UE MU4RBI08-audio
Compte-rendu de TP 3 :
Reconstruction du signal audio : TFCT inverse

KADI - koussaila
DA ROCHA MARTINS - Mickael
Mention : Automatique Roborique (ISI)

Introduction

Dans ce TP on va s'intéresser à la restitution du signal à partir de la matrice TFCT en utilisat la méthode de Overlap-add (OLA) et en implémtant la trasformée à court terme inverse ITFCT. et analyse du signal restitué. L'algorithme de OLA nous permet de récupérer le signal de départ on appliquant un processus inverse à la TFCT, c'est à dire, on détermine la TFD inverse pour chaque trame (chaque colonne de la matrice TFCT) pour en déduire les valeurs des trames de départ, et à la fin pour restituer le signal, il faudra additionner toutes les trammes avec un chevauchement en décalant à chaque fois d'un pas de Nhop, exactement l'inverse, car au moment de l construction de la TFCT, on a eu un chevauchement au moment du découpage du signal en trames.

Partie 1 : Implémentation de la TFCT inverse par Overlap-Add (OLA)

1. allocation de la mémoire pour le signal restitué : Y=Nhop * (L-1) + 2*Nhop
   on veut que le signal reconstruit a la même longeur que le signal de départ, donc le signal de départ à une longeur len(X). on va perdre quelques points ou échantillons lors du découpage du signal en trames, donc le signal Y reconstruit va avoir un nombre de point inférieur à celle du signal d'origine X
2. La reconstruction de yl de chaque trame de la matrice FTCT et cela se fait par la trasformée discrete inverse TFD de chaque colonne de Xmat. donc yl il va avoir le même nombre de point que les colonnes de Xmat, qui ont un nombre de points fréquentiels Nfft
3. On sommes toutes les trames pour reconstruire le signal Y
   chaque trame est décalé de (l-1)*Nhop telque : l E [0, L]
4. On normalise le signal Y reconstruit en le dévisant sur K avec K la somme de tout les échantillonnons de w[n] divisé par Nhop
5. On test l'algorithme sur le audio "sound.wav"
   Nwin=4096 Nfft=Nwin Nhop=Nwin//2 avec ces paramètres, on a L=82 trames, et M= 4096 bins on une petite pertes de points temporels en calculant L
   L= [(longueur du signal d'entrée)/Nwin]*2 - 1 ce qui nous donne L= 82,85... et cette virgule sont des points temporels perdu car on ne peut pas avoir une trame complète.

Partie 2 Principe d'incertitude temps-fréquence

1. 7. Charger le signal audio bruité mix.wav. Calculer la TFCT du signal bruité X à l'aide de l'algorithme implémenté au TP2, en choisissant les paramètres N win et N hop de manière pertinente.
   Nwin=1024
   Nfft=1024
   Nhop=Nwin/2
   c'est choix sont pris car Nwin/Fs(fréquence d'échantillonnage)=temps de la trame. ça nous permet.
   On a testé plusieurs longueures de fenêtres, exemple 500, 1024, 2048 et on a trouvé la meilleure c'était 1024.
   on calcul la TFTC avec Nwin=1024
2. 8. Commenter le spectre du signal obtenu :
   1- le tracé est sur le notebook.
   2- commentaire : on constate que le signal de la parole commence après quelques trames, c'est à dire de la trame 0 à la trame 7, on a que le bruit.
3. 9. on constate que le spectogramme obtenu après le débruitage,

---

Xclean= X - Xbuit

---

et ce calcul peut nous donner des valeurs négatives, ce qui n'est pas logique pour un spectre. pour cela on devra faire un redréssement dans les questions suivantes.

4. 10. il est important de faire le redréssement du signal pour ne pas avoir des valeurs négatives dans notre spectre.
5. 11. On ne peut pas reconstruire le signal à partir que de Xclean, car on doit prendre en compte les valeurs de l'argument, pour ne pas avoir les décalages (dphasage) donc ce qui conduit à avoir des retards sur le signal.
6. 12. On a réussi à retirer le bruit, mais, on a des piques d'amplitudes très élevés. car lors de l'extractions du bruit, on a extrait quelques points du signal désiré, ce qui conduit à avoir des piques d'amplitude élevés et que le signal obetenu n'est plus aussi lisse. ce qui fait entendre des sons élevés en certains instants.
7. 13. On entend plus le bruit au niveau de l'ecoute, mais le son a perdu en amplitude.

pour le code

A Annexes

A.1 Code

Partie 1:

```
#import des bibliotheques
```

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav
from math import *
import math
import scipy

#TFCT:
```

```
## la TFCT


def tfct(x, Nwin, Nhop, Nfft, fe):

    # L nombre de trame     extraire dans xvect une partie du signal
    L=int((len(x)-Nwin)/Nhop) +1
    # M nombre minimal de points fr quentiels utiles pour chaque spectre
    M=Nfft

    # initialisation de la matrice xmat de taille Mx
    xmat=np.zeros((M,L))
    sigMat = []
    colonne=0
    i=0
```

```
    # fenetre de hamming de taille Nwin
    hamming=np.hamming(Nwin)


    for l in range(L):
        for j in range(int(i),int(i+Nwin)):
             #on recupere la trame correpondant i+Nwin trame
            sigMat.append(x[j])

        # on multiplie notre fenetre par les Nfft points de notre signal x
        TFD=np.fft.fft(sigMat*hamming)
         #le spectre
        xmat[:,colonne]=np.real(TFD)

        # on incremente la variable qui change les collonnes de la matrice Xmat
        colonne+=1
        # on d clale d'une demi trame, soit le pas d'avancement Nhop
        i=i+Nhop
        # on efface la trame ancienne
        sigMat=[]


    f=np.fft.fftfreq(len(x),1/fe)   #fr quence
    t=len(x)/fe                     #vecteur temps
    print(M,L)
    return xmat,t,f

>>> execution de la TFCT: >>>

Nwin = 4096
Nfft = Nwin
Nhop = Nwin//2 # pas d'avancement de chaque trame.

fe,xvect=wav.read("sound.wav")
#normalisation du signal xvect:

xmat,temps,freq=tfct(xvect,Nwin,Nhop,Nfft,fe)
print(xmat.shape)

plt.figure()
plt.imshow(20*np.log10(xmat),aspect ='auto', origin='lower',cmap='jet')
plt.xlabel("trames du signal audio")
plt.ylabel("fr quence")
plt.show()


#ITFCT (TFCT inverse):
-----------------------------------------------------------

def itfct(xmat,Nhop,Nfft,fs):
    #etape 1
    Y=np.zeros(Nhop*(xmat.shape[1]-1)+Nfft) # cr ation du vecteur rempli de 0
    #etape 2
    yl=np.zeros(Nfft)
```

```
    e=0
    for i in range(xmat.shape[1]):
        yl=np.fft.ifft(np.transpose(xmat[:,i]))  # TFD inverse de chaque trame
        for j in range(e,e+Nfft):
            Y[j]+=yl[j-e] # somme des
        e+=Nhop  # d calage

    k=np.sum(np.hamming(Nfft))/Nhop
    Y=Y/k #normalisation
    temps=np.linspace(0,len(Y)/fs,len(Y))
    return temps, Y

>>> execution de la ITFCT sur la matrice Xmat generee par la TFCT>>>
#lecture du fichier sound.wav
fe,x=wav.read("sound.wav")

#parametres
Nwin=4096
Nfft=Nwin
Nhop=Nwin//2

#TFCT
xmat, ts,fs= tfct(x, Nwin, Nhop,Nfft, fe)

#ITFCT
te,ye=itfct(xmat,Nhop,Nfft,fe)
print("la taille du signal apr s itfct est :",len(ye),len(x))

#visualisation des deux signaux original et reconstrui
plt.figure()
plt.plot(x/2**15,'y',label='signal origine')
plt.plot(ye/2**15,label='signal restitu ')
plt.title("representation temporelle du signal avant et apres itfct ")
plt.ylabel("|Signal Audio|")
plt.xlabel("temps(s)")
plt.legend()
plt.show()


#Erreur de construction:
-----------------------------------------------------
E_quad=np.sum((x[0:len(ye)]-ye)**2)/len(ye)
print(E_quad)
>>>2054607.7988644708

on constate que l'erreur est tres grande, car au moment de la reconstruction, on


-------------------------------------------------------------------------------
Partie 2:
-------------------------------------------------------------------------------


#QST 7
#question 7
```

```
f_mix,x_mix=wav.read("mix.wav")
t=np.arange(start=0, stop=len(x_mix)/f_mix, step=1/f_mix)
print("nb echantillons :",len(x_mix))
print("fr quence echantillonage :",f_mix)
display(Audio(x_mix,rate=f_mix))


#--------------------
Nwin=1024
Nfft=Nwin
Nhop=Nwin//2

plt.figure()
plt.plot(t,x_mix, label='mix.wav')
plt.legend()
plt.show()

plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.imshow(20*np.log(abs(xmat)),aspect ='auto', origin='lower',cmap='jet')
plt.xlabel("trames du signal audio")
plt.show()

#-----spectograme:---
xmat,temps,freq=tfct(x_mix,Nwin,Nhop,Nfft,f_mix)

#----- les trames de silences: dans la r gion de la 60 me  trame-----
t=np.arange(start=0,stop=(len(x_mix)/f_mix),step=1/f_mix)
trame=[0*Nwin,10*Nwin]


plt.title("zone de silence")
plt.plot(t[trame[0]:trame[1]],x_mix[trame[0]:trame[1]], label='mix.wav')
plt.plot(t[0*Nwin:7*Nwin],x_mix[0*Nwin:7*Nwin], label=' 7 trames de silence')
plt.legend()
plt.show()

#QST9:

#-----le spectre d'amplitude |Xnoise| ----------
Xnoise=x_mix[0*Nwin:1*Nwin]  # on prend une trame dans la r gion de silence

TFD1=np.fft.fft(Xnoise)
spectre_Xnoise=np.fft.fftshift(TFD1)
spectre_Xnoise=np.absolute(spectre_Xnoise)
freq1=np.arange(-f_mix/2,f_mix/2,f_mix/len(spectre_Xnoise))


TFD2=np.fft.fft(x_mix)
spectre_xmix=np.fft.fftshift(TFD2)
spectre_xmix=np.absolute(spectre_xmix)
freq2=np.arange(-f_mix/2,f_mix/2,f_mix/len(spectre_xmix))
```

```
#tracer le spectre
plt.figure()
plt.plot(freq2,spectre_xmix,label="|Xmix|")
plt.title("spectre Xmix")
plt.ylim(0,2.5*10**7)
plt.legend()

plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
plt.plot(freq1,spectre_Xnoise,label="|Xnoise|")
plt.title("r gion de silence")
plt.ylim(0,2.5*10**7)
plt.legend()

plt.subplot(1,2,2)
plt.plot(freq1,spectre_Xnoise,label="|Xnoise|")
plt.title("r gion de silence zoom")
plt.ylim(0,0.1*10**7)
plt.legend()
plt.show()


#QST 9:
#question 9:

# ---- soustraction --------
xmat_mix, ts,fs= tfct(x_mix, Nwin, Nhop,Nfft, f_mix)

def bruitExtract(xmat,bruit):
    xmat_clean=np.zeros((xmat.shape[0],xmat.shape[1]))
    for i in range(xmat.shape[1]):
        xmat_clean[:,i]=xmat[:,1]-bruit
    return xmat_clean

xmat_clean=bruitExtract(xmat_mix,spectre_Xnoise)

#----spectogramme----------
plt.figure(figsize=(12,5))
plt.imshow(20*np.log(abs(xmat_clean)),aspect ='auto', origin='lower',cmap='jet')
plt.xlabel("trames du signal audio")
plt.ylabel("fr quence")
plt.show()

#QST 10:

def bruitExtract_redressement(xmat,bruit):
    xmat_clean=np.zeros((xmat.shape[0],xmat.shape[1]))
    x=np.zeros((xmat.shape[0],1))
    for i in range(xmat.shape[1]):
        x=xmat[:,i]-bruit/2
        for j in range(len(x)):
            if x[j]<0:
                x[j]=0
        xmat_clean[:,i]=x
```

```
    return xmat_clean

xmat_clean_redresse=bruitExtract_redressement(xmat_mix,spectre_Xnoise)

#-----spectogramme------------
plt.figure(figsize=(12,5))
plt.imshow(20*np.log(abs(xmat_clean_redresse)),aspect ='auto', origin='lower',cma
plt.xlabel("trames du signal audio")
plt.ylabel("fr quence")
plt.show()

#QST 11:

#------ spectogramme ---------
te,y=itfct(xmat_clean_redresse,Nhop,Nfft,f_mix)

print("la taille du signal apr s itfct est :",len(ye),len(x))
plt.figure()
plt.plot(x_mix/2**15,'y',label='signal origine')
plt.plot(y/2**15,label='signal restitu ')
plt.title("representation temporelle du signal avant et apres itfct ")
plt.ylabel("|Signal Audio|")
plt.xlabel("temps(s)")
plt.legend()
plt.show()


#------- couter  le signal reconstrui et sans bruit --------
display(Audio(y,rate=f_mix))

#------ on recalcul l'erreur ---------
E_quad_clean=np.sum((x_mix[0:len(y)]-y)**2)/len(y)
print("erreur quadratique apr s clean:",E_quad_clean)

#QST 12:
def tfct_12(x, Nwin, Nhop, Nfft, fe):

    # L nombre de trame    extraire dans xvect une partie du signal
    L=int((len(x)-Nwin)/Nhop) +1
    # M nombre minimal de points fr quentiels utiles pour chaque spectre
    M=Nfft

    # initialisation de la matrice xmat de taille Mx
    xmat=np.zeros((M,L))
    sigMat = []
    colonne=0
    i=0

    # fenetre de hamming de taille Nwin
    hamming=np.hamming(Nwin)


    for l in range(L):
```

```
            for j in range(int(i),int(i+Nwin)):
                #on recupere la trame correpondant i+Nwin trame
                sigMat.append(x[j])

            # on multiplie notre fenetre par les Nfft points de notre signal x
            TFD=np.fft.fft(sigMat*hamming)
             #le spectre
            xmat[:,colonne]=((np.real(TFD)+np.conj(TFD))**2)*np.exp(np.angle(TFD))

            # on incremente la variable qui change les collonnes de la matrice Xmat
            colonne+=1
            # on d clale d'une demi trame, soit le pas d'avancement Nhop
            i=i+Nhop
            # on efface la trame ancienne
            sigMat=[]


        f=np.fft.fftfreq(len(x),1/fe)   #fr quence
        t=len(x)/fe                     #vecteur temps
        print(M,L)
        return xmat,t,f

        # refaire la TFCT sur le signal pour r cup rer m me l'argument (r cup re

xmat_mix,t,f = tfct_12(x_mix, Nwin, Nhop, Nfft, f_mix)

# faire le d bruitage et le redressement: ---

xmat_clean_redresse=bruitExtract_redressement(xmat_mix,spectre_Xnoise)

# TFCT inverse: ----------------------------------
te,y=itfct(xmat_clean_redresse,Nhop,Nfft,f_mix)

print("la taille du signal apr s itfct est :",len(ye))
plt.figure()
plt.plot(x_mix/2**15,'y',label='signal origine')
plt.plot(y/2**15,label='signal restitu ')
plt.title("signal audio d bruit ")
plt.ylabel("|Signal Audio|")
plt.xlabel("temps(s)")
plt.legend()
plt.show()


#------ couter  le signal reconstrui et sans bruit --------
display(Audio(y,rate=f_mix))



************fin**************
```