Práctica 1 - Análisis por bloques

1st Jaume Colom Hernandez
Universitat Politècnica de Catalunya - ETSETB
Processament audio i veu
Barcelona, España
jaumecolomhernandez@gmail.com

2nd Miquel Martinez Blanco
Universitat Politècnica de Catalunya - ETSETB
Processament audio i veu
Barcelona, España
mbmiquel@gmail.com

***Abstract*—En esta práctica vemos una introducción al laboratorio de PAV. Realizamos un análisis de señal de audio por bloques, calculamos la potencia, la amplitud y el ZCR. Para comprobar el resultado utilizamos el Wavesurfer con sus propias funciones para calcular estas métricas. Hemos obtenido un resultado satisfactorio en todos los casos.**

***Index Terms*—Procesado, audio, voz, C, programación.**

I. Introducción

Los objetivos de esta práctica son: recordar las bases de programación en C, como pasar argumentos por la linea de comandos, mapas de bits, aprender sobre *little o big endian*, analizar el formato `.wav`, introducir el procesado a bloques y calcular métricas temporales sencillas, integraremos en `Wavesurfer` las métricas calculadas y usaremos algún programa de representación gráfica.

Tenemos las siguientes partes principales en la memoria: Descripción teórica del diseño, aspectos de implementación, evaluación (descripción del marco experimental), discusión de los resultados, bibliografía y el apéndice con el código.

II. Descripción del diseño

A. Funciones del programa

El programa principal tiene la tarea de abrir el fichero de audio leerlo, calcular las métricas, y exportar los datos a fichero de datos y a imagen.

B. Esquema funcionamiento

1 Abrimos el fichero de texto.
2 Leemos la cabecera del fichero de audio.
3 Leemos un bloque de datos, procesamos y guardamos en un documento `.txt` .
4 Repetimos el paso 3 hasta que se acabe el fichero.
5 Exportamos los resultados por columnas para luego usarlos con el `Wavesurfer`.
6 Generamos los plots a partir de los datos exportados.

III. Aspectos implementación

A. Funciones de procesado a bloques

1) Potencia (dB): Calculamos la potencia definida como:

$$Power = 10log\frac{1}{N}\sum_{n=0}^{N-1} x^2[n] \tag{1}$$

2) Amplitud: Calculamos la amplitud definida como:

$$Amplitude = 10log\frac{1}{N}\sum_{n=0}^{N-1} x^2[n] \tag{2}$$

3) Zero Cross Rate: Calculamos el `zero coss rate` definido como:

$$ZCR = 10log\frac{1}{N}\sum_{n=0}^{N-1} x^2[n] \tag{3}$$

4) Ampliación: Potencia con ventana Hamming: Como ampliación implementamos una ventana de Hamming para calcular la potencia, con esto evitamos el ruido debido a los bordes del tramo.

$$Power = 10log\frac{\sum_{i=0}^{N-1} x[i] * w[i]}{\sum_{i=0}^{N-1} w^2[i]} \tag{4}$$

$$HammingWindow = 0.54 - 0.46 * cos(\frac{2 * pi * n}{M-1}) \tag{5}$$

B. Acceso y loop de lectura del fichero

Para leer el fichero utilizamos la función `fopen( file, mode)` y la función `fread( buffer, nbits, number, file)`. Luego al guardar los datos en un buffer utilizamos las funciones explicadas en el apartado anterior para calcular las métricas.

Finalmente las métricas se exportan a un fichero de texto mediante la función `fprint('str')`

C. Transformación del fichero exportado

Para utilizar los datos generados en el programa `Wavesurfer` se necesita transformar a un fichero donde únicamente estén los datos en una única columna, para hacer esto utilizamos comandos de Bash para generar el fichero.

```
cut -f 2 ./exports/exports >
./exports/power
```

D. Generación de los plots

Para generar los plots hemos elegido usar `Python` y la librería `Matplotlib`. Es un script muy simple que hace lo siguiente, lee el fichero que hemos exportado en el paso previo con al función `fromfile(file)` y lo guarda en un array de numpy. Después crea unos ejes, el plot y lo guarda en formato `.png`

E. Automatización del proceso

Durante el desarrollo de la práctica y posteriormente cuando estaba acabada vimos que la ejecución manual de todos los comando era lenta, así que creamos un script de `Bash` que lanza todos los pasos secuencialmente. De esta forma podíamos hacer un cambio y automáticamente veíamos toda la ejecución y donde fallaba.

IV. Evaluación

Los datos utilizados han sido únicamente un fichero de audio de los dos integrantes del grupo. En la grabación de voz los dos interlocutores han realizado pausas entre frases y se han utilizado gran variedad de sonidos y fonemas. A partir de estos datos se han calculado distintas características como la potencia, la amplitud y la frecuencia de corte en cero, posteriormente todos estos datos han sido exportados en distintos documentos de texto. En esta práctica se ha optado por representar los resultados de dos formas distintas: mediante la librería `Matplotlib` de `Python` tal y como se ha explicado en apartados anteriores y utilizando el panel de `Data Plot` del programa `Wavesurfer`. La utilización de `Wavesurfer` ha tenido un doble cometido, "plotear" los datos de una manera sencilla y sincronizada con la señal de audio y también comprobar que los datos calculados por nosotros coincidían con los que el propio `Wavesurfer` calcula.

V. Discusión de los resultados

En referencia a la Figura 1. analizando de arriba a abajo primero se encuentra la representación de la potencia calculada por el propio `Wavesurfer`, debajo se encuentra la representación de los valores de la potencia que se han calculado mediante nuestro programa en C. Como se puede observar las dos representaciones son muy similares en cuanto a forma pero tienen un nivel base diferente, esto es debido a que la representación que hemos calculado nosotros había utilizado los valores de la señal y posteriormente los había normalizado, de manera que se ha representado el contorno en dBFS, a diferencia del `Wavesurfer` que lo ha hecho en dB al utilizar la señal original. A la derecha adjuntamos los plots finales.

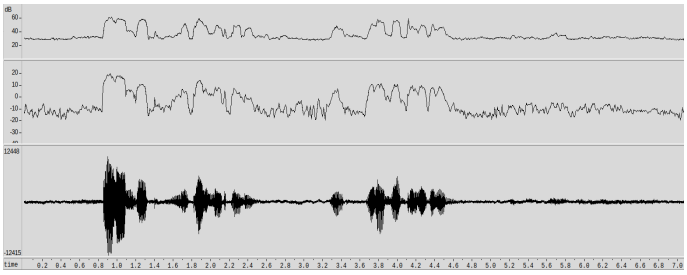

Fig. 1. Power plot en Wavesurfer.

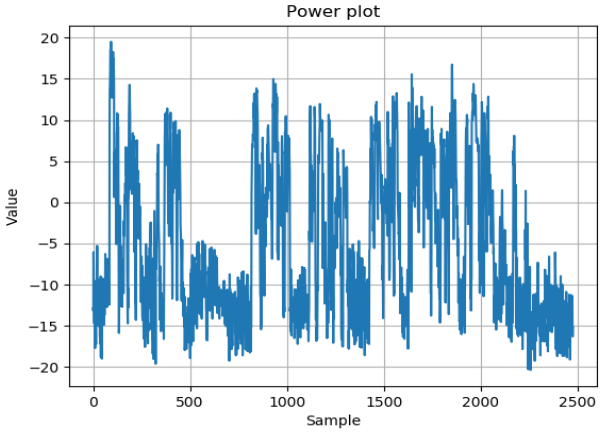

Fig. 2. Power plot

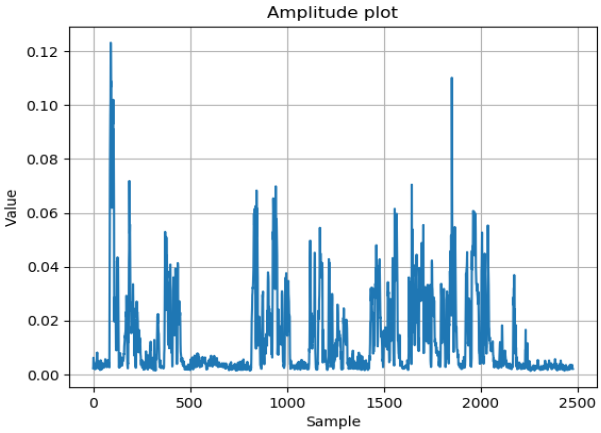

Fig. 3. Amplitude plot

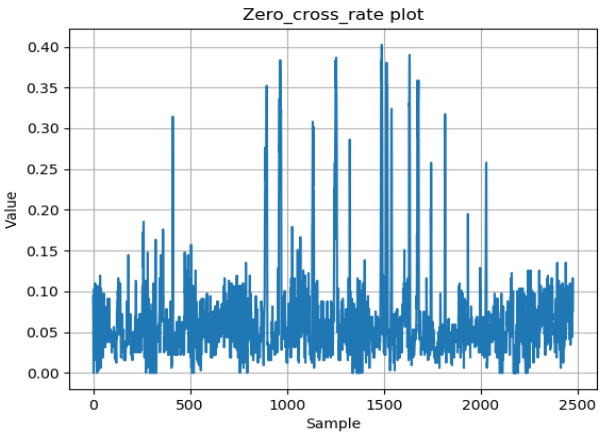

Fig. 4. Zero cross rate plot

VI. Conclusiones del trabajo

Se han podido recordar conocimientos básicos de programación en C así como la utilización de la consola de comandos para compilar nuestros programas cuando se tienen varios ficheros. Se ha aprendido como procesar archivos en formato `.wav` byte a byte mediante funciones de lectura de ficheros y como posteriormente guardar todos estos datos para realizar un procesado a bloques y calcular métricas temporales de la señal de audio.

Paralelamente a todos los conocimientos de C se nos ha introducido un nuevo programa `Wavesurfer` de programario libre y desarrollado para grabar, representar y analizar las señales de audio de una manera sencilla. En esta primera aproximación al programa hemos aprendido como manejarnos por los menús principales y como añadir nuestros propios `Data plots` con los datos calculados de manera externa mediante las funciones en C.

References

[1] Matplotlib. Matplotlib: Python plotting documentation. Febrero 2019. https://matplotlib.org/users/index.html

[2] Wikipedia Foundation (SF). DBFS. Febrero 2019. https://es.wikipedia.org/wiki/DBFS

[3] Wavesurfer. Wavesurfer User Manual. Febrero 2019. https://www.speech.kth.se/wavesurfer/man.html

[4] Tutorials Point. C library function fread. Febrero 2019 https://www.tutorialspoint.com/c-standard-library/c-function-fread.htm