MODIFICACIÓN DE LA ESCALA TEMPORAL DE AUDIO

Juan Almaraz[1]

[1]Universidad Nacional de Tres de Febrero
juan.almaraz097@gmail.com

Resumen — *En este informe se detalla la implementación de un algoritmo en python diseñado para escalar temporalmente una señal de audio. El mismo está basado en las técnicas Overlap and Add para las componentes percusivas de la señal y Phase Vocoder para las componentes armónicas. Se implementa un filtro de mediana movil para la separación de los componentes de la señal y luego de su procesado se suman para obtener la señal escalada. Además, se analizan distintos algoritmos existentes para luego compararlos con los resultados obtenidos en el algoritmo implementado en este trabajo.*

1. INTRODUCCIÓN

El escalado temporal de señales de audio (Time Scale Modification) es una técnica muy utilizada en industrias como la producción musical o la post-producción de audio para imágenes. Cuando se busca sincronizar señales de audio con un vídeo, cuando un DJ utiliza pistas musicales con distinto tempo, es necesaria una herramienta que realice un escalado temporal y de ser posible, que no modifique el espectro de la señal. Para ello, existen múltiples algoritmos [1] que obtienen mejores o peores resultados dependiendo de las características de la señal. En general, existen algoritmos orientados a señales percusivas que se enfocan en preservar las características transitorias de los sonidos percusivos. Por otro lado, hay algoritmos que funcionan mejor para señales con contenido armónico, pero destruyen los transitorios de la señal. Es por esto que la mayoría de algoritmos de TSM utilizan una combinación de procesos aplicados a las componentes mencionadas de la señal por separado. Para ello es necesario un separador de señales armónicas-percusivas, o HPSS por sus siglas en inglés, que permita realizar este procesamiento por separado.

En este informe se propone un algoritmo que implemente un HPSS basado en un filtro de mediana móvil, un algoritmo que procese la parte percusiva de la señal basado en la técnica Overlap and Add, y un algoritmo para la parte armónica basado en la técnica de Phase Vocoder. En la figura 1 se puede ver el diagrama en bloque del algoritmo, y a continuación el análisis de cada componente.

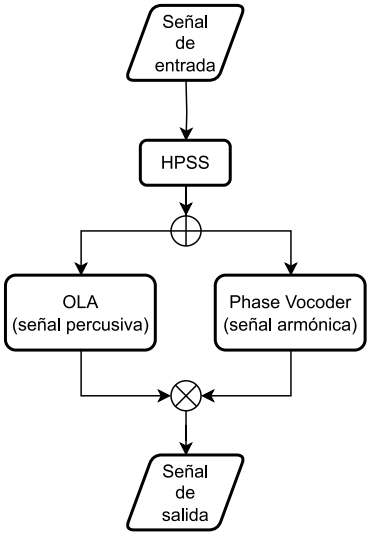

Figura 1: Diagrama de bloque del algoritmo a implementar.

2. MARCO TEÓRICO

2.1. Separador armónico-percusivo

Como se mencionó anteriormente, el separador de componentes armónicas y percusivas que se utiliza en este trabajo está basado en un filtro de mediana móvil [2] , el cual se aplica a un espectrograma en sentido horizontal (para obtener la parte armónica) y luego en sentido vertical (para obtener la parte percusiva). Para verlo en más detalle, en la figura 2 se puede ver un espectrograma que corresponde a un fragmento de una señal de audio que se utilizó para probar el algoritmo. En la imagen se ve

que las componentes armónicas se ven representadas de manera horizontal a lo largo del eje temporal, mientras que las componentes percusivas, al ser sonidos impulsivos, exitan un gran rango de frecuencias pero tienen una corta duración, por lo cual su energía se distribuye en sentido vertical.

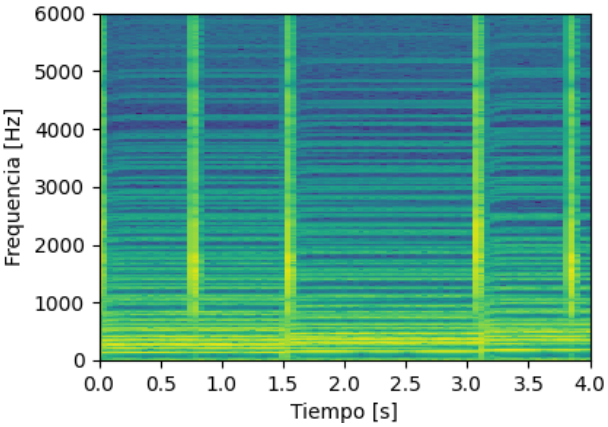

Figura 2: Espectrograma de una señal de prueba.

Entonces, el primer paso para separar la señal es obtener el espectrograma del mismo. Para ello se debe computar la STFT (short time fourier transform), la cual necesita de una señal de entrada ventaneada temporalmente. La STFT de una señal $x(r)$ ventaneada con una ventana $w(r)$ de N muestras, se calcula de la siguiente manera:

$$X(m,k) = \sum_{r=-N/2}^{N/2-1} x_m(r)w(r)e^{-2\pi jkr/N} \quad (1)$$

Donde x_m corresponde a la señal de la m-ésima ventana, $w(r)$ al tipo de ventana, N es la cantidad de muestras de la ventana, k la frecuencia. Graficando la magnitud de $|X(m,k)| = Y$, se obtiene el espectrograma de la figura 2, donde los valores de m se representan en el eje temporal, mientras que cada valor de frecuencia asociado a k está representado en el eje vertical.

Una vez obtenida la STFT, se procede a realizar un filtro de mediana móvil en sentido horizontal y vertical. En sentido horizontal se reemplaza un elemento del espectrograma por la mediana de los elementos a izquierda y derecha del mismo.

$$Y_{hor}(m,k) = med(Y(m-l,k), ..., Y(m+l,k) \quad (2)$$

Donde $2l+1$ es la cantidad de elementos sobre los cuales se toma la mediana para reemplazar el valor del elemento. Realizando el mismo proceso para valores de k, se obtienen dos espectros en los cuales la parte percusiva y armónica se ven aumentados en magnitud. En la figura 3 se pueden ver los espectros aumentados vertical y horizontalmente.

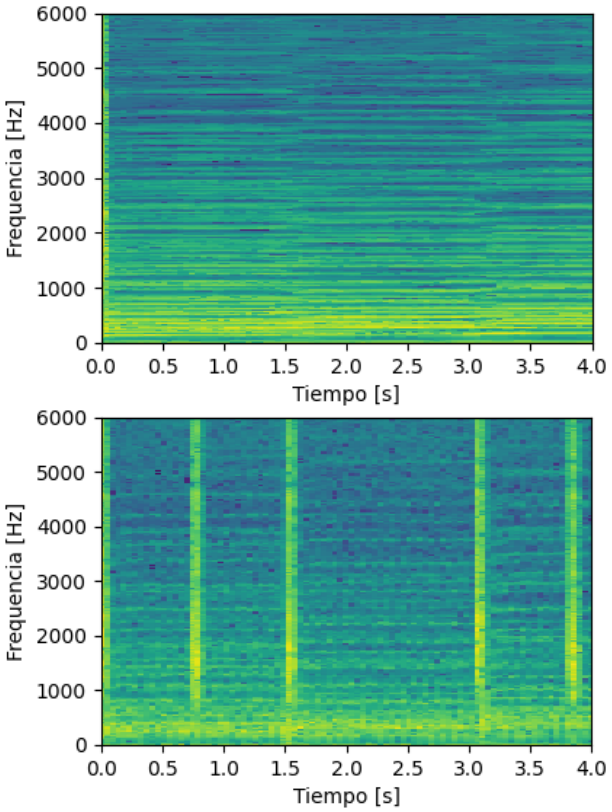

Figura 3: Espectrogramas aumentados en sentido horizontal y vertical.

Luego, se compara el valor de $Y_{hor}(m,k)$ con el de $Y_{ver}(m,k)$ para cada par de muestra-frecuencia. El objetivo es crear máscaras binarias para la componente percusiva y la armónica. Si $Y_{hor}(m,k)$ es mayor a $Y_{ver}(m,k)$, ese par (m,k) tendrá un valor de 1 en la máscara armónica, y de ser menor valdrá 0. Lo mismo vale para la máscara percusiva. Finalmente, se aplican dichas máscaras al espectrogrmama original, se computa la STFT inversa y se obtienen dos señales con las componentes percusivas y harmónicas separadas.

2.2. Overlap and Add

Para el procesamiento de la parte percusiva, existen múltiples algoritmos que cumplen con la tarea. La técnica Overlap and Add no es la única, también existen otros algoritmos como el Waveform Similarity Overlap and Add (WSOLA) [3] o el Synchronized Overlap and ADD (SOLA) [4]. Estas técnicas pueden llegar a obtener mejores resultados que el algoritmo OLA en ocasiones, pero si la señal de entrada es mayormente percusiva, el Overlap and Add es una opción simple y válida. Este algoritmo se basa en la decomposición de la señal en cuadros equidistantes de L muestras para luego, a través de un factor

de escala, separar o juntar esos cuadros y recomponer la señal escalada temporalmente. Se establece una cantidad m de cuadros, cada uno con L muestras y una distancia entre cuadros de H_a muestras.

El factor de escala se define como $\alpha = H_a/H_s$, donde H_s es la distancia entre cuadros que luego serán sumados y recomponen al señal final. En la literatura, H_a suele llamarse 'analysis Hopsize', y H_s 'synthesis Hopsize'. Para tener un control sobre el solapado de los cuadros de síntesis, se establece primero $H_s = N/2$ (valor que se suele utilizar para ventanas tipo hanning, siendo N la cantidad de muestras de la ventana), y a partir del factor de escala se obtiene H_a. La ventaja de utilizar ventanas hanning es que, para un solapado de $N/2$ muestras, la suma del total de los cuadros no produce modificaciones en la magnitud de la señal resultante con respecto a la señal de entrada. En la figura 4 se encuentra un diagrama del proceso mencionado.

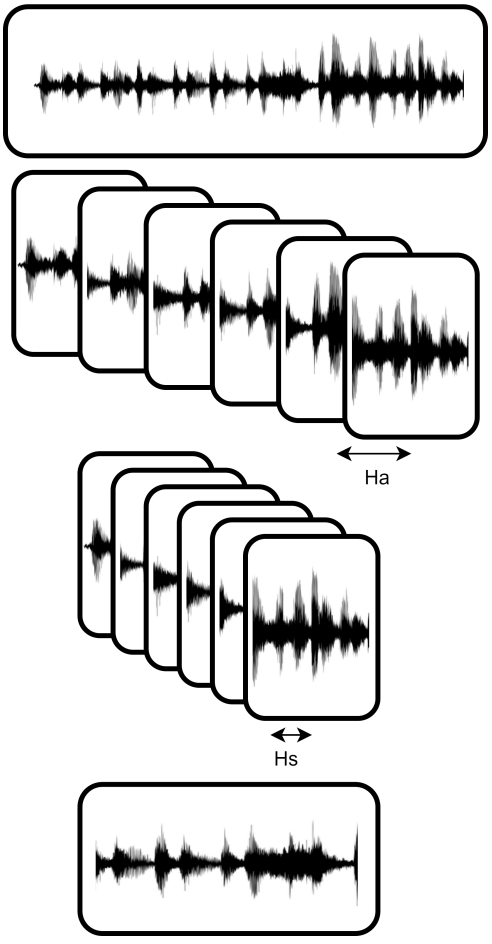

Figura 4: Método overlap and add.

El problema de esta técnica son los artefactos que aparecen al solapar los cuadros y escalar la distancia entre los mismos. Esto se produce ya que la forma de onda en los extremos de los cuadros casi nunca van a coincidir, por lo cual la forma de onda resultante presenta discontinuidades que se manifiestan como artefactos audibles. Sin embargo, este método presenta buenos resultados al utilizar señales percusivas que no presenten esctructuras periódicas.

2.3. Phase Vocoder

Para el procesamiento de las componenetes armónicas de la señal, se optó por la técnica de Phase Vocoder. Cada valor de la STFT mencionada en la ecuación 1 se puede interpretar como un componente senoidal de amplitud $|X(m,k)|$ y fase $\phi(m,k)$. Al ser un proceso en tiempo discreto, existe un número discreto de frecuencias que pueden ser representadas por la transformada, por lo tanto en la mayoría de los casos la frecuencia estimada por la STFT suele tener un desvío considerable con respecto a la frecuencia real. El algoritmo de phase vocoder utiliza la información de la fase para mejorar la precisión en la estimación de la frecuencia.

Para un cuadro de análisis m, se tiene un valor de fase $\phi_1 = \phi(m,k)$, y para el cuadro siguiente $(m+1)$ se tiene una fase $\phi_2 = \phi(m+1,k)$. El valor de fase del cuadro $m+1$ se puede predecir a partir de la fase del cuadro m, la frecuencia estimada por la STFT y la diferencia temporal entre cuadros Δt de la siguiente manera:

$$\phi_{Pred} = \phi_1 + \omega \Delta t \tag{3}$$

A su vez, se puede obtener el error en la estimación a partir del valor de la fase del cuadro $m+1$.

$$\phi_{err} = (\phi_2 - \phi_{Pred}) \tag{4}$$

Este valor es válido para diferencias de fase menores a medio período de la señal, por lo cual si la diferencia es muy grande, se deben tomar Δt más cortos.

Con estos valores, el estimado de la frecuencia instantánea se calcula de la siguiente manera:

$$IF(\omega) = \omega + \frac{(\phi_2 - (\phi_1 + \omega \Delta t)}{\Delta t} \tag{5}$$

Siendo ω la frecuencia estimada por la STFT y $\Delta t = H_a/Fs$.

Una vez obtenidas la frecuencia instantánea para un valor de $X(m,k)$, se pueden corregir los saltos de fase producto del solapamiento de los cuadros, el cual era el principal problema de la técnica Overlap-Add. Para ello se comienza un proceso iterativo llamado propagación de fase, partiendo de $X(0,k)$ con $\phi(0,k)$, obteniendo el valor de $\phi(1,k) = \phi(0,k) + IF_0(\omega)\frac{Hs}{Fs}$, luego para $X(2,k)$ su fase $\phi(2,k)$ será igual a $\phi(1,k) + IF_1(\omega)\frac{Hs}{Fs}$, etc.

3. METODOLOGÍA

El algoritmo fue escrito en python (3.11.4) utilzando las librerías numpy (1.24.3), scipy (1.10.1) y soundfile (0.12.1). Se creó un archivo llamado '$tsm.py$', el cual llama a las funciones '$hpss.py$', '$pv.py$' y '$ola.py$'. Estas funciones se encuentran en la carpeta '/funciones'. Tanto las señales de prueba como las señales escaladas se encuentran en la carpeta '/audio files'.

3.1. HPSS

Para la separación de la señal $x[r]$ en sus componentes armónicos y percusivos, se computó la STFT utilizando scipy, luego se obtienen los espectros aumentados vertical y horizontalmente con la función $median\_filter$ de la misma librería y finalmente se computa la ISTFT para obtener las señales $x_{arm}[r]$ y $x_{per}[r]$ que luego serán procesadas por el algoritmo de OLA y el de Phase Vocoder. Para esta etapa, los parámetros a modificar para probar su funcionamiento son el número de muestras N de la ventana de la STFT y las longitudes de los filtros de mediana movil vertical y horizontal l_{ver} y l_{hor}. A mayor N, se obtiene mayor resolución frecuencial, pero también menor resolución temporal. A menor N, ocurre lo contrario. Es por esto que se debe elegir un valor que logre un detalle suficiente en cada eje para obtener correctamente la separación armónica-percusiva. Luego, la longitud del filtro de mediana móvil determina la cantidad de elementos que se median para obener cada valor de $Y(m, k)$.

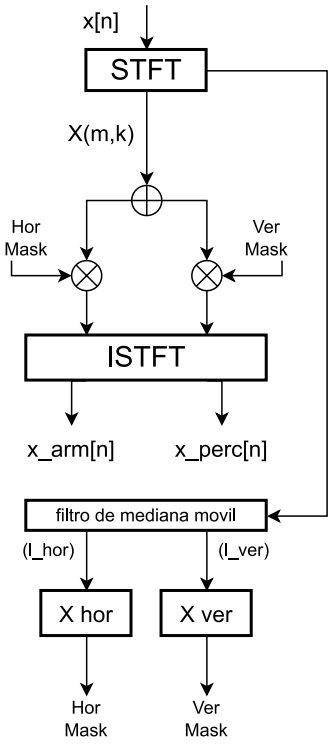

Figura 5: Diagrama de bloque del algoritmo de HPSS.

En la figura 5 se puede ver un diagrama de bloque de la función HPSS. Los argumentos de entrada son $x[r]$, fs, l_{hor}, l_{ver} y N, siendo este el tamaño de la ventana para la STFT y la ISTFT.

3.2. OLA

En cuanto al algoritmo de escalado para la señal percusiva basado en la técnica Overlap and Add, a continuación se mencionan los parámetros de entrada. El primero es el tamaño de la ventana N, a partir del cual se obtiene Hs, ya que se considera a $Hs = N/2$. El segundo parámetro es el factor de escala α, a partir del cual se obtiene $Ha = \alpha Hs$.

Una vez obtenidos los parámetros de entrada, se procede a calcular Ha y Hs a partir de los valores de α y N. Luego, simplemente se toman tramos de la señal x[n] de N muestras, con un solapado de Ha muestras, y se mapean a y[n] con un solapado de Hs muestras.

La única librería utilizada fue numpy, principalmente para utilizar una ventana tipo hanning.

3.3. Phase Vocoder

Luego, para el algoritmo de Phase Vocoder, se computa la STFT de $x_{arm}[r]$, con un tamaño de ventana N que debe ser relativamente alto para obtener mayor resolución frecuencial. Otros parámetros de entrada son el factor de escala α, y la distancia entre cuadros de análisis Hs.

El algoritmo implementado utiliza las librerías scipy para el computo de la STFT y la ISTFT posteriormente. El procedimiento es el mismo que se describe en el marco teórico. Para ello, se utiliza un ciclo for para modificar la fase de cada cuadro de análisis. También se utiliza la libreria numpy para la obtención de la fase de cada valor de X(m,k).

4. RESULTADOS Y DISCUSIONES

Primero se analizó el funcionamiento del separador de componentes. El análisis visual tanto de la señal en dominio temporal como frecuencial da una idea de qué tan bien se está separando a la señal, pero la prueba de escucha subjetiva resulta más clara para la toma de decisiones. Los parámetros de entrada se eligieron escuchando distintas señales de prueba.

Mediante prueba y error, se definió una ventana de 2048 muestras, un $l_{ver} = 50$ y un $l_{hor} = 25$. En la figura 7 se puede ver una señal de prueba junto con la componente armónica y la percusiva.

El correcto funcionamiento de esta función entonces se puede comprobar visualmente, por lo menos para la señal de prueba utilizada. De todas maneras, las pruebas de escucha subjetiva y comparación con otros algoritmos también tuvieron resultados aceptables.

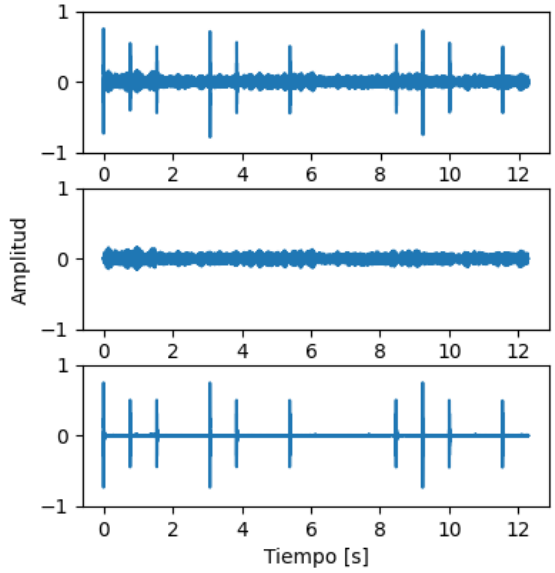

Figura 6: Separación de las componentes de la señal.

Luego, al realizar el procesamiento de la componente percusiva mediante el algoritmo de OLA se obtiene que con una ventana de 128 muestras, la señal escalada se mantiene idéntica a la señal de entrada.

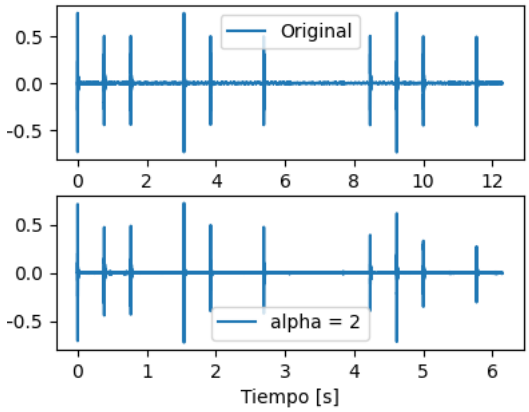

Figura 7: Escalado de la componente percusiva con α=0.5.

El problema se presenta cuando se realiza una prueba de escucha sobre otro tipo de señales. En el caso de una grabación de un solo de batería con un $\alpha = 2$, se perciben artefactos. Estos se deben a que los golpes de la batería tienen información frecuencial que no se logró separar completamente en la etapa anterior, por lo tanto se debe utilizar una ventana con un numero mayor de muestras para preservar la señal original. Al utilizar N = 512, estos artefactos desaparecen. Sin embargo, volviendo a la señal de prueba anterior se obtiene que la señal percusiva escalada presenta diferencias en la amplitud de los impulsos. Por lo tanto se debe tomar una decisión de compromiso entre los diferentes casos.

Luego, para el escalado de la señal armónica, las pruebas de escucha subjetiva resultan mas útiles que la comparación visual de la señal. De todas maneras, se comparó la STFT de la componente armónica con la STFT de la señal escalada con un factor de α=0.5. En la figura 8 se puede ver la comparación.

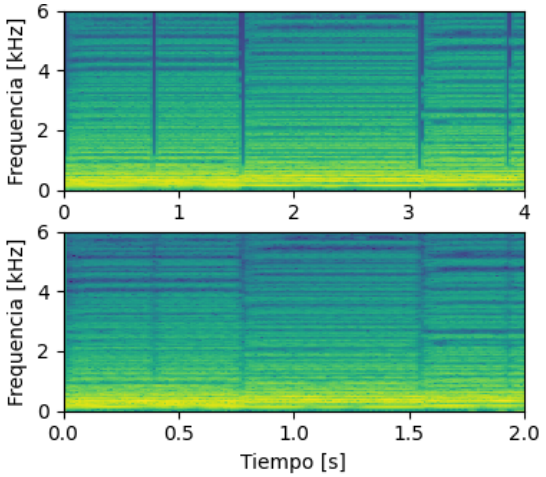

Figura 8: Escalado de la componente armónica con α=0.5.

Se puede ver que la distribución de la energía por frecuencia se mantiene similar. De todas maneras, el espectrograma aparenta ser más 'difuso', aunque es dificil saber si esto se puede apreciar auditivamente.

Finalmente, se utilizaron 5 señales distintas para comparar el funcionamiento del algoritmo con las distintas librerías existentes. Para ello se utilizan las librerías PyTSMod [5] y audiotsm [6] disponibles en github, y los algoritmos de TSM Toolbox implementados en matlab [7].

Se utilizaron 5 señales distintas con el siguiente contenido:

- Solo de Batería.[7]
- Canción Pop.[7]
- Cantante masculino.[7]
- Violin y castañuelas.
- Xilofón.

De esta manera se puede evaluar el funcionamiento del algoritmo con distintas señales. Cada señal se escaló en factores de 0.5, 1.2 y 1.8, utilizando el algoritmo desarrollado en este trabajo, el algoritmo PyTSMod [5] y el TSM de Audio Toolbox en MATLAB [7].

Con el algoritmo de este trabajo se puede apreciar que para factores de escala mayores a 1, el contenido armónico empeora su calidad. Esto puede deberse a una mala separación ya en la parte de HPSS, o puede ser que

la propagación de fase del Phase Vocoder tenga errores de diseño. Sin embargo, para un factor de 0.5, la calidad de la señal escalada es similar e incluso (subjetivamente) mejor a la de los otros algoritmos. Por lo tanto puede ser que cambiando los parámetros de entrada de cada etapa se pueda lograr un resultado parejo para factores mayores y menores a 1.

5. CONCLUSIONES

El algoritmo desarrollado cumple su función, aunque la calidad percibida resulta 'peor' en su parte armónica. Se podría realizar un test subjetivo para validar esta afirmación, pero en una primera instancia se concluye que el algoritmo requiere de ajustes para estar a la altura de las opciones disponibles en la web.

REFERENCIAS

[1] Jonathan Driedger y Meinard Müller. «A review of time-scale modification of music signals». En: *Applied Sciences* 6.2 (2016), pág. 57.

[2] Derry Fitzgerald. «Harmonic/percussive separation using median filtering». En: *Proceedings of the International Conference on Digital Audio Effects (DAFx)*. Vol. 13. 2010, págs. 1-4.

[3] Shahaf Grofit y Yizhar Lavner. «Time-scale modification of audio signals using enhanced WSOLA with management of transients». En: *IEEE transactions on audio, speech, and language processing* 16.1 (2007), págs. 106-115.

[4] Salim Roucos y Alexander Wilgus. «High quality time-scale modification for speech». En: *ICASSP'85. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 10. IEEE. 1985, págs. 493-496.

[5] *PyTSMod*. `https://github.com/KAIST-MACLab/PyTSMod`.

[6] *A real-time audio time-scale modification library*. `https://github.com/Muges/audiotsm`.

[7] Jonathan Driedger y Meinard Müller. «TSM Toolbox: MATLAB Implementations of Time-Scale Modification Algorithms.» En: *DAFx*. 2014, págs. 249-256.