Synteza mowy projekt - MFCC - wstępna klasyfikacja segmentów

Autorzy : Filip Bachura, Daniel Kaczmarek

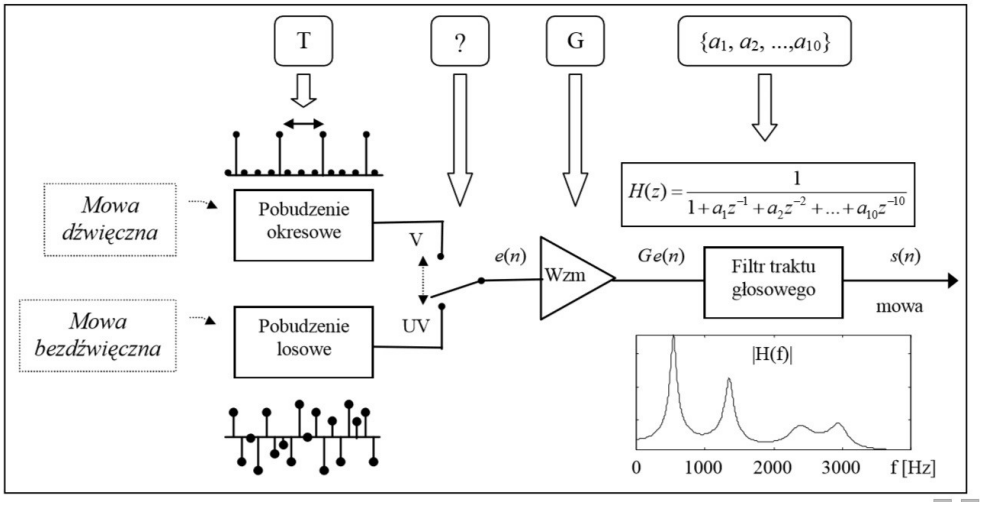
Spis treści:

1.Cel projektu :

Celem projektu było wykonanie programu, służącego do rozpoznawania izolowanych słów bez zastosowania metod statystycznych. Zadanie zostało wykonane korzystając ze środowiska programistycznego MATLAB, w którym przeprowadzono próby rozpoznania kilku słów, porównując je ze wzorami umieszczonymi w bazie składającej się z sześciu różniących się fonetycznie jak i znaczeniowo wyrazów (pkt 6. Wydajność) .

2.Model traktu krtaniowego :

Generację ludzkiego głosu można przedstawić jako model odpowiedniego pobudzenia, reprezentującego odpowiednio dla głosek bezdźwięcznych szum, bądź dla dźwięcznych okresowe pobudzenie, które w naszym przypadku możemy uprościć jako deltę Kroneckera. Pierwszym punktem modelu powinien być układ decyzyjny, określający jakie pobudzenie należy zastosować w danej chwili . Następnie, następuje wzmocnienie sygnału oraz poprowadzenie go przez filtr modelujący odpowiednie ułożenie języka, budowę przegrody nosowej oraz innych anatomicznych aspektów danego osobnika występujących w danej chwili czasu.



Rys. 1 Fotografia zaczerpnięta z książki T. Zielińskiego w celu zobrazowania modelu.

3. Wstępna analiza kodu :

```
 clear all


%Const variables
SAMPLERATE = 16000;
NSEC = 1.5;
N = 12;
Nfilter = 26;

%Record and play voice sample
%%SIGNAL = wavrecord(NSEC*SAMPLERATE,SAMPLERATE,'double');
%wavplay(SIGNAL,SAMPLERATE);
[SIGNAL,SAMPLERATE]=wavread('jeden.wav');

SIGNAL = silence( SIGNAL );

%Signal divided into 25milisecond frames
[ FramesMatrix ] = Framing( SIGNAL, SAMPLERATE );
```

Do rozpoznawania mowy nagrywano fragmenty mowy o długości 1.5 sekundy z częstotliwością próbkowania 16 kHz.

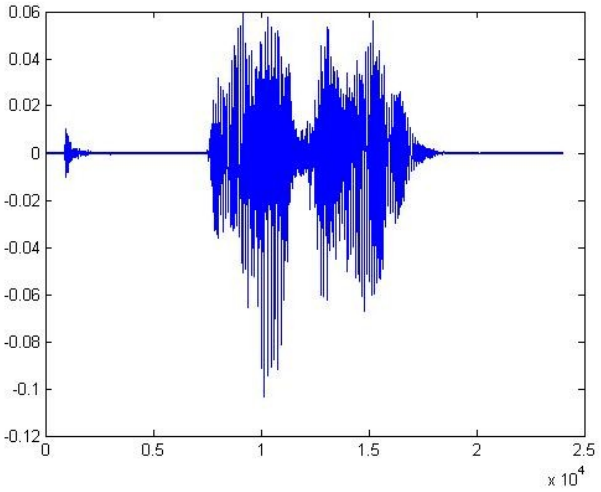
W celu analizy, zamiennie stosowano fragmenty mowy w formacie *.wav, identyczne z umieszczonymi w bazie.

W kolejnym kroku następowało wycięcie ciszy z sygnału w celu polepszenia wydajności algorytmu.

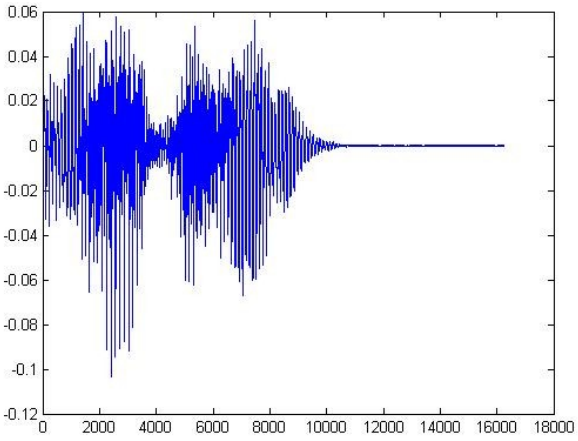
```
function [ nosilence ] = silence( SIGNAL )
%Silence function cuts silence from wav sample
%   In signal with silence
%   Out signal without silence

Th = 0.25;

tmpsig = SIGNAL/max(SIGNAL);
sth=(abs(tmpsig)>Th);
N = length(tmpsig);
sigstart = 0;


for i = 1:N
    if(sth(i) == 1)
        sigstart = i;
        break
    end
end
SIGNAL = SIGNAL(sigstart:end);
N = N - sigstart;
sigend = N;
figure
```

```
plot(sth);
for i = N:-1:1
    if(sth(i) == 1)
        sigend = i
        break
    end
end

SIGNAL = SIGNAL(1:sigend);
nosilence = SIGNAL;

end
```

Po tym zabiegu sprawność algorytmu wydajnie wzrosła, przy zastosowaniu bardziej zaawansowanych technik wycinania ciszy można znacznie polepszyć wydajność opisywanego algorytmu. W projekcie zastosowano najprostszy sposób pozbywania się ciszy z sygnału, czyli ustalenie stałego progu, powyżej którego fragment można jeszcze uznać jako sygnał, a nie jako ciszę; problemem okazało się jednak określenie stałego progu dla głosek bezdźwięcznych i dla ciszy, jednak przy uwzględnieniu tego ,że algorytm rozpoznaje wyłącznie izolowane słowa, ciszę wycinamy tylko na początku i na końcu sygnału, po wykryciu fragmentu nie będącego ciszą przerywamy wykonywanie funkcji.



Wykres 1 fragment sygnału przed wycięciem ciszy.



Wykres 2 fragment sygnału po wycięciu ciszy.

4. Stacjonarność analizowanego sygnału:

Sygnał mowy stale zmienia swoje widmo. W celu zapewnienia stacjonarności sygnału został on podzielony na ramki 25 ms. Zwykle budując algorytmy rozpoznawania mowy używa się ramek o szerokości od 20 ms do 40 ms, jest to spowodowane tym, że przy mniejszej ilości próbek nie otrzymujemy wiarygodnej estymaty widma, a dla większej ilości próbek sygnał zmienia się zbyt bardzo by zostać poddany analizie.

```
function [ FramesMatrix ] = Framing( SIGNAL, SampleRate )
%Framing function divide discrete function into
%many frames 25ms each and 10ms step
%   Input parameter is SIGNAL variable it contain recorded signal and
%   Sample Rate
%   Output parameter is FramesMatrix it is two dimensional matrix whitch
%   contains set of frames.

%Divide into 25ms frames
FrameSamples = round(0.025*SampleRate);
STEP = round(0.010*SampleRate);
N = length(SIGNAL);
NSTEPS = floor(N/STEP);
SIGNAL = [SIGNAL' zeros(1,160)];
for counter = 1:(NSTEPS -1)
    StartFrame = ((counter - 1)*STEP + 1);
    EndFrame = StartFrame + FrameSamples - 1 ;
    FramesMatrix(counter,:) = SIGNAL( StartFrame: EndFrame);
end

end
```

Funkcja zwraca dwuwymiarową macierz będącą zbiorem wszystkich ramek, na które został podzielony sygnał (z nakładaniem 10ms). Od tej funkcji wszystkie operacje nie odbywają się na całym sygnale lecz na pojedynczej ramce.

4. Ekstrakcja cech:

Przy tworzeniu algorytmu, którego celem jest rozpoznawanie mowy najważniejszą czynnością jest obranie odpowiedniej metody służącej do określenia cech konkretnego modelu. W opisywanym algorytmie, do wyekstrahowania wektora cech została użyta metoda Mel-Cepstralna. Jest ona rozszerzeniem metody cepstralnej, w którym uwzględniamy skalę melową, za pomocą której określamy sposób percepcji dźwięku przez człowieka, który lepiej rozróżnia wyższe niż niższe tony.

- Metoda Cepstralna – jest metodą wzorującą się na rozpatrywaniu filtru traktu głosowego ,jako filtra liniowego, różnej reprezentacji operacji mnożenia w dziedzinie czasu jak i częstotliwości oraz podstawowych własności logarytmu. Za pomocą tej metody możemy przedstawić sygnał jako sumę pobudzenia oraz odpowiedzi impulsowe filtru traktu głosowego .

$$c(k)=F^{-1}(\ln|H(e^{j\Omega})P(e^{j\Omega})|)=F^{-1}(\ln|H(e^{j\Omega})|)+F^{-1}(\ln|P(e^{j\Omega})|)$$

Gdzie P określa pobudzenie a H odpowiedz impulsową filtru.

- Metoda Mel-Cepstralna

a) Skala melowa – Logarytmiczna skala uwzględniająca percepcję dźwięku przez człowieka.

$$M(f) = 1125\ln(1 + f/700) \qquad M^{-1}(m) = 700(\exp(m/1125) - 1)$$

Implementacja funkcji przetwarzającej liniową skale częstotliwości na skale melowa.

```
function [ Mel ] = HertzToMel( Hertz )
%Function return value in Mel scale
%   Output Mel Scale Vector
%   Input Hertz Vector

Mel = 1125*log(1+Hertz/700);

end
```

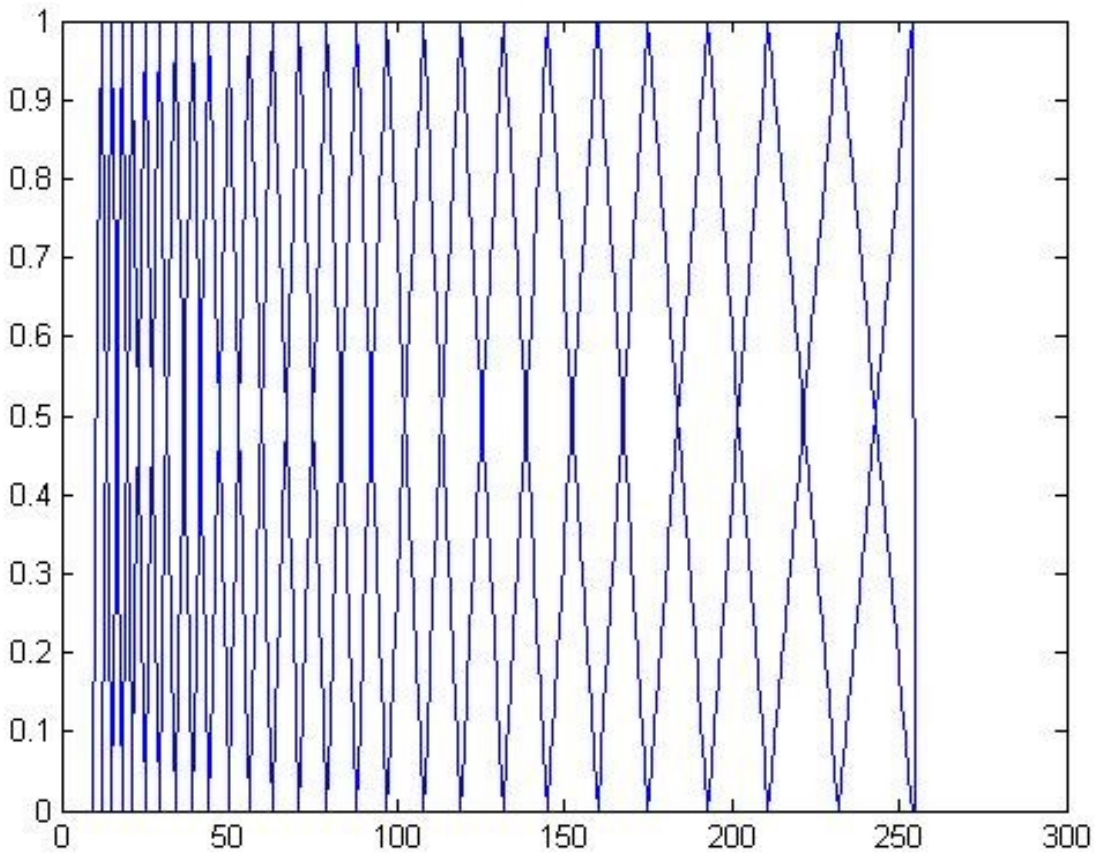
Implementacja funkcji przetwarzającej skale melową na skale częstotliwości.

```
function [ Hertz ] = MelToHertz( Mel )
%Function return value in Hertz scale
%   Output Hertz Scale Vector
%   Input Mel Scale Vector

Hertz = 700*(exp(Mel/1125)-1);

end
```

b) Algorytm MFCC

```
%For each Frame
for k = 1:SHor
    MPSD = [];
    %Multiply Frame with Hamming window
    frame = FramesMatrix(k,:).*window';
    %Preemfaza
    frame = filter([1 -0.9735], 1, frame);
    %Calculating Power Spectrum Density
    PSD = abs(fft(frame,512)).^2;
    PSD = PSD(256:end);
    %Calculate Mel filterbank
    [ MelMatrix ] = MelFilterBanks( 300, 8000, SAMPLERATE, Nfilter );

    %MelEnergies vector initialization
    MelEnergies = zeros(26,257);
    %Init vector for scalar calculation
    MelScalarEnergies = zeros(26);

    %Multiple each filter with PSD
    for i = 1:26
        MelEnergies(i,:) = MelMatrix(i,:).*PSD;
        %Energy inside filter
        MelScalarEnergies(i) = sum(MelEnergies(i,:));
    end

    %Energies Logaritm
    MelLogEnergies = log(MelScalarEnergies);
    for j = 1:N
        CepstralSum = 0;
        for n = 1:Nfilter
            CepstralSum = CepstralSum +
MelLogEnergies(n).*cos((pi*j/Nfilter)*(n-0.5));
        end
        MelCoefs(k, j) = sqrt(2/Nfilter)*CepstralSum;
    end
end
```

Każda ramka sygnału jest poddawana preemfazie zwiększając składowe wysokich częstotliwości, następnie wyliczana jest 512 punktowa estymata widma (PSD) która w kolejnych krokach będzie filtrowana przez wyliczony zestaw 26 filtrów Melowych.



Wykres 3. Zestaw 26 filtrów melowych zamieszczonych w dwuwymiarowej macierzy MelMatrix

```
function [ Mel ] = MelFilterBanks( Low, High, SampleRate, Nfilter )
%This function compute mel triangular filter bank used to generate Mel
%Cepstral Coefficients
%   Input : Sample Rate, Low frequency, High frequency of mel filter bank
%   and number of filters
%   Output: Mel FIlter Bank

Nfilters = Nfilter + 1;
FilterPoints = 257;
HighFreq = High;
LowFreq = Low;
LowMel = HertzToMel(LowFreq);
HighMel = HertzToMel(HighFreq);
STEP = (HighMel - LowMel)/26;
Melfreqs(1) = LowMel;
for i= 1:(Nfilters + 1)
   Melfreqs(i+1) = LowFreq + i*STEP ;
end
Melfreqs(end) = HighMel;
HzFreqs = MelToHertz(Melfreqs);
SampleFreqs = floor(512*HzFreqs/SampleRate) ;
for i = 2:(Nfilters +1)
    for j = 1:FilterPoints
        if(j < SampleFreqs(i-1))
            MelMatrix(i, j) = 0 ;
```

```
        else
            if(SampleFreqs(i-1) <= j && j<= SampleFreqs(i))
                MelMatrix(i, j) = (j - SampleFreqs(i - 1))/(SampleFreqs(i) -
SampleFreqs(i - 1));
            else
                if(SampleFreqs(i) <= j && j <= SampleFreqs(i + 1))
                    MelMatrix(i, j) = (-j + SampleFreqs(i+1))/(SampleFreqs(i+1)
- SampleFreqs(i));
                else
                    MelMatrix(i, j) = 0;
                end
            end
        end
    end
end
 Mel = MelMatrix(3:end,:);
end
```

Zbiór filtrów został obliczony na podstawie wzorów ze strony „http://practicalcryptography.com/” , zakres częstotliwości został obrany z zaleceniami ze strony oraz innych źródeł od 300 do 8000 Hz. Górna częstotliwość jest określona z częstotliwością próbkowania a dolna z częstotliwościami głosu człowieka.

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

Po wyliczeniu banku filtrów, widmo ramki sygnału jest przemnażane przez każdy z filtrów tworząc dwuwymiarową macierz `MelEnergies` zawierającą przefiltrowane sygnały oraz jednowymiarową macierz `MelScalarEnergies` zawierającą ilość zsumowanej energii „zawartej” w każdym z filtrów.

Wektor ten następnie logarytmujemy oraz realizujemy sumę po wszystkich elementach wektora, zgodnie ze wzorem na transformatę DCT II .

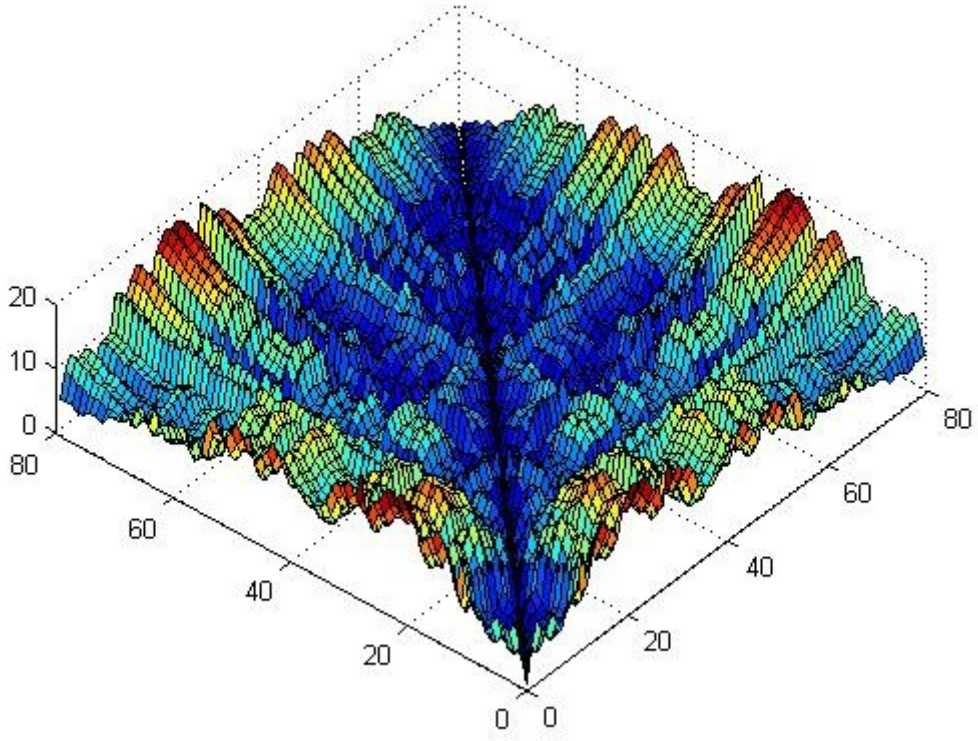
$$c_k = \sqrt{\frac{2}{L}} \cdot \sum_{l=0}^{L-1} \ln\left(\tilde{S}(l)\right) \cos\left(\frac{\pi k}{L}(l+1/2)\right), \quad k = 0, 1, 2, \ldots, q-1$$

6. Porównywanie wzorców – DTW :

Do porównywania sygnałów z bazy z nagraną próbką została użyta nieliniowa transformacja czasowa DTW.
Na początku tworzona jest macierz Ns × Nw , której każdy element ma wartość odległości euklidesowej między cepstrum elementu z bazy a nagranym sygnałem.

$$d(n_s, n_w) = \sqrt{\sum (c_p(n_s, i) - c_w(n_w, i))^2}$$

Liczony jest koszt przejścia od lewego dolnego rogu do prawego górnego rogu, dla wyrazów identycznych droga przejścia jest równa zero, co widać na wykresach dla analizy identycznych wyrazów (nagranie jeden.wav).



Wykres 4. ilustracja drogi przejścia dla nagranie jeden.wav

```
function [ sumdist ] = DTW( MelCoefs, MelCoefsx )
%DTW function calculating distance between sample model
%and recorded model
%   Input Recorded Model, Sample Model
%   Output Calculated distance

COEFSNUMBER = 12;

[samph, ~] = size(MelCoefsx);
[sizh, ~] = size(MelCoefs);


for ns = 1:sizh
    for nw = 1:samph
        sum = 0;
        for k = 1:COEFSNUMBER

           sum = sum + (MelCoefs(ns, k) - MelCoefsx(nw, k)).^2;
        end
        d(ns, nw)  = sqrt(sum);
    end
end

surf(d);
g = d;

for ns = 2: sizh, g(ns, 1) = g(ns-1, 1) + d(ns, 1); end
for nw = 2: samph, g(1, nw) = g(1, nw-1) + d(1, nw); end

for ns = 2:sizh
    for nw = 2:samph
        dd = d(ns,nw);
        tmp(1) = g(ns-1, nw) + dd;
        tmp(2) = g(ns-1, nw-1) + 2*dd;
        tmp(3) = g(ns, nw-1) + dd;
        g(ns, nw) = min(tmp);

    end
end

sumdist = g(sizh, samph)/sqrt(sizh^2 + samph^2);
```

Pierwsza część kodu określa wyliczenie macierzy „odległości”, a druga część kodu wyliczenie kosztu przejścia poruszając się w określonych kierunkach , prawo skos góra .

```
tmp(1) = g(ns-1, nw) + dd;
tmp(2) = g(ns-1, nw-1) + 2*dd;
tmp(3) = g(ns, nw-1) + dd;
```

7.Wydajność :

Przy braku zastosowania metod statystycznych algorytm nie jest bardzo wydajny. Efektywność rozpoznawania waha się w pobliżu 50 % - 60 % . Próbki były nagrywane przez dwóch mówców zależnie od wyboru mówcy osiągnięto inną wydajność.

Osoba	Liczba wzorów nagranych w bazie	Próba 1	Próba 2	Wydajność
Autor 1	4	6/6 wykryto ostatnią po dwóch próbach	5/6 wykryto	80 % - 90 %
Autor 2	2	5/6 wykryto	4/6 wykryto	75%
Osobnik 1	0	4/6 wykryto jedną po dwóch próbach	3/6 wykryto	58%

Do wykonania projektu wykorzystano :

Artykuł ze strony internetowej „Mel Frequency Cepstral Coefficient (MFCC)”
http://practicalcryptography.com

Tadeusz Zieliński „Cyfrowe przetwarzanie sygnałów od teorii do zastosowań” s. 568 - 593