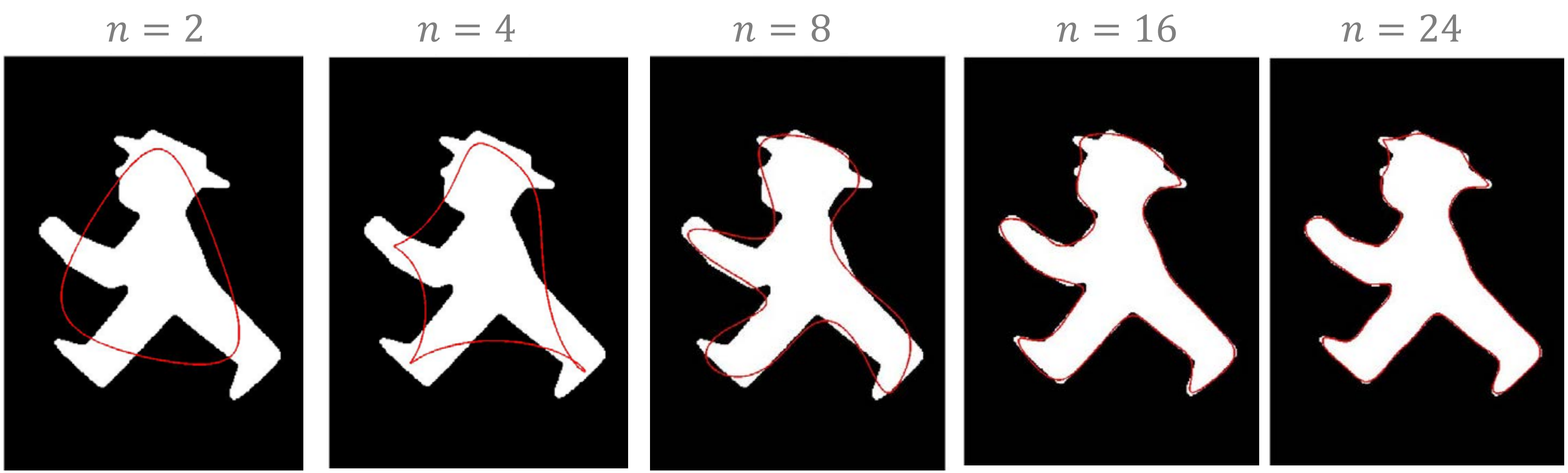
$$D = b(:,2) + \boldsymbol{j} * b(:,1);$$

- Don‘t use ***j*** as variable in your code!

Fourier descriptor manipulation

- Number of elements n in $D_f \rightarrow$ generalization





- Reducing elements of D_f: Extract the first n elements (low frequency values) of D_f and forget the rest

Task B 1/4

a. Read the image *trainB.png* and convert it to a grayscale image from data type double with 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 (`graythresh`, `im2bw`)

Task B 2/4

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 $1 \times n$ vector
- → A 1×1 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`)

Matlab cells

Output of `bwboundaries`: $(k \times 1)$ 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 34th 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
```

Task B 3/4

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

→ Results for each image:

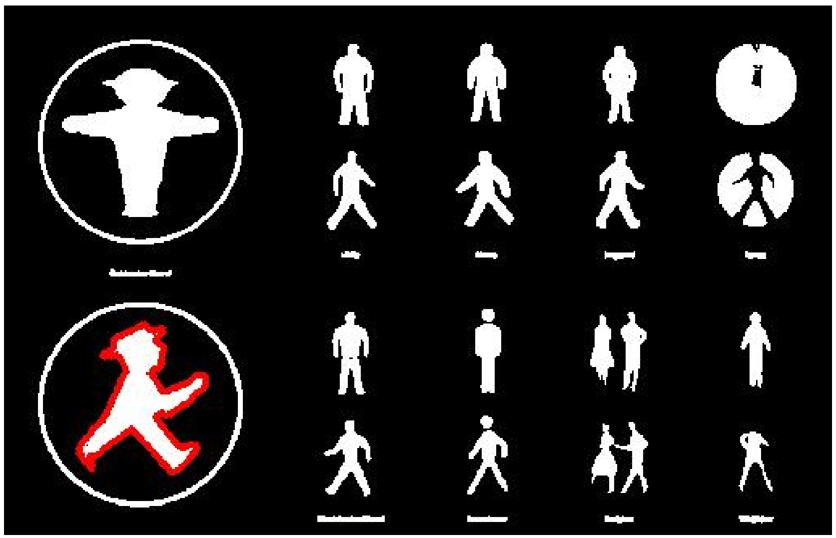
- → **Descriptors**: $\mathrm{k} \times n$ array, where k is the number of identified boundaries
- → **Boundaries**: $\mathrm{k} \times 1$ cell containing k $(m \times 2)$ arrays which represent the corresponding border **pixel coordinates** of the k found shapes

e. Identify the searched object by **comparison** of Fourier-descriptor $D_{f,train}$ (result of c) with all identified descriptors of the two test images $D_{f,test}$ (result of d). Use the **Euclidean distance** of the element-wise differences, e.g. if

$$\mathtt{norm}\left(D_{f,train} - D_{f,test}\right) < 0.06$$

Task B 4/4

f. **Plot** the identified **boundaries** on the masks of the test images in order to validate the results (*imshow, hold on, plot*).



- Use the pixel coordinates of the shapes for plotting (result of *bwboundaries*)

Thank you!