Analisis de .Wav

Edwin Mahecha

Noviembre, 2014

1 Introducción

Los siguiente códigos fueron elaborados con la finalidad de realizar un analisis a los archivos ejemplo de sonido proporcionados por la NASA para el proyecto Jove. Estos archivos de sonido pueden ser descargados directamente de la página del proyecto `http://radiojove.gsfc.nasa.gov/`. Los datos tomados seran utilizados para ser confrontados frente a los datos de la NASA. Cabe resaltar que las unidades de RadioSkyPipe(programa suministrado) no van a ser muchas veces compatibles con los resultados de los archivos *.wav.

El programa está dividido en cinco secciones que se explicaran a continuación. El programa no es completamente universal, ya que es necesario el uso de librerias externas al la instalación básica de python, por lo cual se recomienda verificar que librerias son requeridas o instalar el paquete de anaconda disponible aquí: `https://store.continuum.io/cshop/anaconda/`.

2 Código

Este código nos permite realizar un análisis directo a archivos de audio. Hacendo uso de archivos de ejemplo proporcionados por la NASA, y otros archivos grabados localmente, permitieron el desarrollo de este análisis. En el repositorio de GitHub `https://github.com/SemilleroKonradLorenz/jove` se puede tener acceso a los códigos y archivos utilizados en este informe.

2.1 Importando Librerías

El programa en total importa 12 liberias , las cuales nos permiten el manejo gráfico y de datos en el programa.

```
import wave
import contextlib
import scipy
from pylab import*
import numpy as np
import math
from scipy.signal
import hann from scipy.fftpack import rfft
from scipy.io import wavfile
import matplotlib.pyplot as plt
from Tkinter import Tk
from tkFileDialog import askopenfilename
```

2.2 Selección del archivo

Para esto se hace uso de la librería Tkinter. Esta ejecuta un cuadro de dialogo en el cual se puede buscar el archivo en específico.

Si se ve con detalle, se puede apreciar la impelemntación anterior a Tkinter. Dicha implementación era limitada, ya que solo permitia buscar archivos en determinada carpeta, a diferencia de Tkinter que permite hacer un uso más extensivo.

```
#List files in Directory - Non Graphic mode
'''direct=os.chdir("/home/%s/jove/Sample Data"%getuser())
files=glob.glob('*.wav')
num=0
for x in files:
    print "- %s = %i"%(x, num)
    num+=1
var=input("Select File:")
#Select Files
if var>(num-1) or var<0:
    raise ValueError("No valid argument")'''

#Graphic Mode
Tk().withdraw()
fname = askopenfilename()
# show an "Open" dialog box and return the path to the selected file
print "File Selected: "+fname
```

2.3 Duración de Archivo

Esta implementación nos permite saber la duración del archivo. Esta es dada en segundos y es el resultado de:

$d = \frac{frames}{rate}$

```
#fname="/home/edwmapa/jove/Sample Data/"+files[var]
#File Duration frames/rate
with contextlib.closing(wave.open(fname,'r')) as f:
    frames = f.getnframes()
    rate = f.getframerate()
    duration = frames / float(rate)
    print "File length: %f Sec."%duration
```

2.4 Gráfica Amplitud Vs. Tiempo

Se hace uso de la librería scipy.io, para leer el archivo *.wav.

Esta librería lee el archivo *.wav, leyendo dos datos, sampleFeq y snd. snd contiene los puntos de la gŕafica(Amplitudes). Se puede apreciar que se hace

uso de la función shape() la cual nos permite determinar cuantos canales tiene el sonido. Los archivos de ejemplo únicamente poseen un canal, sin embargo otras grabaciones puedesn poseer varios canales, por eso unicamente se toma el primer canal.

```
#wavfile.read() returns rate and data in numpy array
input_data=wavfile.read(fname)
sampFreq, snd = input_data

snd = snd / (2.**15)
print "This is the shape of the sound (sample points, chanels sound):"
print(snd.shape)
#If the sound has more than one channel, only load first channel
if len(snd.shape)!=1:
    snd=snd[:,0]
#Arange of ms
samplePoints=snd.shape[0]
timeArray = arange(0,float(snd.shape[0]), 1)
timeArray = timeArray / sampFreq
timeArray = timeArray
#scale to seconds
maxV=(np.amax(snd))+.05#Max Value of snd
minV=(np.amin(snd))-.05
#yDuration yduration = np.array(duration)
yduration = np.repeat(yduration,2)

%matplotlib inline
plt.figure(figsize=(20,10))
plt.plot(timeArray,snd,'b')
plt.plot(yduration,np.linspace(minV,maxV, 2), "r--")#Limit line
plt.ylim(-maxV,maxV)
plt.title("File: "+fname)
ylabel('Amplitude')
xlabel('Time (sec.)')
```

La amplitud dada en esta gráfica está dada en volts (v), y los valores graficados estan en el orden de los milivolts.

2.5 Histograma

El histograma básicamente permite ver las veces que una amplitud se repite a lo largo de todo el archivo.

```
#%matplotlib inline
plt.figure(figsize=(20,10))
plt.hist(snd, bins=200)
xlabel('Amplitude')
ylabel('Times Repeated')
```

3 FFT (Fast Fourier Transform)

La Transformada Rápida de Fourier, y más específicamente la Transformada de Fourier Discreta, nos permite tomar una seie de datos de longitud finita, para dar una representacion del dominio de la frecuencia. La siguiente fórmula corresponde a la Transformada Discreta:

$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n]\,,$$

En la formula N es una seuencia de números complejos (X_0, ...,X_{N-1}). Esta secunecia es transformada en otra secuencia de N valores por medio de la ecuación anterior.
La DFT busca hallar $y\,[k]$, partiendo de $X\,[n]$.
$y\,[k]$es la frecuencia resultante.

```
input_data1 =wavfile.read(fname)
audio1 = input_data1[1]
#Sampling samp=2048
#Here we check the sound shape. We cant work with the
    sound if th sound has more than one channel.
if len(audio1.shape)!=1:
    audio1=audio1[:,0]
#Hanning window
window=hann(samp)
audio1 = audio1[0:samp] * window
#Fast Fourier Transform fft
mags= abs(rfft(audio1))
#dB values
mags= 20 * scipy.log10(mags)
#To 0 dB, set max mags to zero
mags -=max(mags)
#plot
plt.figure(figsize=(20,10))
plt.plot(mags, label="Graph")
ylabel("Magnitude $(dB)$", size =30)
xlabel("Frequency Bin", size=30)
plt.legend()
plt.title("Flute Spectrum", size=30)
plt.show()
```

En el códgo tambien se pude apreciar el uso de una esala logarítmica con el fin de calular decibles(dB). Esta escala coresponde a la ecuación:

$$G_{dB} = 20log_{10}(\frac{V_1}{V_0})$$

Esta formula es aplicada en circuitos electricos, en donde V_0 es un voltaje de referencia, V_1 es un voltaje dado.

4 Las Gŕaficas

4.1 Amplitud Vs. Tiempo

En la figura se puede apreciar como se ve graficado el audio de uno de los archivos de ejemplo proporcionados por la NASA.

Se puede apreciar que las unidades de Amplitd van de -1 a 1. La amplitud va dada en volts, y se trabaja en el rango de los milivoltios(mV), por ello es que se puede hacer una conversión a decibeles.

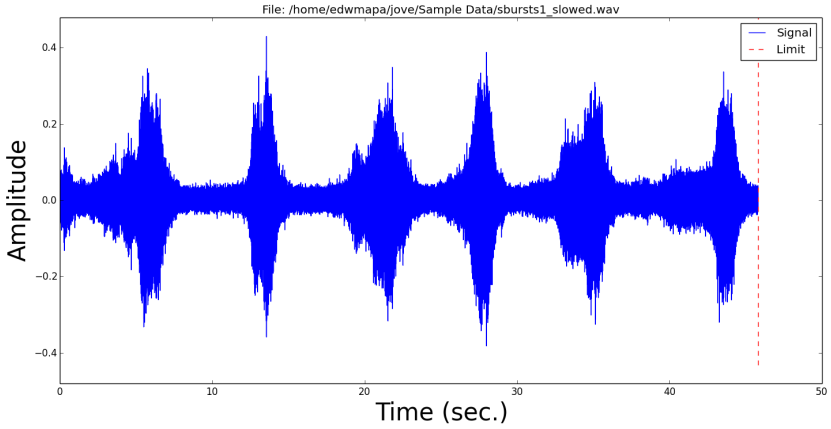

Figure 1: Amplitud Vs. Tiempo

Esta gráfica nos brinda información de como la señal es interpretada por el computador, aún así no se ha realizado ningún análisis de datos respecto al *.wav.

4.2 Histograma

El histograma se realiza con el objetivo de identificar las veces que se repite una amplitud, en este caso nos ayuda a descartar algunas interferencias, debido a que usualmente las amplitudes más bajas corresponde a interferencias del ambiente como lo pueden ser:

- Líneas de corriente (usualmente sus frecuencias oscilan entre los 50-60 Hz).
- Emisoras de radio, teléfonos, etc.
- Eventos naturales como lo son rayos, truenos, etc.

- Otros.

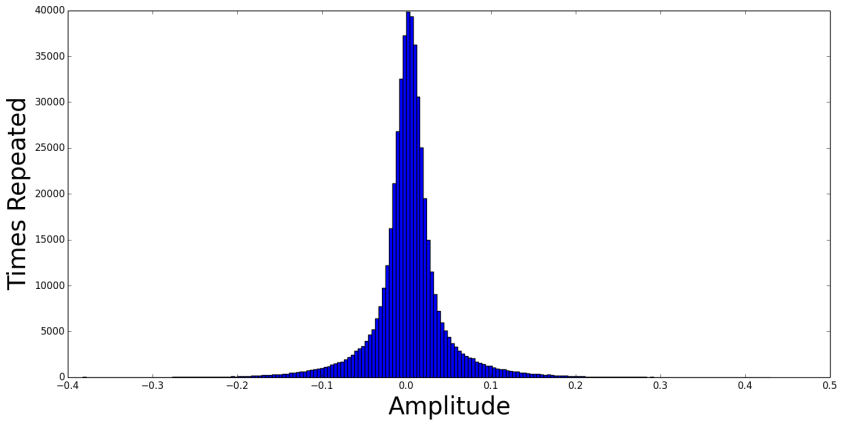

Figure 2: Histograma

4.3 dB Vs. Frecuencia

En esta gráfica se hace uso de la Transformada Rápida de Fourier (FFT) y la conversión logaritmica para decibeles en el caso de los milivoltios(mV). En este caso, el voltaje de referencia $V_0 = 1V$, debido a que la escala es trabajada desde 1 a -1, por lo cual la formula quedaría de la siguiente forma:

$$G_{dB} = 20log_{10}(\frac{VoltajeRecibido}{1V})$$

La siguiente tabla muestra los valores más comunes:

dB	V
0	1.0
-6	0.5
-12	0.25
-18	0.125
-24	0.0625

Table 1: Tabla Valores

La gráfica nos permite conocer las frecuencias del sonido junto con sus valores en decibeles, esto nos es sumamente util para discriminar valores que se consideren interferencia siempre y cuando conozcamos sus frecuencias, como es el caso de líneas de corriente o las emisoras de rádio.

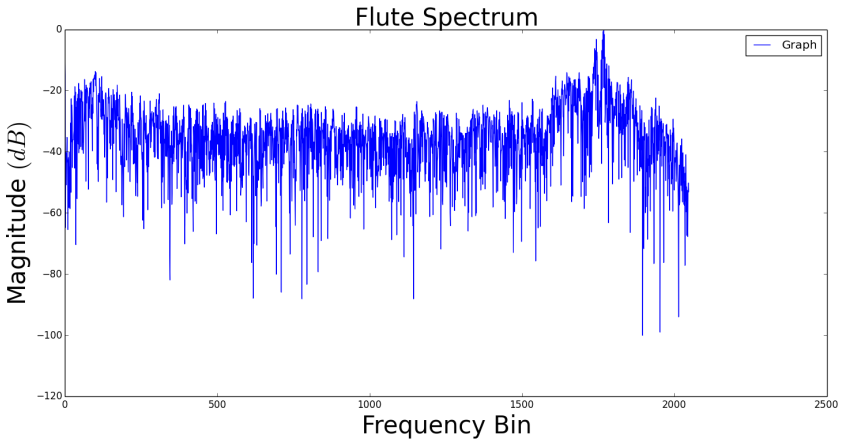

Figure 3: dB Vs. Binario de Frecuencias

5 Referencias

- http://lac.linuxaudio.org/2011/download/python_for_audio_signal_processing.pdf
- http://stackoverflow.com/questions/9690413/trying-to-use-fft-to-analyze-audio-signal-in-python
- http://samcarcagno.altervista.org/blog/basic-sound-processing-python/?doing_wp_cron=1411220939.3
- http://lac.linuxaudio.org/2011/papers/40.pdf