Labolatorium PTI – ćwiczenie 3, przykład 3

Ekstrakcja i kompresja informacji

Oczyszczanie i kompresja sygnałów

1 PRZETWARZANIE WSTĘPNE: Filtrowanie i oczyszczanie

1.1 Opóźnienie sygnału po filtrowaniu

Rozważany przykład to filtr o skończonej odpowiedzi impulsowej. Filtr ten wprowadza stałe opóźnienie. Może ono być skompensowane przez przesunięcie sygnału w czasie. W poniższym przykładzie użyjemy filtru dolnoprzepustowego o następującej charakterystyce:

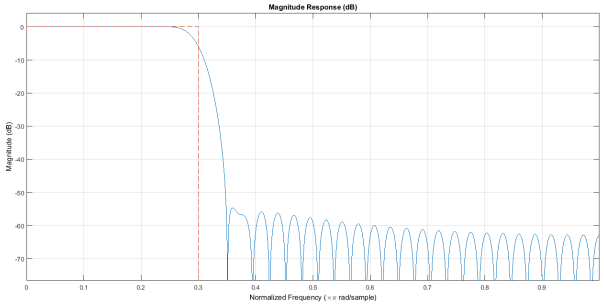

Figure 1: Charakterystyka filtru

Następnie użyjemy go do odfiltrowania sygnału i porównamy wyniki:

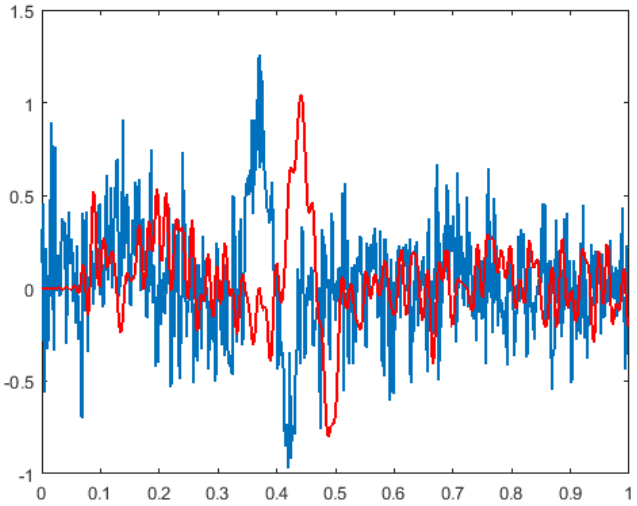

Figure 2: Sygnał oryginalny i opóźnienie

Jak widać powstało stałe opóźnienie sygnału. Używamy wyliczonej wartości opóźnienia by przesunąć syngał i otrzymujemy:

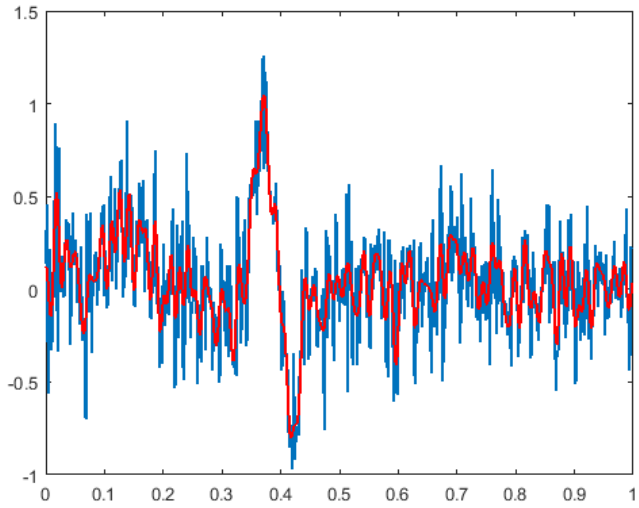

Figure 3: Skorygowane

```
Fs = 500;                          % sample rate in Hz
N = 500;                           % number of signal samples
rng default;
x = ecg(N)'+0.25*randn(N,1);       % noisy waveform
t = (0:N-1)/Fs;                    % time vector

% Design a 70th order lowpass FIR filter with cutoff frequency of
    75 Hz.
Fnorm = 75/(Fs/2);                 % Normalized frequency
df = designfilt('lowpassfir','FilterOrder',70,'CutoffFrequency',
    Fnorm);

hfvt = fvtool(df);

D = mean(grpdelay(df)) % filter delay in samples

% y = filter(df,x);

% figure
% plot(t,x,t,y,'r','linewidth',1.5);

 y = filter(df,[x; zeros(D,1)]); % Append D zeros to the input data
 y = y(D+1:end);                  % Shift data to compensate for
    delay
%
 figure
 plot(t,x,t,y,'r','linewidth',1.5);
```

przyklad1.m

Figure 4: Listing kodu

1.2 Oczyszczanie sygnału

W rozważanym przykładzie występują piki zniekształcające sygnał.

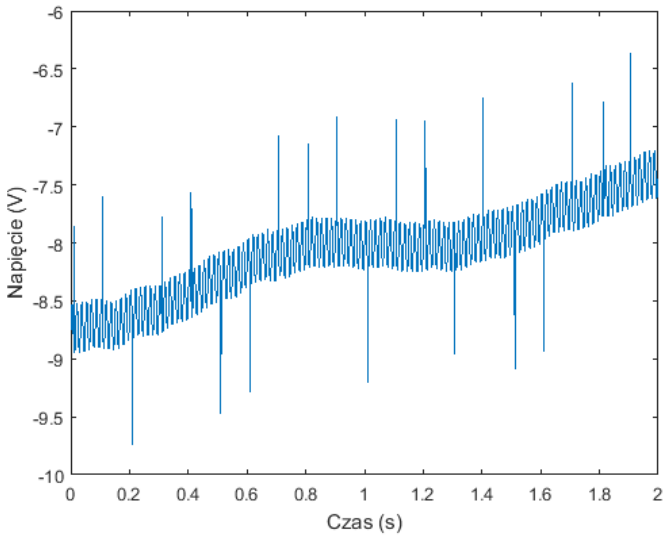

Figure 5: Sygnał przed odfiltrowaniem

Do ich usunięcia użyjemy prostego filtru medianowego. Zastępuje on wartości bardzo odstające przez medianę wartości sąsiednich punktów.

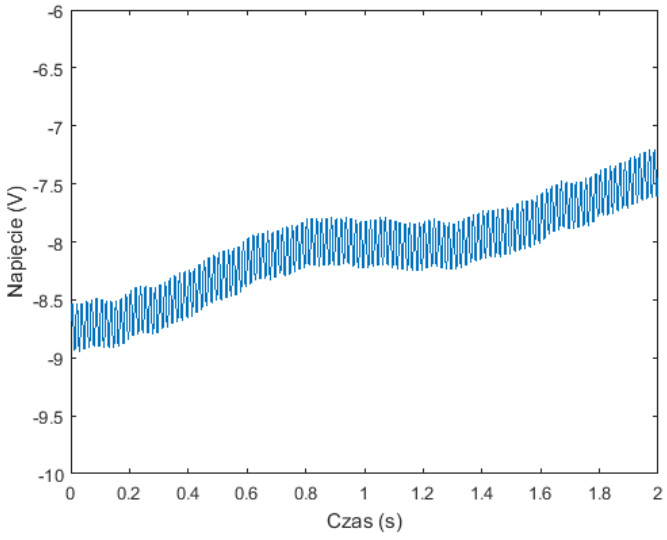

Figure 6: Sygnał po filtrowaniu

```
load openloop60hertz

fs = 1000;
t = (0:numel(openLoopVoltage) - 1)/fs;

rng default

spikeSignal = zeros(size(openLoopVoltage));
spks = 10:100:1990;
spikeSignal(spks+round(2*randn(size(spks)))) = sign(randn(size(spks
    )));

noisyLoopVoltage = openLoopVoltage + spikeSignal;

plot(t,noisyLoopVoltage)

xlabel('Czas (s)')
ylabel('Napiecie (V)')
yax = ylim;


medfiltLoopVoltage = medfilt1(noisyLoopVoltage,3);

plot(t,medfiltLoopVoltage)

xlabel('Czas (s)')
ylabel('Napiecie (V)')
ylim(yax)
```

przyklad2.m

Figure 7: Listing kodu

1 REPREZENTACJA SYGNAŁÓW: Kompresja obrazu z użyciem transformacji falkowej

1.1 Wstęp

Algorytm kompresji

- Dekompozycja obrazu
- Wycięcie współczynników reprezentacji, o wartościach poniżej ustalonego poziomu.
- Rekonstrukcja obrazu

W kroku drugim, wycinanie współczynników może być przeprowadzone na dwa sposoby. Pierwszy z nich zakłada wprowadzanie globalnej wartości, poniżej której współczynniki będą wycinane. Drugi to oddzielne ustalenie poziomów dla detali pionowych, poziomych oraz ukośnych.

W celu oceny kompresji posłużymy się wartościami

PERF0 - procent wyciętych współczynników do wszystkich
PERFL - norma współczynników pozostawionych podzielona przez normę wszystkich

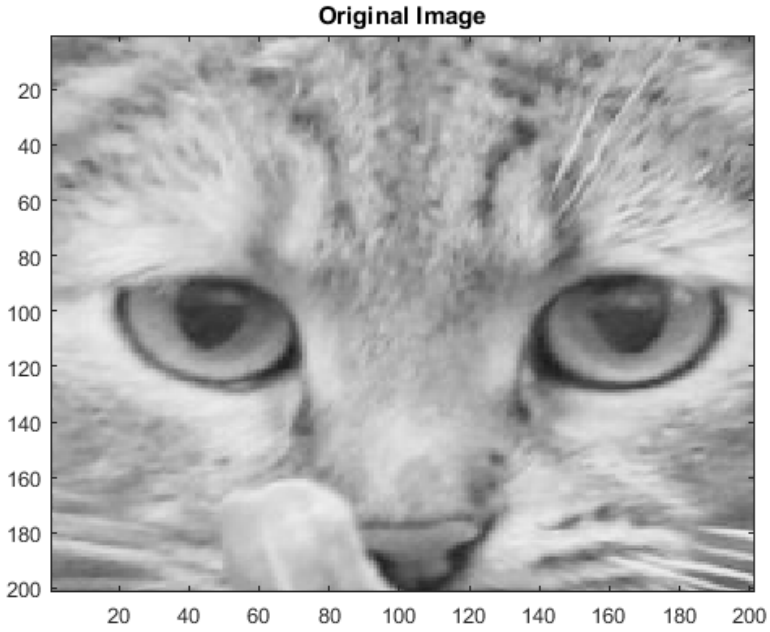

Figure 1: Obrazek oryginalny

1.2 Globalna

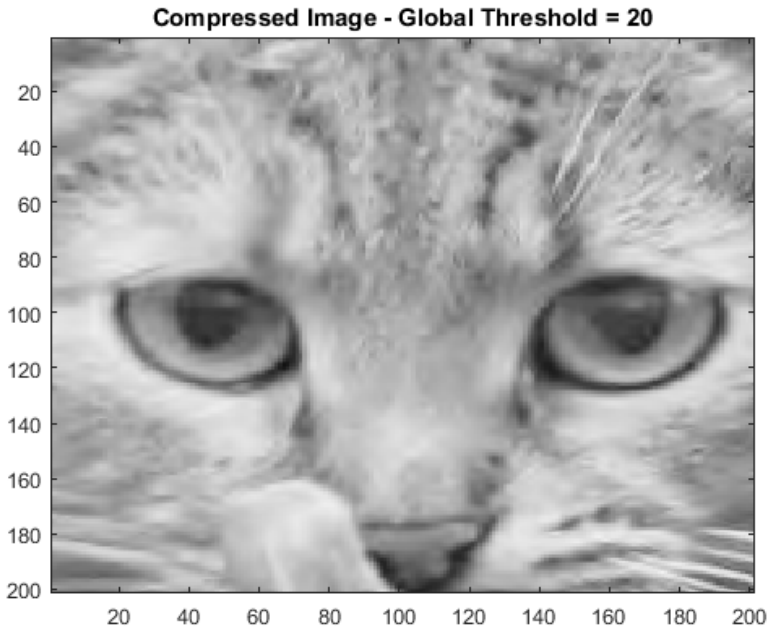

Figure 2: Obrazek po rekonstrukcji (globalny współczynnik)

Otrzymaliśmy następujące wartości miar:

perf0	85.2906
perfl2	99.9762

Jak widać jakość kompresji jest bardzo duża, przy czym ilość danych została zredukowana o prawie rząd wielkości.

1.3 Podzielona

Otrzymaliśmy następujące wartości miar:

perf0	86.5506
perfl2	99.9035

W tym przypadku otrzymaliśmy niewielką poprawę współczynnika kompresji, przy niewielkim pogordzeniu się jakości - na obrazku jest on praktycznie niedostrzegalny.

1.4 Użycie innych falek

Oryginalnie w przykładzie była użyta falka Symlet 8. Eksperymenty z innymi falkami przeprowadzone zostały na oddzielnych współczynnikach wycinania. Otrzymane zostały następujące wyniki:

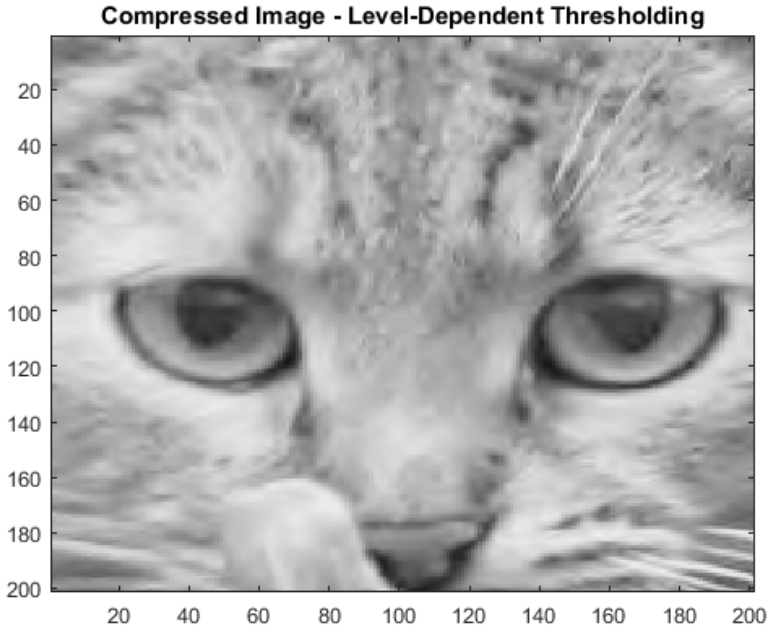

Figure 3: Obrazek po rekonstrukcji (oddzielne współczynniki)

Falka	perf0	perfl2
Symlet8	86.5506	99.9035
Haar	83.6760	99.8284
Daubechies4	87.6268	99.8906
Coiflets4	85.4645	99.9073
Symlet 4	87.4472	99.8926

Poprawę współczynnika kompresji uzyskano w przypadku Daubechies4 i Symlet4 aczkolwiek, kosztem nieznacznego pogorszenia jakości.

```
RGB = imread('../mietus.png');
X = double(rgb2gray(RGB));
% load woman;

x=X(100:300,400:600);
 image(x)
title('Original Image')
colormap(map)

n = 5;                    % Decomposition Level
w = 'sym4';
[c l] = wavedec2(x,n,w); % Multilevel 2-D wavelet decomposition.

% opt = 'gbl'; % Global threshold
% thr = 20;    % Threshold
% sorh = 'h';  % Hard thresholding
% keepapp = 1; % Approximation coefficients cannot be thresholded
% [xd,cxd,lxd,perf0,perfl2] = wdencmp(opt,c,l,w,n,thr,sorh,keepapp)
    ;
% image(x)
% title('Original Image')
% colormap(map)
% figure('Color','white'),image(xd)
% title('Compressed Image - Global Threshold = 20')
% colormap(map)

opt = 'lvd';          % Level dependent thresholds
thr_h = [17  18];     % Horizontal thresholds.
thr_d = [19  20];     % Diagonal thresholds.
thr_v = [21  22];     % Vertical thresholds.
thr = [thr_h ; thr_d ; thr_v];

[xd,cxd,lxd,perf0,perfl2] = wdencmp(opt,x,w,2,thr,sorh);
image(x)
title('Original Image')
colormap(map)
figure('Color','white'),image(xd)
title('Compressed Image - Level-Dependent Thresholding')
colormap(map)


perf0
perfl2
```

lab3.m

Figure 4: Listing kodu