Uniwersytet Rzeszowski

Wydział Nauk Ścisłych i Technicznych

Instytut Informatyki

BIOMETRYCZNE SYSTEMY ZABEZPIECZEŃ

INSTRUKCJA DO ĆWICZEŃ LABORATORYJNYCH

TREŚCI KSZTAŁCENIA: TRANSFORMATA FOURIERA, CZĘSTOTLIWOŚCIOWA ANALIZA OBRAZU

Spis treści

1. Cele laboratorium

Transformata Fouriera (TF) to jedno z podstawowych narzędzi w analizie sygnałów i obrazów, które pozwala na przejście z dziedziny przestrzennej do dziedziny częstotliwościowej. Dzięki temu można analizować, jak różne składowe częstotliwościowe wpływają na obraz i stosować różne techniki filtracji. W laboratorium przenalizowane będą informacje związane z obliczaniem transformaty Fouriera obrazu, wizualizacji jego widma częstotliwościowego oraz stosowane operacje filtracji.

2. Wprowadzenie

Interaktywny wstęp do transformaty Fouriera można znaleźć pod adresem https://www.jezzamon.com/fourier/pl.html lub https://github.com/Jezzamonn/fourier (dostęp: 18.02.2025r). Szczegółowe informacje odnośnie transformaty Fouriera dostępne są pod adresem https://home.agh.edu.pl/~zobmat/2020/II_mich_mar/index.html (dostęp: 18.02.2025r).

Transformata Fouriera 2D

Dyskretna transformata Fouriera (DFT) dla obrazu dwuwymiarowego $f(x, y)$ jest zdefiniowana jako:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{1}$$

gdzie: $f(x, y)$ – obraz w dziedzinie przestrzennej, $F(u, v)$– reprezentacja częstotliwościowa obrazu, u, v– współrzędne w dziedzinie częstotliwości, M, N – rozmiary obrazu.

Transformata odwrotna jest dana wzorem:

$$\mathrm{f}(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{2}$$

W MATLAB-ie DFT jest obliczana za pomocą funkcji fft2(), a odwrotna transformata Fouriera za pomocą ifft2(). Wywodzący się z francuskiej szkoły matematyki Jean Baptiste Fourier zaproponował, że dowolny periodyczny sygnał $x(t)$ można przedstawić przy pomocy (spełniający również twierdzenie Dirichleta) skończonego szeregu sumy ortogonalnych funkcji sinus i cosinus:

$$\mathrm{x}(t) = a_0 + \sum_{n=1}^{\infty} (a_n cos\, n\omega_0 t + b_n sin\, n\omega_0 t) \tag{3}$$

Funkcja $x(t)$ jest funkcją okresową o okresie $T = 2\pi/\omega 0$.

Własności współczynników:

1. Jeśli $x(t)$ parzysta to $b_n = 0$

2. Jeśli $x(t)$ nieparzysta to $a_n = 0$

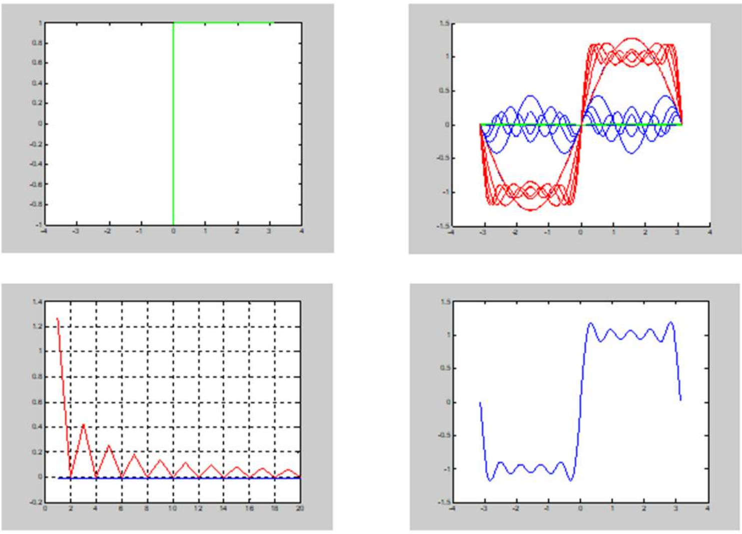
3. Jeśli $x(t + T/2) = -x(t)$ to znikają parzyste harmoniczne

Gdzie poszczególne składowe (amplitudy) szeregu- harmoniczne mogą być obliczone następująco:

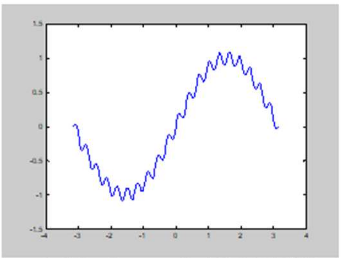
$$a_0 = \frac{1}{T} \int_0^T x(t)dt \tag{4}$$

$$a_n = \frac{2}{T} \int_0^T x(t) cosn\omega_0 tdt \tag{5}$$

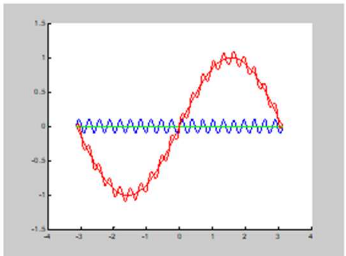
$$b_n = \frac{2}{T} \int_0^T x(t) sinn\omega_0 tdt \tag{6}$$



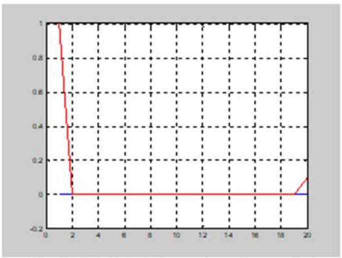
Rys. 1. Przykłady rozkładu w szereg wybranych funkcji.



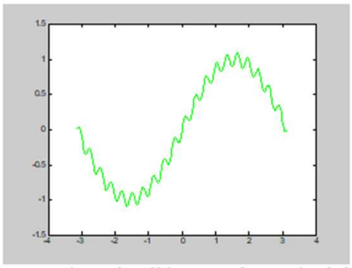
Funkcja podstawowa x(t)=sin(t)+0.1*sin(20*t)
Długość wektora x to 628 liczb typu double



Kolejne przybliżenia sygnału wejściowego dla $n \in \langle 1, \ldots, 20 \rangle$



Amplitudy kolejnych harmonicznych parzystych i nieparzystych. Dlaczego nieparzyste mają wartość 0?



Restytuowanie sygnału wejściowego z wykorzystaniem dwóch harmonicznych 1 i 20 to kompresja ~300

Rys. 2. Przykłady rozkładu w szereg mechanizm kompresji-filtracji

3. Zadania do samodzielnego rozwiązania

Zadanie 1.

W oparciu o poniższy opis struktury algorytmu związanego z rozkładem w szereg Fouriera zaproponuj jego implementację. Struktura algorytmu:

- Inicjalizacja – zamknięcie otwartych okien, czyszczenie pamięci, ustawienie parametrów.
- Generowanie funkcji wejściowej – tworzenie sygnału do analizy.
- Rozkład na harmoniczne – obliczanie współczynników Fouriera.
- Sumowanie składowych – rekonstrukcja funkcji.
- Wizualizacja wyników – rysowanie wykresów poszczególnych harmonicznych i sygnału końcowego.

Poniżej przedstawiono schemat dla powyższej struktury algorytmu:

Rys. 3. Struktura algorytmu Transtormaty Fouriera

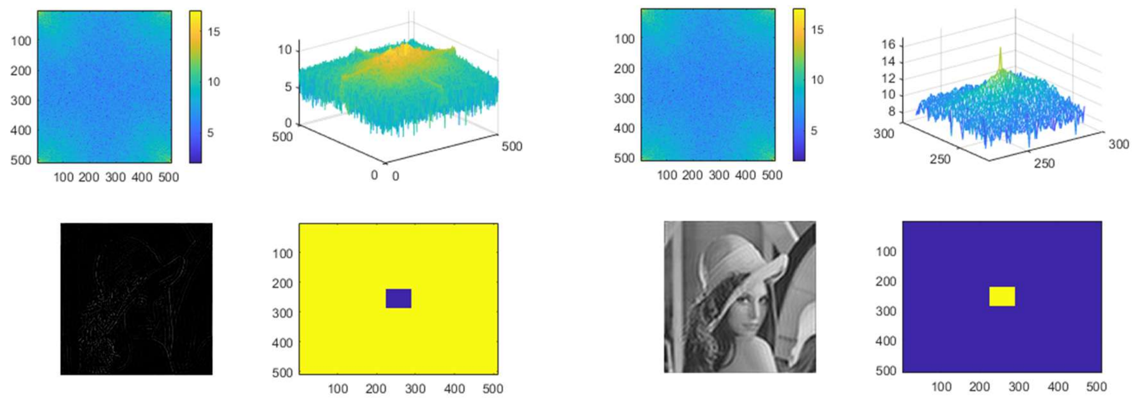
```
%% Fourier Series Decomposition
% ============================================

close all; clear all; clc;

% Basis description
roz = 5; % Number of harmonics
a = -180:1:180;          % Angular measure in degrees
t = a * pi / 180;        % Convert to radians
Tmax = t(end) - t(1);    % Period length
krok = t(2) - t(1);      % Step size

% Define the different signal types
y = sin(t) + 0.1*sin(20*t); % Custom signal to analyze
y = (t > 0)*2 - 1;          % Try other signals for fun
% y = t;
y=sin(t)+0.2*cos(5*t)+0.1*sin(10*t);

% Plot the original signal
figure;
plot(t, y, 'g');
title('Original Signal');
grid on;

%% Fourier Series Coefficients
% ============================================
Ao = (1/Tmax) * sum(y .* krok);
A = zeros(roz, 1);
B = zeros(roz, 1);

for j = 1:roz
    ac = 0; bs = 0;
    for i = 1:length(t)
```

```
        ac = ac + krok * (2/Tmax) * y(i) * cos(j * 2*pi/Tmax * t(i));
        bs = bs + krok * (2/Tmax) * y(i) * sin(j * 2*pi/Tmax * t(i));
    end
    A(j) = ac;
    B(j) = bs;
end

%% Reconstruct Signal from Harmonics
% ============================================
Skc = zeros(roz, length(t));
Sks = zeros(roz, length(t));

for i = 1:roz
    Skc(i,:) = A(i) * cos(i * 2*pi/Tmax * t);
    Sks(i,:) = B(i) * sin(i * 2*pi/Tmax * t);
end

% Initialize reconstructed signal with Ao/2
W = zeros(1, length(t)) + Ao / 2;

% Sum of harmonics
figure;
hold on;
for i = 1:roz
    W = W + Skc(i,:) + Sks(i,:);
    plot(t, Sks(i,:), 'b', t, Skc(i,:), 'g', t, W, 'r');
end
title('Harmonic Components and Signal Reconstruction');
xlabel('t');
ylabel('Amplitude');
grid on;
hold off;

%% Final Reconstructed Signal
figure;
plot(t, W, 'r', 'LineWidth', 1.5);
title('Final Reconstructed Signal from Fourier Series');
xlabel('t');
ylabel('Amplitude');
grid on;

%% Plot Coefficients
figure;
subplot(2,1,1);
% Stem plot or liner mode
% stem(1:roz, A, 'g', 'filled');
% Or linear mode
plot(1:roz,A, 'g')
title('Cosine Coefficients A_n');
xlabel('Harmonic n');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
stem(1:roz, B, 'r', 'filled');
title('Sine Coefficients B_n');
xlabel('Harmonic n');
ylabel('Amplitude');
grid on;

%% Optional: Magnitude Spectrum
figure;
% stem(1:roz, sqrt(A.^2 + B.^2), 'k', 'filled');
plot(1:roz, sqrt(A.^2 + B.^2), 'r')
title('Fourier Magnitude Spectrum |C_n|');
xlabel('Harmonic n');
```

```
ylabel('Amplitude');
grid on;
```

Zadanie 2.

Wykorzystując dwuwymiarową transformatę Fouriera zaproponuj jego implementację oraz wykonaj eksperymenty z filtracją środkowo przepustową obrazu tęczówki oka podanej przez prowadzącego. Poniżej przedstawiono przykładowe rezultaty uzyskane dla obrazu Lena z wykorzystaniem maski prostokątnej.



(a) (b)

Rys. 4. Przykłady filtracji górnoprzepustowej (a) i dolnoprzepustowej (b)

```
% Step 1: Load and preprocess normalized iris image
close all; clear; clc;

% Load a normalized (polar) iris image
I = imread('teczowka_2.png');  % Assumed grayscale, 64x512 or similar
if size(I,3) == 3
    I = rgb2gray(I);
end
I = im2double(I);

% Display original
figure; imshow(I, []); title('Original Normalized Iris');

% Step 2: Compute 2D FFT and shift frequency spectrum
F = fft2(I);
F_shifted = fftshift(F);
magF = log(1 + abs(F_shifted));

figure; imshow(magF, []); title('Frequency Spectrum (log scale)');

% Step 3: Design bandpass filter
[rows, cols] = size(I);
[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));
% The cone like mask shape
D = sqrt(u.^2 + v.^2);

% Define cutoff frequencies
D_low = 5;   % Remove very low frequency (lighting, iris boundary drift)
```

```
D_high = 25; % Remove high freq noise (salt & pepper, reflections)

% Create ideal bandpass mask
H = double(D > D_low & D < D_high);

% Optional: visualize mask
figure; imshow(H, []); title('Bandpass Filter for Iris');

% Step 4: Apply filter
F_filtered = F_shifted .* H;

% Step 5: Inverse FFT
I_filtered = real(ifft2(ifftshift(F_filtered)));
I_filtered = mat2gray(I_filtered);

% Step 6: Compare results
figure;
subplot(1,2,1); imshow(I, []); title('Original Normalized Iris');
subplot(1,2,2); imshow(I_filtered, []); title('Enhanced Iris (Fourier Filtered)');

% Assesing space shape of the iris
figure
mesh(I_filtered)
title('Morphology of the enrolled iris in the polar plane')


% Frequency fingerprint filtering
%====================================================
% Step 1: Load fingerprint image
close all; clear; clc;
I = imread('finger1.jpg');  % Use grayscale fingerprint
if size(I,3) == 3
    I = rgb2gray(I);
end
I = im2double(I);

% Display original
figure; imshow(I); title('Original Fingerprint');

% Step 2: Compute 2D FFT and shift zero freq to center
F = fft2(I);
F_shifted = fftshift(F);
magnitude = log(1 + abs(F_shifted));
% Display frequency spectrum
figure; imshow(magnitude, []); title('Frequency Spectrum (log scale)');
% Step 3: Create bandpass filter mask
[rows, cols] = size(I);
[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));
D = sqrt(u.^2 + v.^2);
% Define cutoff frequencies
D_low = 10;    % Suppress low frequencies (lighting, background)
D_high = 60;   % Suppress high freq noise

% Ideal bandpass filter
H = double(D > D_low & D < D_high);
% Optional: visualize filter
figure; imshow(H, []); title('Bandpass Filter Mask');
% Step 4: Apply filter in frequency domain
F_filtered = F_shifted .* H;
% Step 5: Inverse FFT to get enhanced image
I_filtered = real(ifft2(ifftshift(F_filtered)));
% Normalize for display
I_filtered = mat2gray(I_filtered);
% Show enhanced result
figure;
subplot(1,2,1); imshow(I); title('Original');
```

```
subplot(1,2,2); imshow(I_filtered); title('Enhanced via Fourier Filtering');
```

Zadanie 3.

Przetestuj przedstawiony powyżej mechanizm filtracji częstotliwościowej na obrazie daktyloskopijnym.

```
% Frequency fingerprint filtering
%===================================================

% Step 1: Load fingerprint image
close all; clear; clc;
I = imread('finger1.jpg');  % Use grayscale fingerprint
if size(I,3) == 3
    I = rgb2gray(I);
end
I = im2double(I);

% Display original
figure; imshow(I); title('Original Fingerprint');

% Step 2: Compute 2D FFT and shift zero freq to center
F = fft2(I);
F_shifted = fftshift(F);
magnitude = log(1 + abs(F_shifted));

% Display frequency spectrum
figure; imshow(magnitude, []); title('Frequency Spectrum (log scale)');

% Step 3: Create bandpass filter mask
[rows, cols] = size(I);
[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));
D = sqrt(u.^2 + v.^2);

% Define cutoff frequencies
D_low = 10;    % Suppress low frequencies (lighting, background)
D_high = 60;   % Suppress high freq noise

% Ideal bandpass filter
H = double(D > D_low & D < D_high);

% Optional: visualize filter
figure; imshow(H, []); title('Bandpass Filter Mask');

% Step 4: Apply filter in frequency domain
F_filtered = F_shifted .* H;

% Step 5: Inverse FFT to get enhanced image
I_filtered = real(ifft2(ifftshift(F_filtered)));

% Normalize for display
I_filtered = mat2gray(I_filtered);

% Show enhanced result
figure;
subplot(1,2,1); imshow(I); title('Original');
subplot(1,2,2); imshow(I_filtered); title('Enhanced via Fourier Filtering');
```