L1. Procesarea semnalelor audio folosind functiile Matlab

1. Semnale audio

Semnalele audio se refera, în general, la semnale care pot fi percepute de om. Semnalele audio, de obicei, provin dintr-o sursă de sunet care vibrează în gama de frecvențe audio (20Hz-20kHz). Vibrațiile pun in miscare aerul pentru a forma „valuri de presiune" care se propaga cu aproximativ 340 m/s. Urechile pot primi aceste semnale de presiune și le trimit creierului pentru recunoașterea acestora.

Există numeroase moduri de a clasifica semnalele audio. Dacă luăm în considerare **sursa de semnale audio**, le putem clasifica în două categorii:

• **Sunete produse de vietuitoare**: voci umane, latrat de caine, mieunat de pisica, orăcăitul broastelor etc. În particular, bioacustica este o știință inter-disciplinara, care investigheaza producerea sunetelor de catre vietuitoare și recepția lor de catre acestea.

• **Sunete de la non-vietuitoare**: Sunete de la motoarele auto, tunete, trîntitul uși, instrumente muzicale etc.

Dacă luăm în considerare modul de repetare a semnalelor audio, acestea se pot clasifica în două categorii:

• **Sunete cvasi-periodice**: formele de undă sunt aproape periodice, astfel încât putem detecta perioada de repetitie (pitch). Exemple de astfel de sunete includ redarea monofonica a majoritatii instrumentelor muzicale (cum ar fi pianul, vioara, chitara etc) și vorbirea in anumite zone sau cantatul uman.

• **Sunete aperiodice**: formele de undă nu sunt formate din tipare evidente repetate (forme periodice), astfel că nu putem percepe o frecventa de repetitie stabila. Exemple de astfel de sunete includ tunete, batut din palme, parti nesonore in rostirea umana șamd.

În principiu, putem clasifica fiecare segment scurt de vorbire (cunoscut si sub numele de cadru, cu o lungime de aproximativ 20 ms) în două tipuri:

• **Segment sonor:** Acestea sunt produse de vibrația periodica a corzilor vocale, deci pot fi observate perioadele fundamentale într-un cadru. Mai mult decât atât, ca urmare a existenței perioadei fundamentale, poate fi estimata valoarea sa.

• **Segment nesonor:** Acestea nu sunt produse de vibrația corzilor vocale ci de fluxul rapid de aer expulzat prin intermediul tractului vocal. Deoarece aceste sunete sunt produse de un zgomot, cum ar fi fluxul de aer rapid, perioada fundamentala nu poate fi observata si nici o frecventa stabila nu poate fi detectata.

Semnalele audio reprezintă, de fapt, variatia presiunii aerului ca o funcție de timp, care este continua atât în timp cat și in amplitudine. Când se doreste achizitia semnalelor pentru stocarea într-un calculator, există mai multi parametri care trebuie luati in considerare:

1. **Rata de eșantionare:** aceasta reprezinta numărul de puncte de eșantionare pe secundă, în unitatea de Hertz (Hz). O rată de esantionare mai mare indica o calitate mai bună a sunetului, dar spațiul de stocare necesar este de asemenea mai mare. Ratele utilizate în mod obișnuit in eșantionare sunt prezentate în continuare:
- 8 kHz: calitatea vocii pentru telefoane și jucării;

- 16 KHz: Frecvent utilizate pentru recunoașterea vorbirii;
- 44,1 KHz: calitate de CD;

2. **Rezoluția**: Numărul de biți folosiți pentru a reprezenta fiecare eșantion din semnalul audio. Rezoluțiile utilizate în mod obișnuit sunt :

- 8-bit: Gama corespunzătoare este 0 - 255 sau + 127... -128 sau
- 16-bit: Gama corespunzător este -32768 - + 32767.

Cu alte cuvinte, fiecare punct esantionat este reprezentat de un număr întreg de 8 sau 16 biți. Cu toate acestea, în MATLAB, toate semnalele audio sunt transformate in virgulă mobilă în intervalul [-1, 1], pentru o manipulare mai eficienta. Dacă se doreste revenirea valorile inițiale întregi, este nevoie ca valorile in virgula mobila sa fie inmultite cu $2^{nbits-1}$, unde nbits este rezoluția in biti.

3. **Nr. de Canale**: Mono pentru un singur canal și stereo pentru 2 canalele stereo.

2. Caracteristici acustice de baza ale semnalului vocal

Când sunt analizate semnalele audio, este adoptata, de obicei, metoda de analiză pe termen scurt, deoarece semnale audio sunt mai mult sau mai puțin stabile într-o perioadă scurtă de timp, de aproximativ 10-30 ms. Când se face analiza cadru cu cadru, pot exista suprapuneri între cadrele vecine pentru a surprinde schimbarea fină în semnale audio. De retinut este că fiecare cadru este unitatea de bază pentru analiza facuta. În fiecare cadru, se pot observa/defini urmatoarele caracteristici acustice.

• **Volumul:** Această caracteristică reprezintă intensitatea semnalului audio, care este corelat cu amplitudinea semnalelor. Uneori se refera, de asemenea, ca fiind energia sau intensitatea semnalelor audio.

• **Pitch:** Această caracteristică (inaltimea) reprezintă frecventa de repetitie a semnalelor audio, care poate fi reprezentata de frecvența fundamentală (FF sau F0), corespunzator perioadei fundamentale a semnalelor audio sonore.

• **Timbrul:** este un atribut multidimensional care depinde de mai multe variabile fizice. De multe ori timbrul este definit într-o manieră pur negativă ca "tot ceea ce nu este intensitate, pitch sau percepție spațială". Timbrul este definit ca "acel atribut de senzație auditivă, în care un ascultător poate judeca că două sunete prezentate în mod similar și având aceiasi intensitate și același pitch sunt diferite". Există, în primul rând, conținutul in frecvență și profilul spectral al sunetului. Deoarece urechea umană are putere de rezoluție limitată in frecvență, vectorul compoziției spectrale poate fi adesea redus la un vector care reprezintă cantitatea de energie acustică instantanee în fiecare bandă critică (Plomp 1970), fără prea mari pierderi de informații perceptuale. În cele din urmă, anvelopa temporală a unui sunet instrumental, inclusiv atacul, caderea și modularea porțiunii stabile, influențează timbrul perceput într-o asemenea măsura care poate face sunetul unui instrument de nerecunoscut (Berger 1964).

Procedura de bază la extragerea de caracteristici acustice poate fi urmatoarea:

1. Se face impartirea pe cadre, astfel încât o secventa a semnalului audio este impartita intr-un set de cadre. Durata de timp pentru fiecare cadru este de aproximativ 10-30 ms. Dacă durata cadrului este prea mare, nu putem surprinde caracteristicile rapid variabile in timp ale semnalului audio. Pe de altă parte, dacă durata cadrului este prea mica, atunci nu putem extrage caracteristici acustice valide. În general, un cadru ar trebui să conțina mai multe perioade fundamentale ale semnalului audio. De obicei, dimensiunea cadrului (în numar de eșantionate) este egala cu puterile lui 2 (cum ar fi 256, 512, 1024 etc), astfel că este potrivita pentru transformata Fourier rapidă.

2. Dacă se doreste reducerea diferențelor dintre cadre vecine, se permite suprapunerea acestora. De obicei, suprapunerea este de obicei intre 1/2 si 2/3 din cadru original. Cu cat suprapunerea este mai mare, cu atat numarul calculelor creste.
3. Presupunând că semnalele audio într-un cadru sunt cvasistaționare, putem extrage diferiti parametri sau caracteristici cum ar fi numarul de treceri prin zero, volum, pitch, MFCC, LPC etc.

Lucrarea de fata isi propune o prezentare succinta funcțiilor din MATLAB care pot fi folosite pentru preluarea, stocarea sau redarea semnalului audio, cat si a metodelor de baza pentru procesarea semnalului audio. Functiile Matlab descrise vor fi insotite de exemple.

3. Procesarea semnalului audio folosind functiile Matlab

Acest capitol introduce cateva dintre cele mai importante functii din cadrul mediului Matlab, functii ce sunt utilizate pentru procesarea semnalului audio. Astfel, vor fi prezentate urmatoarele functii pentru:

1. Citirea fisierelor .wav;
2. Redarea semnalelor audio;
3. Inregistrarea de la microfon;
4. Salvarea fisierelor .wav.

3.1 Citirea fisierelor .wav

In MATLAB se pot citi fișiere de tip .wav prin intermediul comenzii "wavread". Următorul exemplu citește fisierul "boy.wav" și afiseaza forma de undă a acestuia.

```
[y, fs]=wavread('boy.wav');
sound(y, fs);            % Reda semnalul audio
time=(1:length(y))/fs;   % Vectorul timp pe axa X
plot(time, y);           % Afiseaza forma de unda in timp
```

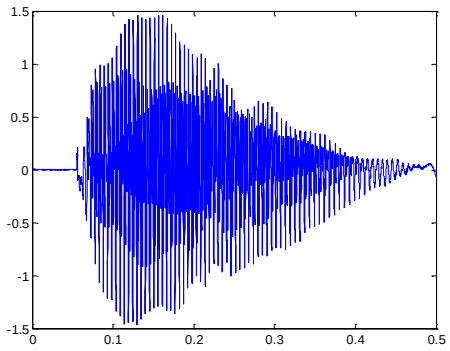

Figure 1 - Forma de unda a rostirii "boy"

În exemplul de mai sus, "fs" este frecventa de eșantionare, care este 16000, în acest caz. Acest lucru indică faptul că există 16000 de eșantioane pe secundă atunci când sunetul a fost înregistrat. Vectorul "y" este un vector coloană care conține eșantioanele semnalului vocal. Putem folosi comanda "sound(y, fs)" pentru a reda semnalul audio citit din fișier. "time" este un vector de timp, în care fiecare element corespunde la momentul fiecărui eșantion. Prin urmare, putem afisa "y" vs. "t" adica y(t), pentru a vizualiza forma de undă directa.

Cele mai multe semnale audio sunt digitizate cu o rezoluție bit de 8 sau 16 biți. Dacă vrem să știm rezoluția de biți a fișierului de intrare .wav, putem folosi parametri suplimentari de ieșire la functia "wavread" pentru a obține informații, cum ar fi:

```
[y, fs, nbits]=wavread('girl.wav');
```

Mai mult decât atât, dacă vrem să știm durata de timp a unui fisier audio, putem folosi direct functia "length(y)/fs". Următorul exemplu poate obține cele mai importante informații dintr-un fișier .wav.

```
close all;
clear all;
[filename1, pathname1]=uigetfile('*.wav')
[y, fs, nbits]=wavread(filename1);
fprintf('Informatiile fisierului "%s":\n', numeFisier);
fprintf('Durata = %g secunde\n', length(y)/fs);
fprintf('Frecventa de esantionare = %g esantioane/secunda\n', fs);
fprintf('Rezolutie bit = %g biti/esantion\n', nbits);
```

```
Informatiile fisierului "girl.wav":
Durata = 0.889188 secunde
Frecventa de esantionare = 16000 esantioane/secunda
Rezolutie bit = 16 biti/esantion
```

Din forma de unda de mai sus data ca exemplu, se poate observa că semnalul audio este normalizat, amplitudinea avand valori între -1 și 1. Cu toate acestea, fiecare eșantion este reprezentat de un număr întreg pe 8/16 biți. Cum sunt corelate intre ele? Mai întâi de toate, trebuie să știm următoarea convenție:

a) Dacă un fișier .wav are o rezoluție de 8 biți, atunci fiecare esantion este stocat ca un întreg fara semn între 0 și 255 (2^8-1).
b) Dacă un fișier .wav are o rezoluție de 16 biți, atunci fiecare punct esantionat este stocat ca un întreg cu semn între -32768 (2^{15}) și 32767 (2^{15}-1).

Din moment ce aproape toate variabilele din MATLAB au tipul de date de "double", prin urmare, toate eșantioanele sunt convertite într-un număr în virgulă mobilă între [-1 și 1), pentru manipulare/procesare usoara. Prin urmare, pentru a prelua valorile originale întregi ale semnalelor audio, se poate proceda după cum urmează.

a) Pentru rezolutie de 8-biti, se multiplica "y" (valoarea obținută prin wavread) cu 128 și apoi, se aduna 128.
b) Pentru 16-bit rezolutie, se multiplica "y" (valoarea obținută prin wavread) cu 32768.

Se poate folosi, de asemenea, comanda "wavread" pentru a citi un fișier .wav stereo. Variabila returnata va fi o matrice de 2 coloane, fiecare conținând semnalele audio de la un singur canal.

3.2 Redarea semnalului audio

Odată ce putem citi fișierul vawe în MATLAB, putem începe procesarea semnalelor audio prin modificarea intensităților lor, sau modificarea ratelor de eșantionare a acestora, șamd. După ce semnalele audio sunt procesate, este nevoie de redarea lor pentru o inspecție audio. Comanda de baza pentru redarea semnalelor audio este "wavplay". Următorul exemplu poate încărca un flux de date audio din fisierul "handel.mat" și reda imediat semnalul audio incarcat.

```
load handel.mat      % Incarca semnale stocate in handel.mat
wavplay(y, Fs);      % Reda semnalul audio
```

Deoarece volumul de redare este determinat de amplitudinea semnalelor audio, putem schimba amplitudinea pentru a modifica volumul, după cum urmează.

```
[y, fs]=wavread('do.wav');
wavplay(1*y, fs, 'sync');    % Reda semnalul cu amplitudinea originala
wavplay(3*y, fs, 'sync');    % Reda semnalul amplificat de 3 ori
wavplay(5*y, fs, 'sync');    % Reda semnalul amplificat de 5 ori
```

În exemplul de mai sus, amplitudinea este crescuta treptat, astfel încât să putem percepe creșterea volumului în timpul redării. În particular, comanda "wavplay" presupune ca semnalele de intrare sunt între -1 și 1. Când semnalele de intrare sunt în afara acestui interval (prea mare sau prea mic), putem auzi "sunetul intrerupt". Se poate încerca acest lucru executand comanda "wavplay(100*y,fs)" pentru a auzi rezultatul. Mai mult decât atât, argumentul suplimentar de intrare "sync" în exemplul de mai sus face comanda "wavplay" sa redea semnalele sincron, adica redarea nu va începe până când redarea precedenta nu este terminata.

În exemplul de mai sus, deși avem creștere in amplitudine cu un factor de 5, intensitatea percepută de către urechea umana nu este de factorul 5. Acest lucru servește pentru a ilustra faptul că percepția volumului nu este proporțională liniar cu amplitudinea. De fapt, aceasta este proporțională cu logaritmul amplitudinii.

Dacă vom schimba frecventa de eșantionare în timpul redării, aceasta va afecta durata de timp, precum și frecventa pitch perceputa. În următorul exemplu, vom crește frecventa de eșantionare treptat, astfel încât se va auzi un sunet scurt, cu o frecventa pitch , similar cu sunetul personajului din desenele animate Disney Donald Duck.

```
[y, fs]=wavread('church.wav');
wavplay(y, 1.0*fs, 'sync'); % Reda semnalul la viteza normala
wavplay(y, 1.2*fs, 'sync'); % Reda semnalul la o viteza de 1.2 ori mai mare
wavplay(y, 1.5*fs, 'sync'); % Reda semnalul la o viteza de 1.5 ori mai mare
wavplay(y, 2.0*fs, 'sync'); % Reda semnalul la o viteza de 2 ori mai mare
```

Pe de altă parte, dacă am reduce rata de eșantionare treptat, vom avea sunete mai lungi și cu frecventa fundamentala mai scăzuta, iar în cele din urmă acesta va suna ca un muget de vaca.

```
[y, fs]=wavread('church.wav');

wavplay(y, 1.0*fs, 'sync'); % Reda semnalul la viteza normala
wavplay(y, 0.9*fs, 'sync'); % Reda semnalul la o viteza 0.9 din fs
wavplay(y, 0.8*fs, 'sync'); % Reda semnalul la o viteza 0.8 din fs
wavplay(y, 0.6*fs, 'sync'); % Reda semnalul la o viteza 0.6 din fs
```

Dacă am inversa semnalul audio prin înmulțirea sa cu -1, percepția va fi exact la fel ca originalul. Acest lucru, de asemenea, serveste pentru a demonstra că percepția umană a semnalului audio nu este afectată de faza. Cu toate acestea, în cazul inversarii semnalului audio pe axa timpului, atunci acesta va suna ca o limbă necunoscută ca in urmatorul exemplu.

```
[y, fs]=wavread('floare.wav');
wavplay(y, fs, 'sync');          % Reda semnalul original
wavplay(-y, fs, 'sync');         % Reda semnalul inversat pe axa Y (sus-jos)
wavplay(flipud(y), fs, 'sync'); % Reda semnalul oglindit dreapta-stanga
                                 %("reverse speech")
```

Alte comenzi similare cu “wavplay” pentru redarea semnalelor audio este “sound” si “play”, aceasta ultima comanda este folosita pentru obiecte de tip "apObj" (un obiect audio player creat de catre Matlab).

3.3 Inregistrarea sunetelor de la microfon

Se poate folosi comanda Matlab "wavrecord" pentru a inregistra semnalele audio direct de la microfon. Comanda este:

y = wavrecord (n, fs);

unde "n" este numărul de esantioane care urmează să fie înregistrate, iar "fs" este frecventa de eșantionare. In următorul exemplu se înregistreaza 5 secunde de la microfon.

```
fs=16000;                       % Frecventa de esantionare
duration=2;                     % Durata de inregistrare
fprintf('Apasa orice tasta pentru a incepe inregistrarea a %g secunde...',
duration); pause
fprintf('Se inregistreaza...');
y=wavrecord(duration*fs, fs);   % Durata*fs este numarul total de esantioane
fprintf('Inregistrare terminata.\n');
fprintf('Apasa orice tasta pentru a reda inregistrarea...'); pause;
fprintf('\n');
wavplay(y,fs);
```

În exemplul de mai sus, "durata*fs" este numărul de eșantioane care urmează să fie înregistrate. Eșantioanele înregistrate sunt stocate în variabila "y", care este un vector de dimensiune 32000x1. Tipul de date al lui "y" este dublu și spațiul de memorie ocupat de "y" este 256000 bytes.
În exemplul anterior, numărul de canale este 1 și tipul de date pentru punctele eșantionate este double. Dacă vrem să schimbăm aceste doua setări implicite, putem introduce parametrii suplimentari de intrare la comanda "wavrecord". O comanda detaliata "wavrecord" este:

```
y = wavrecord(n, fs, channel, dataType);
```

unde "channel” (de obicei 1 sau 2) este numărul de canale de înregistrare, și "dataType" (cum ar fi "double", "single", "int16", "uint8") este tipul de date a pentru eșantioanele înregistrate. Tipuri de date diferite necesita spațiu de memorie diferit pentru stocare. Exemplul următor:

```
fs=16000;       % Frecventa de esantionare
duration=2;     % Durata de inregistrare
channel=1;      % Canal mono
fprintf('Apasa orice tasta pentru a incepe inregistrarea a %g secunde...',
duration); pause
fprintf('Se inregistreaza...');
y=wavrecord(duration*fs, fs, channel, 'uint8');
                % Durata*fs este numarul total de esantioane
fprintf('Inregistrare terminata.\n');
fprintf('Apasa orice tasta pentru a reda inregistrarea...'); pause;
fprintf('\n');
wavplay(y,fs);
```

Acest exemplu este aproape același ca și cel precedent, cu excepția faptului că tipul de date este "uint8". Punctele de eșantionare sunt păstrate în variabila "y" cu aceasi dimensiune 32000x1. Dar elemente din "y" sunt numere întregi cuprinse între 0 și 255. Spațiul de memorie a lui "y" este acum doar 32000 bytes, care este numai 1/8 din spatiul ocupat în exemplul anterior.

Tabelul de mai jos prezintă tipurile de date suportate de comanda wavrecord.

Data types	Space requirement per sample	Range of the sample data
double	8 bytes/sample	Real number within [-1, 1]
single	4 bytes/sample	Real number within [-1, 1]
int16	2 bytes/sample	Integer within [-32768, 32767] or [-2^(nbits-1), 2^(nbits-1)-1]
uint8	1 byte/sample	Integer within [0, 255] or [0, 2^nbits-1]

Figure 2 - Tipuri de date suportate de comanda wavrecord

MATLAB mai ofera, de asemenea, o altă comandă, "audiorecorder", pentru a oferi un control de precizie asupra înregistrării.

```
fs=16000;               % Frecventa de esantionare
nbits=16;
nChannels=1;
duration=3;             % Durata de inregistrare
arObj=audiorecorder(fs, nbits, nChannels);
fprintf('Apasa orice tasta pentru a incepe inregistrarea a %g secunde...',
duration); pause
fprintf('Se inregistreaza...');
recordblocking(arObj, duration);
fprintf('Inregistrare terminata.\n');
fprintf('Apasa orice tasta pentru a reda inregistrarea...'); pause;
fprintf('\n');
play(arObj);
fprintf('Se afiseaza forma de unda a semnalului inregistrat\n');
y=getaudiodata(arObj);  % Ia datele audio esantionate
plot(y);                % Afiseaza forma de unda
```

Apasa orice tasta pentru a incepe inregistrarea a 3 secunde...Se inregistreaza...Inregistrare terminata.
Apasa orice tasta pentru a reda inregistrarea...
Se afiseaza forma de unda a semnalului inregistrat.

3.4 Salvarea/ scrierea fisierelor audio

Putem scrie fisiere audio ".wav" utilizând comanda MATLAB "wavwrite".

Comanda este:

wavwrite(y, fs, nbits, waveFile);

unde "y" este de data audio, "fs" este rata de eșantionare, "nbits" este rezoluția esantioanelor in biti, și "waveFile" este fișierul .wav unde va fi scris semnalul. Următorul exemplu scrie datele audio înregistrare într-un fișier "test.wav".

În acest exemplu, vom stoca datele audio cu tipul "uint8" in fișierul "test.wav". Vom invoca apoi aplicația corespunzătoare pentru redarea fișierului. Deoarece variabila "y" pentru comanda "wavwrite" ar trebui să fie de tipul double în intervalul [-1, 1], este nevoie de executarea unor conversii în cazul în care datele înregistrate sunt de alte tipuri de date, cum ar fi "single", "int16 ", sau" uint8 ". Aici este tabelul pentru conversie.

```
fs=11025;                % Frecventa de esantionare
duration=2;              % Durata de inregistrare
waveFile='bye.wav';      % Fisierul wav care va fi salvat
fprintf('Apasa orice tasta pentru a incepe inregistrarea a %g secunde...',
duration); pause;
fprintf('Se inregistreaza...');
y=wavrecord(duration*fs, fs);
fprintf('Inregistrare terminata.\n');
fprintf('Apasa orice tasta pentru a salva inregistrarea in %s...', waveFile);
pause;
fprintf('\n');
nbits=8;                 % Rezolutia bit
wavwrite(y, fs, nbits, waveFile);
fprintf('S-a terminat scrierea fisierului %s\n', waveFile);
fprintf('Apasa orice tasta pentru a reda semnalul scris %s...\n', waveFile);
dos(['start ', waveFile]);  % Porneste aplicatia pentru fisierul .wav
```

Data types of "y"	How to convert it to 'double' within [-1, 1]
double	No conversion needed
single	y = double(y);
int16	y = double(y)/32768;
uint8	y = (double(y)-128)/128;

Figure 3 - Conversia datelor in tipul double

MATLAB poate scrie, de asemenea, alte fișiere audio, cum ar fi ".au", care sunt fișierele audio utilizate în stațiile de lucru NeXT /SUN. Comanda corespunzător este "auwrite".

3.5. Controlul volumului in fisierele audio

Intensitatea sonoră a semnalelor audio este cea mai proeminenta caracteristica pentru percepția auditivă umana. În general, există mai multi termeni similari care sunt utilizati în mod obișnuit pentru a descrie tăria semnalelor audio: volum, intensitate sau energie. Din motive de coerență, aici vom folosi termenul de "volum" pentru a descrie intensitatea. Practic, volumul este o caracteristică acustica, care este corelata cu amplitudinile eșantioanelor dintr-un cadru. Pentru a defini volumul cantitativ, putem folosi două metode pentru a calcula volumul unui cadru dat:

1. Suma esantioanelor in valoare absoluta din fiecare cadru:

$$volume = \sum_{i=1}^{n} |s_i|$$

unde s(i) este al i-lea esantion într-un cadru, iar n este dimensiunea cadrului. Această metodă necesită doar operații întregi și este potrivita pentru sisteme low-end, cum ar fi microcontrolerele.

2. Sau cel dat de formula:

$$volume = 10 * log_{10} \sum_{i=1}^{n} s_i^2$$

Această metodă necesită calcule în virgulă mobilă, dar este (mai mult sau mai putin) liniar corelata cu percepția umana a intensitatii semnalelor audio. (http://www.phys.unsw.edu.au/~jw/dB.html)

Cateva dintre caracteristicile intensitatii sonore sunt prezentate in cele ce urmeaza:

• Pentru înregistrarea într-un loc liniștit folosind un microfon uni-directional, volumul sunetelor sonore este de obicei mai mare decât cel al sunetelor nesonore, iar volumul sunetelor nesonore este de obicei mai mare decât cea a zgomotului ambiental (cu toate acestea, acest lucru nu este aplicabil inregistrarii prin intermediul microfonului omni-directional);

• Volumul este foarte mult influentat de setarile microfonului, mai ales câștigul microfonului;

• Volumul este de obicei folosit pentru detectarea regiunilor de activitate a vocii (voice activity detection).

Înainte de a calcula volumul, se va efectua, de obicei, o eliminare a offsetului (prin scăderea simpla a mediei cadrului din fiecare esantion), pentru a elimina un potențial DC.
Pentru metoda 1, vom aplica, de obicei, scăderea medianei pentru eliminarea offsetului.
Pentru metoda 2, vom aplica, scăderea mediei pentru ajustarea zeroului.

Utilizarea celor doua metode pentru calculul volumului este ilustrat de exemplul urmator.

```
waveFile='church.wav';
frameSize=256; overlap=128;
[y, fs, nbits]=wavread(waveFile);
fprintf('Durata of %s is %g sec.\n', waveFile, length(y)/fs);
frameMat=buffer(y, frameSize, overlap);
frameNum=size(frameMat, 2);   % Calculeaza volumul cu metoda 1
volume1=zeros(frameNum, 1);
for i=1:frameNum
    frame=frameMat(:,i);
    frame=frame-median(frame); % zero-justified
    volume1(i)=sum(abs(frame)); % metoda 1
end

volume2=zeros(frameNum, 1);   % Calculeaza volumul cu metoda 2
for i=1:frameNum frame=frameMat(:,i);
    frame=frame-mean(frame);  % zero-justified
    volume2(i)=10*log10(sum(frame.^2)+realmin); % metoda 2
end
sampleTime=(1:length(y))/fs;
frameTime=((0:frameNum-1)*(frameSize-overlap)+0.5*frameSize)/fs;
subplot(3,1,1);
plot(sampleTime, y);
ylabel(waveFile);
subplot(3,1,2);
plot(frameTime, volume1, '.-');
ylabel('Volum (Abs. sum)');
subplot(3,1,3); plot(frameTime, volume2, '.-');
ylabel('Volum (Decibels)'); xlabel('Timp (sec)');
```

Cele două metode de calcul a intensitatii sunt doar o aproximare a percepției umane. Cu toate acestea, intensitatea sunetului se bazează pe percepția umana și ar putea exista diferențe semnificative între "intensitatea calculata" și "intensitatea perceputa". De fapt, intensitatea perceputa este foarte mult afectată de frecventa, precum și de timbrul semnalelor audio.

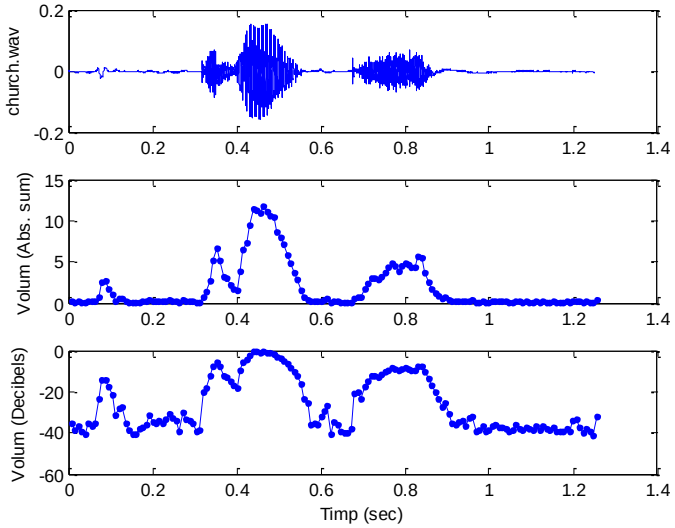

Figura 4 – Calculul volumului prin metodele 1 si 2 („church")

4. Mersul lucrarii.

Pe baza celor studiate anterior in cap.1-3 rezolvati problemele si cerintele urmatoare si realizati raportul cu rezultate, capturi, cod etc.

4.1. Obțineți informații dintr-un fișier audio mono. Scrieti un script MATLAB care poate citi fisierul "church.wav" și va afișa următoarele informații în acest script:
Numărul de puncte de eșantionare.
Rata esantionare
Rezoluție Biti
Numărul de canale
Durata de timp a înregistrării (în secunde)

4.2. Inregistrare Wave. Scrieți un script MATLAB pentru a înregistra 10 de secunde de vorbire, cum ar fi "Numele meu este X....Y.... și sunt un student la master in anul I TM la departamentul de Comunicatii al Universitatii Tehnice din Cluj-Napoca". Salvați înregistrarea ca myVoice.wav. Alți parametri de înregistrare sunt: Frecventa de eșantionare = 16 kHz, rezoluție biți = 16 biți. Script-ul va permite indicarea răspunsurilor la următoarele întrebări, în cadrul ferestrei MATLAB.
Cât spațiu este ocupat cu date audio în spațiul de lucru MATLAB?
Ce tip de date au datele audio?
Cum vă calculati cantitatea de memorie necesară din parametrii de înregistrare?
Care este dimensiunea fisierului myVoice.wav?
Câti octeți sunt utilizati în myVoice.wav pentru a înregistra date, altele decât datele audio în sine?

4.3. Manipularea semnalului audio: Scrieți un script MATLAB pentru a înregistra rostirea ta de "azi e ziua mea de naștere". Încercați să explicați efectul de redare observați după ce încercați următoarele operații asupra semnalelor audio:
Inmultiti semnalele audio cu -1.
Inversati semnalele audio pe axa timpului.
Inmultiti semnalele audio cu 10.
Înlocuiți fiecare esantion cu rădăcină sa pătrată.
Înlocuiți fiecare esantion cu pătratul său.
Taierea/limitarea semnalului, astfel încât eșantioanele din gama [-0.5, 0.5] sunt setate la zero.
Modificati forma de undă astfel încât eșantioanele din intervalul [-0.5, 0.5] sunt setate la zero, iar eșantioanele din afara gamei

de [-0.5, 0.5] sunt mutate spre zero, cu cantitatea de 0,5.

4.4. Experimente pe rata de eșantionare: Scrieți un script MATLAB pentru a înregistra dvs. de rostire "numele meu este ***", cu o rată de eșantionare de 16 kHz și 8-biți rezoluție sau folositi un fisier existent. Încercați să reeșantionati semnalele audio prin scăderea ratelor de eșantionare la 11 kHz, 8 kHz, 4 kHz, 2 kHz, 1 kHz, și așa mai departe. La care rata de esantionare începem să avem dificultăți în a înțelege conținutul enunțului?

4.5. Experimente cu adăugarea de zgomot: Scrieți un script MATLAB pentru a înregistra rostirea "numele meu este ***", cu o rată de eșantionare de 8 kHz și 8-biți rezoluție. Putem adăuga zgomot la semnalele audio prin utilizarea secventei următoare:

```
k = 0.1;
y2 = y + k*randn(length(y), 1);      % Aduna zgomot
sound(y2, 8000);                     % Playback/redare
plot(y2);
```

Creșteti valoarea lui k cu 0,1 de fiecare dată și răspundeți la următoarele întrebări.
a) La ce valoare a lui K începem să avem dificultăți în a înțelege conținutul redat?
b) Se trasează formele de undă la valori diferite ale lui k. La ce valoare a lui k vom începe să avem dificultăți în a identifica perioada fundamentală ?

5. Bibliografie

http://mirlab.org/jang/books/audioSignalProcessing/

http://mirlab.org/jang/matlab/toolbox/sap/

http://mirlab.org/jang/matlab/toolbox/utility/

http://neural.cs.nthu.edu.tw/jang/matlab/toolbox/sap/html/sap/wavReadInt.html