Semnale și sisteme

Lucrare de laborator nr. 8

Filtrarea semnalelor audio utilizand filtre FIR

Filtre FIR

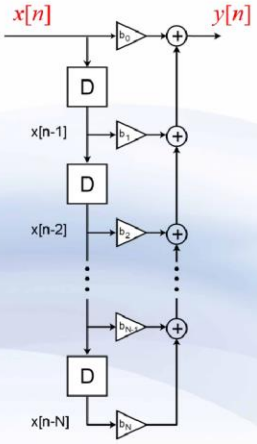
- De obicei, se cere ca semnalele discrete să aparțină clasei ℓ_2 (de energie finită): $\sum_{n=-\infty}^{\infty} |x_n|^2 < \infty$
- Un filtru discret $\mathbf{\Phi}$ poate fi văzut ca un *operator liniar* mărginit $\mathbf{\Phi}: \ell_2 \to \ell_2$ care este *invariant în timp*
- Acțiunea filtrului poate fi reprezentată ca și convoluție:

$$\mathbf{\Phi}x = h * x$$

- Filtrul cu răspuns finit la impuls (Finite Impulse Response - FIR) are un număr finit de coeficienți h_n diferiți de zero
- Filtru discret cauzal: $h_j = 0$ pentru $j < 0$
- Un filtru FIR cauzal este complet determinat de vectorul cu valori ale răspunsului la impuls

$$h = (h_0, h_1, \ldots, h_N)$$

unde N este cel mai mare număr întreg pozitiv astfel încât $h_N \neq 0$ (filtrul are N+1 poziții)



$$y[n] = \sum_{k=0}^{N} b_k \, x[n-k]$$

Filtrarea semnalelor audio

Functii Matlab utile:

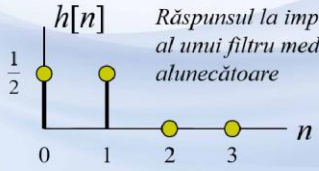
- *fir1* – genereaza coeficientii unui filtru FIR
- *filter* – filtreaza un semnal utilizand coeficientii unui filtru
- *conv* – calculeaza convolutia dintre doua semnale
- *audioread* – incarcarea unui fisier audio in Matlab
- *sound* – redarea unui sunet

Toolbox-ul Matlab *DSP System Toolbox*:

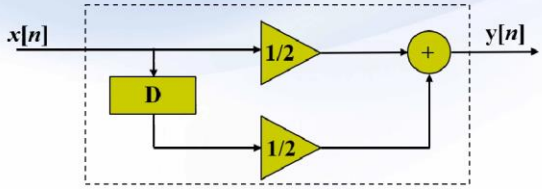
- *dsp.FIRFilter*
- *dsp.MovingAverage*

Exemplu - Filtru FIR

- $y[n] = \sum_{k=-\infty}^{\infty} h[k]\ x[n-k]$ descrie ieșirea $y[n]$ corespunzătoare intrării $x[n]$
- Dacă doar câțiva termeni ai lui $h[n]$ sunt diferiți de zero atunci sistemul poate fi implementat cu ușurință



$$y[n] = h[0]\,x[n] + h[1]\,x[n\text{-}1]$$
$$= (\,x[n] + x[n\text{-}1]\,) / 2$$



Exemplu: Reducerea zgomotului utilizand un filtru medie alunecatoare

```
% generare semnal audio
Fs = 44100;
f_do = 261.6;
f_re = 293.6;
f_mi = 329.6;
t = 0:1/Fs:0.5-1/Fs;
do = sin(2*pi*f_do*t);
re = sin(2*pi*f_re*t);
mi = sin(2*pi*f_mi*t);
silence = zeros(size(t));
t = 0:1/Fs:2.5-1/Fs;
s = [do, re, mi, silence, do];
figure, plot(t,s)
sound(s, Fs)
pause(3)
```

```
% adaugare zgomot
z = 0.1*(2*rand(size(t))-1);
sz = s + z;
sound(sz,Fs)
pause(3)
```

```
% filtrare
M = 50;
h = 1/M*ones(1,M);
sf = conv(h, sz);
sound(sf,Fs)
```

Implementarea filtrului medie alunecatoare in Matlab

- Filtrul medie alunecatoare este un caz special al filtrului FIR. Ambele filtre au răspuns finit la impuls. Coeficientii filtrului medie alunecatoare constau intr-o secvență de valori de 1 scalate cu lungimea filtrului, în timp ce coeficienții filtrului FIR sunt proiectați pe baza specificațiilor filtrului, si, de obicei, nu sunt o secvență de valori de 1.
- Media alunecatoare a datelor de intrare este calculată utilizand o fereastră glisantă de lungime finită:

$$movAvg = \frac{x[n]+x[n-1]+\cdots+x[n-N]}{N+1},$$

N+1 este lungimea filtrului. Acest algoritm este un caz special al filtrului FIR cu vectorul coeficientilor $[b_0, b_1, ..., b_N]$:

$$FIROutput = b_0x[n] + b_1x[n-1] + \cdots + b_Nx[n-N]$$

Implementarea filtrului medie alunecatoare in Matlab

- Pentru calculul iesirii, filtrul FIR înmulțește fiecare esantion al semnalului de intrare cu un coeficient din vectorul $[b_0, b_1, ..., b_N]$ și insumeaza rezultatele. Filtrul medie alunecatoare nu utilizează operatia de inmultire. Algoritmul insumeaza toate esantioanele semnalului de intrare și înmulțește rezultatul cu 1 / *filterLength*.

Exemplu

```
filterlen = 40;
% crearea unui filtru FIR utilizand dsp.FIRFilter
% coeficientii filtrului, [b0, b1, ..., bN], se pun in al doilea parametru, dupa 'Numerator'
FIRfilter = dsp.FIRFilter('Numerator',ones(1,filterlen)/filterlen);
% crearea unui filtru medie alunecatoare utilizand dsp.MovingAverage
mvgAvgFilter = dsp.MovingAverage(filterlen);
% filtrarea unui semnal utilizand filtrele create anterior
input = randn(1024,1);
filterOutput = FIRfilter (input);
mvgAvgOutput = mvgAvgFilter(input);
figure, plot(input), hold on, plot(filterOutput, 'r'), plot(mvgAvgOutput, '--g')
```

Exemplu: Filtrarea unui semnal audio utilizand un filtru trece banda

```
% incarcarea unui semnal audio in Matlab, afisarea grafica a semnalului, redarea semnalului audio
load mtlb
y = mtlb; L=length(y); k = 0:L-1; k = k'; t = k/Fs;
sound(y,Fs)
figure, plot(t, y)

% adaugare zgomot
noise=0.5*sin(2*pi*1000/L*k);
y_z = y + noise;
sound(y_z,Fs)

% generarea unui filtru FIR trece banda
N = 100; Fpl = 100; Fph = 1000; Wbp = [Fpl Fph]./(Fs/2);
bpf = fir1(N, Wbp, 'bandpass');

outbpf = filter(bpf,1,y_z);
sound(outbpf,Fs)

yo = conv(bpf, y_z);
sound(yo,Fs)
```

Convoluția bidimensională și filtrarea imaginilor

Convolutia 2D – Cazul semnalelor continue

$$f(x,y) = h(x,y) * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x',y')g(x-x',y-y')dx'\,dy'$$

Convolutia 2D – Cazul semnalelor discrete

$$f[x,y] = h[x,y] * g[x,y] = \sum_{i=-n}^{n}\sum_{j=-n}^{n} h[i,j]g[x-i,y-j]$$

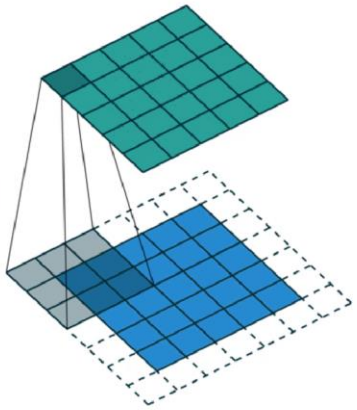
Filtrarea liniară a imaginilor permite aplicarea de diverse efecte asupra imaginilor: efecte de "netezire", efecte de contrastare (detecție muchii, evidențiere muchii), reducere zgomot, etc. prin specificarea unui răspuns la impuls $h[i, j]$ al unui filtru liniar.

Pentru a filtra spațial o imagine utilizând convoluția 2D, este necesară o reprezentare a filtrului $h[i, j]$ ca și matrice (numită nucleu de convoluție) și o reprezentare matriceală a imaginii discrete. Pentru aplicarea filtrului asupra imaginii, pentru fiecare pixel din imagine, se calculează o sumă de produse. Fiecare produs constă din valoarea pixelului curent sau a unui vecin al său înmulțită cu valoarea corespunzătoare din nucleul de convoluție conform formulei pentru convoluția discretă.

$$A_f[x,y] = h[x,y] * A[x,y] = \sum_{i=-n}^{n}\sum_{j=-n}^{n} h[i,j]A[x-i,y-j]$$

Pasii necesari pentru calculul valorii $A_f[x,y]$

1. se roteste nucleul de convolutie, h, cu 180°
2. se suprapune centrul nucleului de convolutie peste elementul de pe pozitia (x, y) din A
3. se inmultesc valorile din h cu valorile din A peste care s-a suprapus h
4. se insumeaza produsele obtinute la pasul 3



h

2	1	0
0	2	2
2	1	0

A

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

A_f

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

- Reguli pentru nucleul de convoluție:
  - dimensiunea nucleului trebuie să fie impară, de exemplu 3x3, 5x5 și 7x7.
  - dacă suma elementelor este 1, imaginea rezultată va avea aceeași luminozitate ca și imaginea inițială.
  - dacă suma elementelor este mai mare decât 1, va rezulta o imagine mai luminoasă, iar dacă este mai mică decât 1 va rezulta o imagine mai întunecată.
- Probleme de implementare:
  - re-normalizare (ex. pentru imagini pe 8 biți, valorile pixelilor trebuie să fie în intervalul [0…255]): se scalează rezultatul convoluției în acest interval **sau** se aleg coeficienții nucleului de convoluție în așa fel încât rezultatul produs să fie în acest interval
  - pentru procesarea pixelilor aflați la marginea imaginii, se poate utiliza: bordarea cu pixeli de valoare zero **sau** nu se procesează pixelii de la margine – li se atribuie valoarea 0 **sau** se copiază pixelii de la margine cu aceeași valoare ca în imaginea inițială **sau** se trunchiază imaginea finală **sau** se aplică o metodă de "înfășurare" a imaginii:

```
M - image width
if x < 0 then
    x = x+M
else if x >= M then
    x = x-M
end
```

Filtrarea trece-jos

Un filtru trece-jos permite trecerea frecvențelor joase, sau a datelor care nu se schimbă brusc de la un pixel la pixelul vecin, atenuând frecvențele înalte, sau datele care se schimbă rapid de la un pixel la pixelul vecin. Aplicând un filtru trece-jos unei imagini, se va obține o imagine "încețoșată" sau "netezită" (*blurred* sau *smoothed*). Filtrarea trece jos se utilizează pentru a elimina detaliile fine și a evidenția caracteristicile mari, și pentru reducerea zgomotului. Exemple de filtre trece jos:

Filtre de mediere

$$h[x,y]=\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad h[x,y]=\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

Filtru Gaussian

$$h[x,y]=e^{-\frac{1}{2}\frac{(x^2+y^2)}{\sigma^2}} \qquad h[x,y]=\frac{1}{256}\begin{bmatrix}1&4&6&4&1\\4&16&24&16&4\\6&24&36&24&6\\4&16&24&16&4\\1&4&6&4&1\end{bmatrix}$$

Filtrarea trece-sus

Filtrele de tip trece-sus atenuează componentele imaginii cu frecvențe joase. Sunt utilizate pentru detectarea muchiilor, evidențierea liniilor și a muchiilor, etc.

Filtru Sobel

$$h_v[x,y]=\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \qquad h_h[x,y]=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

Filtru Prewitt

$$h_v[x,y]=\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix} \qquad h_h[x,y]=\begin{bmatrix}1 & 1 & 1\\ 0 & 0 & 0\\ -1 & -1 & -1\end{bmatrix}$$

Reprezentarea imaginilor în Matlab

Moduri de reprezentare a unei imagini in Matlab:

- prin matrici de dimensiune (N x M x 3) sau (N x M x 4) în funcție de modelul de culoare: RGB, CMYK, HLS etc.
- utilizând o matrice 2D și o paletă de culori – reprezentare indexată.

Funcții pentru importarea/exportarea, vizualizarea imaginilor in Matlab: *imread*, *imwrite*, *imshow*, *imagesc*.

Utilizând funcția *fspecial* (filtre speciale) se pot construi diverse filtre 2D.

Exercitii

1a. Calculati analitic convolutia dintre matricea imaginii A si nucleul de convolutie h (pentru pixelii aflati la marginile imaginii utilizati metoda de "infasurare").

$$A[x,y]=\begin{bmatrix}0 & 16 & 32 & 48\\ 16 & 0 & 16 & 32\\ 32 & 16 & 0 & 16\\ 48 & 32 & 16 & 0\end{bmatrix} \qquad h[x,y]=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

b. Realizati convolutia dintre matricea imaginii A si nucleul de convolutie h definite la punctul a. in Matlab utilizand functia *imfilter* cu optiunile *conv* si *circular*. Afisati imaginea initiala si imaginea filtrata in Matlab (utilizati *imagesc* si *colormap gray*).

c. Scrieti o functie in Matlab pentru calculul convolutiei dintre o imagine 2D si un filtru 2D. Pentru pixelii aflati la marginile imaginii utilizati metoda de "infasurare". Utilizati aceasta functie pentru a calcula convolutia dintre matricea imaginii A si nucleul de convolutie h definite la punctul a.

Exercitii

2. a) Importati si afisati imaginea 'peppers.png' (imagine disponibila in exemplele Matlab)

b) Realizati convolutia dintre imagine si nuclee de convolutie prezentate in continuare (utilizati *imfilter* cu optiunile *conv* si *circular*). Afisati fiecare dintre imaginile filtrare si comparati-le cu imaginea initiala.

```
% Blur Gaussian filter
h = 1/256*[1 4 6 4 1; 4 16 24 16 4; 6 24 36 24 6; 4 16 24 16 4; 1 4 6 4 1];
% Motion blur
h = zeros(9,9);
for i = 1:length(h)
   for j = 1:length(h)
      if (i == j)
         h(i,j) = 1;
      end
   end
end
h = h/9;
```

```
% evidentiere muchii (sharpening)
h = [-1 -1 -1; -1 9 -1; -1 -1 -1];
% filtre de tip Laplacian pentru detectia liniilor si muchiilor
h = [-1 -1 -1; -1 8 -1; -1 -1 -1];   % detectie muchii
h = [-1 -1 -1;  2 2  2; -1 -1 -1];   % detectie linii orizontale
h = [-1  2 -1; -1 2 -1; -1  2 -1];   % detectie linii verticale
h = [-1 -1  2; -1 2 -1;  2 -1 -1];
h = [ 2 -1 -1; -1 2 -1; -1 -1 2];
% filtre Sobel
h = [1 2  1; 0 0  0; -1 -2 -1]; % evidentiere muchii orizontale
h = [1 0 -1; 2 0 -2; 1  0 -1]; % evidentiere muchii verticale
% filtre Prewitt
h = [1 1  1; 0 0  0; -1 -1 -1]; % evidentiere muchii orizontale
h = [1 0 -1; 1 0 -1;  1  0 -1]; % evidentiere muchii vertical
```

c. Adaugati zgomot la imaginea originala (utilizati *imnoise* cu optiunea 'salt & pepper').

d. Exportati imaginea obtinuta la punctul c. sub numele de „photo1_noise.jpg" (*imwrite*).

e. Filtrati imaginea cu zgomot utilizand urmatorul nucleu de convolutie:

```
h = 1/9*[1 1 1; 1 1 1; 1 1 1].
```

Mariti dimensiunea filtrului. Ce observati?