Práctica 3 - Detección de pitch

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 se implementará un detector de pitch hecho en C++ y basandonos en la autocorrelación de la señal. Posteriormente se hará una evaluación comparando los resultados obtenidos con los de la plantilla proporcionada.**

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

I. Introducción

Los objetivos de esta práctica son: implementar un sistema detector de Pitch mediante un programa hecho en C++ y medir su calidad utilizando una base de datos de audios y aprender a programar `bash` para la automatización de las tareas de la consola de comandos, como la compilación o la ejecución de los `plots`.
La parte básica de la práctica, tal y como se ha comentado, consiste en obtener el pitch de un fitchero de voz mediante la autocorrelación de la propia señal. Este método es el más común y utilizado debido a su senzillez, rapidez de implementación y robustez frente a los posibles ruidos presentes.
Como ampliación se propone realizar un prepocesado y un post-procesado de la señal para así mejorar la calidad de nuestras mediciones. Tabién se propone implementar un detector basado en el cepstrum.

II. Descripción del diseño

A. Funciones del programa

El programa principal tiene la tarea de abrir el fichero de audio, leerlo en tramas de tamaño `FRAME_LEN` y guardar en un vector la frecuencia del pitch de cada trama. En paralelo otro programa recibe cada una de las tramas y determina si se trata de voz o silencio y en caso de corresponder a voz calcular el pitch mediante la autocorrelación o el cepstrum.

B. Esquema funcionamiento

1. Abrimos el fichero de audio.
2. Leemos un bloque de datos y aplicamos a cada muestra el efecto del enventanamiento.
3. Analizando las tramas se determina si se trata de voz o silencio y se calcula el pitch.
4. Repetimos los pasos 2, 3 y 4 hasta que se acabe el fichero `.wav`.
5. Exportamos los valores del pitch de cada archivo de audio en ficheros de texto `.f0`
6. Se ejecuta el programa de validación, que ha sido proporcionado, encargado de comparar nuestro fichero `.f0` con el `.f0ref` .
7. Se muestran por pantalla, en un mismo gráfico, los valores del pitch calculado en azul y el pitch de referencia en naranja.
8. Ampliación: Se implementa un preprocesado realizando un center-clipping de la señal.
9. Ampliación: Se aplica un post-procesado de la señal aplicando un filtro de mediana al vector de valores de pitch.
10. Ampliación: Se desarrolla el método del cepstrum como alternativa a la autocorrelación, para encontrar el pitch.

III. Aspectos implementación

A. Acceso y loop de lectura del fichero

El acceso y lectura del fichero de audio ya viene implementado mediante la función `readwav_mono()` obteniendo como resultado un vector con todas las muestras del fichero. Cada trama es después multiplicada por el efecto que introduce el enventanamiento de Hamming.

B. Cálculo pitch mediante autocorrelación

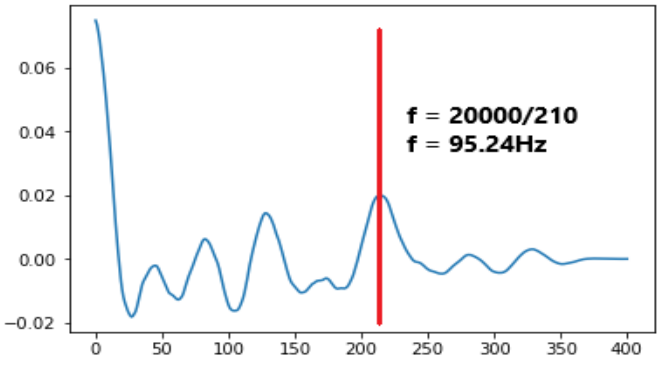

Fig. 1. Autocorrelación sección sonora

Tal y como se ha comentado en la introducción, el método de autocorrelación es uno de los más eficaces, sencillos y robustos para encontrar el pitch de un audio. La metodología es la siguiente, nuestro programa recibe una trama enventanada de 50 ms de duración, realiza la autocorrelación de la señal consigo misma y encuentra el segundo máximo. Con este índice seremos capaces después, de obtener el pitch dividiendo la frecuencia de muestreo entre el índice.

$$R_{xx} = \sum_{n=-\infty}^{\infty} x[n] * x[n+m] \quad (1)$$

Pese a que las señales de audio no son periódicas se cogen tramas lo suficientemente pequeñas como para que de manera local si que lo sean, por lo tanto al aplicar la autocorrelación, la cual lo que hace es desplazar la trama muestra a muestra, y multiplicarla por la misma trama sin desplazar, se obtienen máximos cuando las dos señales están alineadas. Por lo tanto, al coger el segundo máximo, ya que el primero se encuentra en el origen, obtenemos el índice correspondiente a la periodicidad de la trama es decir, el pitch una vez se tenga en cuenta la frecuencia de muestreo. Si se `plotean` las autocorrelaciones se podrá observar como cada vez los máximos son más pequeños, esto es debido al efecto del enventanamiento de Hamming.
Debido a que no se quiere obtener el primer máximo de la autocorrelación se pone como condición necesaria que haya habido una muestra anterior con valor negativo, ya que la autocorrelación de una señal siempre tendrá un paso por cero después de cada máximo. También se añade como condición, que el índice de la muestra encontrada sea mayor que 60, así la resolución máxima será de 333 Hz, frecuencia que se encuentra muy por encima de 220 Hz, la media en mujeres.

C. Detector voz o silencio

Así como en anteriores prácticas nos habíamos ayudado únicamente de la potencia para determinar si se trataba de voz o silencio en esta ocasión también se han utilizado valores de la autocorrelación para poder determinarlo. En concreto se ha utilizado la potencia y la autocorrelación normalizada de la segunda muestra y de la muestra en el punto donde hemos encontrado el segundo máximo. En caso de detectar silencio el valor del pitch de ese trama pasa a ser 0 automáticamente. Para conseguir un correcto funcionamiento del detector se han tenido que encontrar los valores óptimos de estos parámetros.

D. Ampliación: Prepocesado con center-clipping

Una de las técnicas de preprocesado más sencillas es el `center-clipping`, este consiste en recortar la zona media de la señal igualándola a 0. Esto se hace para intentar mejorar la visibilidad de los máximos y mínimos de la señal de audio y que por lo tanto al calcular su autocorrelación esta este más limpia de posibles ruidos. Se ha tomado como referencia para determinar los umbrales de decisión un 20% del valor absoluto del máximo de la señal.

E. Ampliación: Filtro de mediana

Un procesado muy típico a la salida del pitch estimado es utilizar un filtro de mediana. Un filtro de mediana substituye el valor de cada muestra con la mediana de las muestras vecinas. El resultado es que se eliminan muestras con valores extremos y resulta en una señal suavizada.

F. Ampliación: Detector cepstral

Este método consiste en aplicar la FFT (Fast Fourier Transform) a la trama, posteriormente calcular el logaritmo de la potencia de la señal transformada y por último volver al dominio real aplicando la IFFT (Inverse Fourier Transform). Mediante estos cálculos se obtiene un vector de coeficientes `cepstrales` del cual se tendrá que extraer la posición del segundo máximo.

$$C[n] = IFFT\{\ log_{10}(\ |FFT\{x[n]\}|\ )\ \} \quad (2)$$

Debido a que estamos aplicando la FFT es necesario que nuestra trama tenga un tamaño que sea potencia de 2 y mayor que 512, en caso de no cumplir estas condiciones se añadirían ceros a nuestra trama (`zero padding`) hasta cumplirlos. En nuestro caso debido a que se usa una ventana de 50 ms el número de muestras de cada trama es de 1000.
A la hora de escoger el máximo y teniendo el cuenta que la frecuencia del pitch se calcula a partir de la frecuencia de muestreo entre el número del índice se han definido unos valores mínimos y máximos de frecuencia, 70 Hz y 333 Hz respectivamente. Esto en términos de índices del cepstrum corresponden a las muestras mayores que 60 y menores que 285.

G. Automatización del proceso de evaluación

De manera parecida a como hicimos en la práctica 2 hemos creado un par de scripts que nos permiten ejecutar el evaluador para cada archivo. El script `exec.sh` compila el código, lanza el programa `run_pitch.py` encargado de coger cada archivo `.wav` y calcular su `.f0`, a continuación el script ejecuta el evaluador que nos ha sido proporcionado comparando los ficheros `.f0` con los `.f0ref` de referencia.

IV. Evaluación

Para evaluar el funcionamiento de nuestro programa se utiliza el script `pitch_evaluate.pl`, este nos dará información sobre:

- Voiced frames → unvoiced: Número de tramos sordos que han sido clasificados, erróneamente, como sonoras.
- Unvoiced frames → voiced: Número de tramos sonoros que han sido clasificados, erróneamente, como sordos.
- Gross voiced errors: Porcentaje de tramos sonoros con un error mayor del 20% .
- MSE of fine errors: Porcentaje de error de los tramos sonoros con un error menor del 20%

Por lo tanto para obtener unos resultados los mas parecidos posibles al pitch de referencia, nuestro objetivo consistirá en intentar reducir al mínimo estos errores.

V. Discusión de los resultados

A continuación detallaremos los resultados obtenidos con nuestro programa, analizando cualitativamente y cuantitativamente las diferentes mejoras/pruebas realizadas hasta llegar al modelo final con el mejor rendimiento.

A. *Modelo básico (auto-correlación)*

El modelo básico usa la autocorrelación para encontrar el pitch. Tiene unos resultados buenos para los audios masculinos pero malos para los femeninos, tal como se puede ver en la figura 2 y 3. En los resultados de error aparecen unos percentajes muy altos y en los plots muchos picos.

```
#audio dones finestra de 30ms
### Summary
Num. frames:    9786 = 5924 unvoiced + 3862 voiced
Unvoiced frames as voiced:      710/5924 (12%)
Voiced frames as unvoiced:      135/3862 (3.5%)
Gross voiced errors (+20%):     1124/3727 (30%)
MSE of fine errors:     4.2%
--------------------------
```

Fig. 2. Evaluador en voces masculinas

```
#audio homes finestra de 30ms
### Summary
Num. frames:    12344 = 7982 unvoiced + 4362 voiced
Unvoiced frames as voiced:      407/7982 (5.1%)
Voiced frames as unvoiced:      216/4362 (5%)
Gross voiced errors (+20%):     117/4146 (2.8%)
MSE of fine errors:     3.4%
--------------------------
```

Fig. 3. Evaluador en voces femeninas

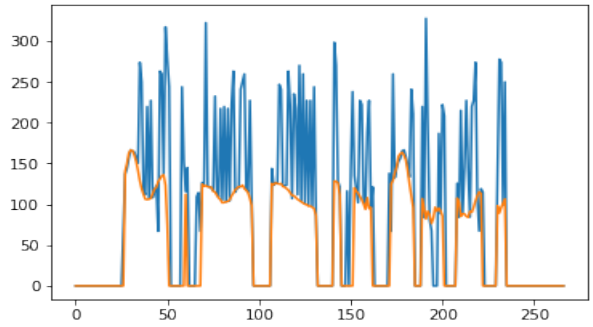

Fig. 4. Evaluador en voces masculinas

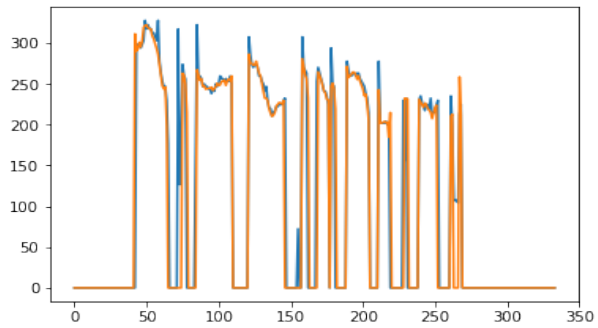

Fig. 5. Evaluador en voces femeninas

Para solventar este caso hicimos pruebas y vimos que si en lugar de utilizar trazas de 30ms usamos de 50ms tiene mucho mejor rendimiento en los audios femeninos. A partir de este punto todos los análisis són usando tramas de 50ms.

```
#finestra de 50ms
### Summary
Num. frames:    22015 = 13792 unvoiced + 8223 voiced
Unvoiced frames as voiced:      1141/13792 (8.3%)
Voiced frames as unvoiced:      497/8223 (6%)
Gross voiced errors (+20%):     685/7726 (8.9%)
MSE of fine errors:     2.6%
```

Fig. 6. Evaluador con 50ms en todas las voces

B. *Modelo con center clipping*

El modelo con center clipping es igual que el anterior pero le añadimos el preprocesado con el center-clipping. No hemos conseguido que nos de buenos resultados, el detector de voiced y unvoiced no funciona bien y hemos hecho numerosos experimentos y no hemos tenido exito.

```
### Summary
Num. frames:    22015 = 13792 unvoiced + 8223 voiced
Unvoiced frames as voiced:      629/13792 (4.6%)
Voiced frames as unvoiced:      4077/8223 (50%)
Gross voiced errors (+20%):     315/4146 (7.6%)
MSE of fine errors:     2.5%
--------------------------
```

Fig. 7. Evaluador con 50ms en todas las voces

C. *Modelo con filtro de mediana*

El filtro de mediana es un postprocesado que se realiza despues de obtener los resultados del algoritmo. Nos ha dado muy buenos resultados, sobretodo despues de hacer el cambio a trazas de 50ms.

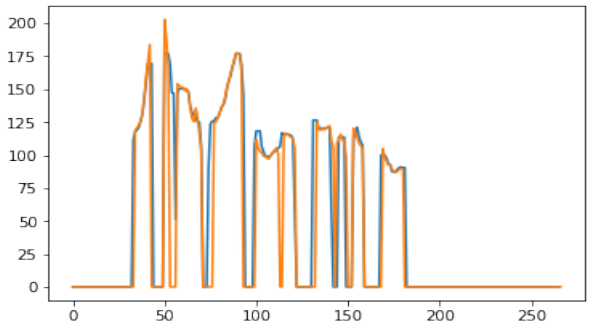

Fig. 8. Plot del audio y del estimado

```
### Summary
Num. frames:    22015 = 13792 unvoiced + 8223 voiced
Unvoiced frames as voiced:      1105/13792 (8%)
Voiced frames as unvoiced:      501/8223 (6.1%)
Gross voiced errors (+20%):     328/7722 (4.2%)
MSE of fine errors:     3.3%
--------------------------
```

Fig. 9. Evaluador con 50ms en todas las voces con filtro de mediana

D. Modelo con detector cepstral

Los resultados con el detector cepstral no han sido muy buenos en comparacion con otros métodos. Tiene un percentaje de errores grande. Nos gustaría mencionar que con el cepstrum no vimos diferencias de precision con los audios masculinos y femeninos, el cambio de longitud de las tramos tampoco modificaron su rendimiento.

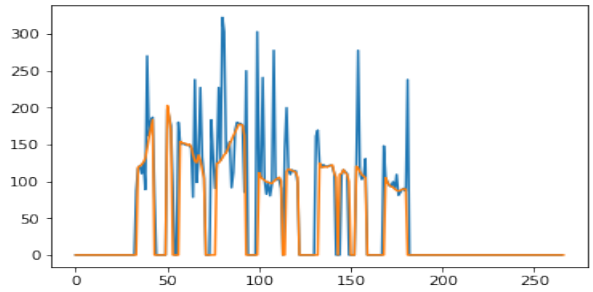

Fig. 10. Plot del audio y del estimado

```
### Summary
Num. frames:    22130 = 13906 unvoiced + 8224 voiced
Unvoiced frames as voiced:      1297/13906 (9.3%)
Voiced frames as unvoiced:      299/8224 (3.6%)
Gross voiced errors (+20%):     1159/7925 (15%)
MSE of fine errors:     4.4%
--------------------------
```

Fig. 11. Evaluador en todas las voces con ventana de 50ms

E. Modelo final

El modelo final que hemos escogido es la autocorrelación con el filtro de mediana, es el que nos ha dado mejores resultados y es el mas estable, analizando los plots de los resultados. Jugando con los parámetros hemos consegido estos numeros en la avaluación.

```
### Summary
Num. frames:    22015 = 13792 unvoiced + 8223 voiced
Unvoiced frames as voiced:      1083/13792 (7.9%)
Voiced frames as unvoiced:      467/8223 (5.7%)
Gross voiced errors (+20%):     298/7756 (3.8%)
MSE of fine errors:     3.4%
--------------------------
```

Fig. 12. Evaluador en modelo final

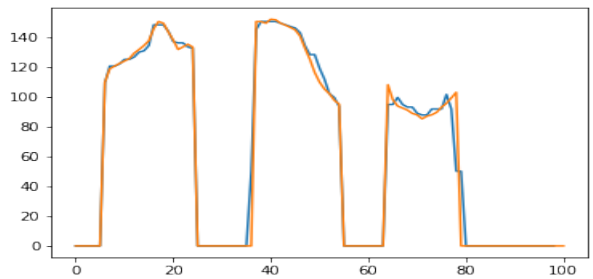

Fig. 13. Plot del audio y del estimado del modelo final

VI. Conclusiones del trabajo

Al finalizar esta práctica se ha conseguido obtener conocimientos sobre C++ y sobre como orientarlo en aplicaciones de análisis de audio. Se han podido analizar los dos metodos más comunes para el cálculo del pitch, mediante autocorrelación o cepstrums.

Como conclusiones podemos decir que la autocorrelación es un método sencillo de calcular y robusto frente al ruido, pero que por contra partida requiere vigilar a la hora de encontrar el segundo máximo de la autocorrelación, ya que esta no es un tarea trivial y puede ser difícil de determinar con exactitud.

Con el cepstrum se han obtenido peores resultados y computacionalmente hablando es un proceso más costoso al tener que operar dos veces la FFT. Debido a que nuestras muestras de voz no contenian mucho ruido no se ha hecho evidente el hecho de que este es un método mucho menos robusto con el ruido, por lo tanto se podría tener problemas a la hora de analizar otro tipo de audios más ruidosos.

Algo que ha sido de gran utilidad y ha mejorado en casi un 10% la calidad de las muestras ha sido el postprocesado de la señal de pitch con el filtro de mediana. Se ha conseguido de esta manera suavizar la discontinuidades que había.

Pese a que se proponia en la ampliación de la práctica el prepocesado basado en el `center clipping` de la señal no ha sido de ayuda ya que tras su implementación tan solo se ha conseguido empeorar los resultados.

References

[1] John Cook. Cepstrum frequency. Marzo 2019. https://johndcook.com/blog/2016/05/18/cepstrum-quefrency-and-pitch/

[2] Note detection. Center clipping. Marzo 2019. http://notedetection.weebly.com/center-clipping.html

[3] Wikipedia. Correlation function. Marzo 2019. https://wikipedia.org/Correlation-function(statistical-mechanics)