TP: ”APLICACIÓN DE TÉCNICAS DE MODIFICACIÓN DE ESCALA TEMPORAL CONSERVANDO PITCH”

Sebastián Carro Morais[1], Matías Di Bernardo[2] y Matías Vereertbrugghen[3]

[1]Universidad Nacional de Tres de Febrero
sebastiancarrom @gmail.com
[2]Universidad Nacional de Tres de Febrero
matias.di.bernardo@hotmail.com
[3]Universidad Nacional de Tres de Febrero
matias.veree@gmail.com

1. INTRODUCCIÓN

En este trabajo práctico se estudian algoritmos diseñados para la modificación de la escala temporal de una señal de audio. La idea principal es conservar las características sonoras originales de la señal, es decir, que el pitch no se modifique al comprimir o expandir el audio.

Se incluye una descripción de los algoritmos a implementar, a la vez que parámetros importantes en el procesamiento y pruebas experimentales tentativas a llevar a cabo.

2. ALGORITMOS Y METODOLOGÍA

2.1. Fundamentos TSM

A modo de introducción, podemos ver que en el método general que es utilizado en los algoritmos TSM, se siguen unas líneas generales de trabajo.

La señal de entrada es una señal de audio discreta $x : \mathbb{Z} \to \mathbb{R}$, sampleada bajo una frecuencia F_s.

Primero la idea es descomponer la señal de entrada x en cortos marcos temporales denominados ‘frames de análisis’ x_m cada uno de longitud N, espaciados por el parámetro ‘Hopsize de análisis’ H_a. Esto se ve en la ecuación (1).

$$x_m(r) = \begin{cases} x\left(r + mH_{\mathrm{a}}\right), & \text{si } r \in [-N/2 : N/2 - 1] \\ 0, & \text{Otro caso.} \end{cases} \tag{1}$$

Luego en la señal de salida reubicamos estos frames en el eje temporal con respecto de un cierto espaciado ‘Hopsize de síntesis’ H_s.

El término $\alpha = H_{\mathrm{s}}/H_{\mathrm{a}}$ indica la modificación de escala temporal de la señal.

Como se suele desear tener cierta superposición de los frames recolocados, se suele fijar el parámetro H_s, y a través de esto el H_a queda definido.

Previo a la reconstrucción de la señal, para evitar discontinuidades de fase y amplitud en las fronteras de los marcos, se adaptan los frames de análisis formando los ‘frames de síntesis’ y_m. Como paso final, se sobreponen los y_m en orden de reconstruir la señal de salida ya modificada temporalmente $y(r)$. (2).

$$y(r) = \sum_{m \in \mathbb{Z}} y_m \left(r - mH_{\mathrm{s}}\right) \tag{2}$$

Los procedimientos que vemos a continuación tratan de diferentes formas la adaptación de los frames de analisis x_m hacia los frames de sintesis y_m, con sus ventajas y desventajas.

Para la validación de los algoritmos propuestos, se elegirán o se generarán distintas señales con el objetivo de comprobar la eficacia de los distintos algoritmos ante distintos casos críticos con el fin de evaluar su comportamiento en función del análisis teórico realizado.

En general las señales reales serán elegidas de distintos contextos (musicales, voces, ruidos, etc) y serán evaluadas subjetivamente de forma auditiva por los integrantes del grupo (y externos que se presten como sujetos de prueba).

Las señales generadas, al ser casos ideales (tonos puros), serán contrastadas con su versión ideal, es decir que si genero un tono puro de una frecuencia de un segundo, puedo también generar un tono puro de la misma frecuencia pero de 2 segundos, y en consecuencia evaluar el

resultado de mi algoritmo al contrastar la diferencia entre el output de mi algoritmo en relación a la versión ideal (3), y así obtener un valor de error que me cuantifique la efectividad del algoritmo.

$$\epsilon_{error} = \frac{|x_{output} - x_{ideal}|}{N} \tag{3}$$

Con estos dos criterios de validación pretendemos encontrar un criterio tanto objetivo como subjetivo para la elección de parámetros que hagan al algoritmo lo más versátil, robusto y efectivo posible.

2.2. OLA: 'Overlapp-and-add'

Este método es uno de los algoritmos TSM mas básicos, en donde se propone aplicar una ventana a los frames de análisis x_m, previo a la reconstrucción de la señal de salida $y(r)$.

La ventana ayuda a quitar las discontinuidades abruptas en las fronteras de los frames de análisis, a la vez que fluctuaciones en la amplitud cuando los frames se superponen. Se suele utilizar la ventana de *Hann*, como indica (4). Tiene una forma cosenoidal que presenta atenuación en las fronteras de los frames. En la ecuacion (5) figura una propiedad útil, que se utiliza a la hora de ejecutar los frames de sintesis y_m.

$$w(r) = \begin{cases} 0{,}5\left(1 - \cos\left(\frac{2\pi(r+N/2)}{N-1}\right)\right), \text{ si } |r| \leq N/2 \\ 0, \text{ otro caso.} \end{cases} \tag{4}$$

$$\sum_{n\in\mathbb{Z}} w\left(r - n\frac{N}{2}\right) = 1 \tag{5}$$

Como procedimiento, primero se deben computar los frames de análisis x_m con la ecuación (1). Luego obtener los frames de síntesis y_m con la ecuación (6). Notar que en el nominador estamos ejecutando el ventaneo al multiplicar punto por punto con la ventana. Y el denominador actúa a modo de normalizar la amplitud del frame para prevenir fluctuaciones en la salida (amplitud y fase). Finalmente haciendo uso de la expresion en (2) obtenemos $y(r)$.

$$y_m(r) = \frac{w(r)x_m(r)}{\sum_{n\in\mathbb{Z}} w\left(r - nH_{\mathrm{S}}\right)} \tag{6}$$

OLA es un algoritmo de TSM que trabaja sobre los frames de análisis puramente en el dominio temporal. Si bien en general logra preservar el timbre de la señal de entrada con buena calidad, genera ciertos artefactos indeseados.

Reporta problemas al trabajar con señales de audio con componentes armónicos, por lo que **no es apropiado para trabajar con música o cualquier señal con alto contenido tonal**.

Esto se debe a que este procedimiento no es capaz de preservar estructuras periódicas locales que están presentes en la señal de entrada, ya que al recolocar los frames de análisis, las estructuras periódicas de x ya no se alinean en los frames de síntesis superpuestos. En la señal de salida y, los patrones periódicos son distorsionados, a lo que se denomina *'phase jump artifacts'*. Y estas periodicidades locales en las formas de onda de las señales de audio corresponden a sonidos armónicos.

Por otro lado, este algoritmo **entrega resultados de buena calidad para señales puramente percusivas**, como es en el caso de grabaciones de batería por ejemplo. Esto es así debido a que las señales con contenido percusivo carecen de estructuras locales periódicas, por lo que, los artefactos de saltos de fase no son notables en la señal de salida.

En este caso, *Driedger y Muller* [1] enuncian que es importante utilizar un largo de frame N muy pequeño (aproximadamente 10 ms) para reducir el efecto indeseado de *'transient doubling'*. Esta duplicación de transientes sucede ya que al recolocar los frames y copiarlos en la señal de salida, si la transiente aparece en dos frames, será duplicada en la salida y se escuchará dos veces en una rápida sucesión, lo que es indeseado y antinatural para esta aplicación.

Como ya sabemos que este algoritmo funciona bien con señal percusivas, y será utilizado de esta forma en conjunto con la separación armónico-percusiva (HPS), por lo que en principio no haría falta hacer pruebas con señales armónicas. Se pueden llevar a cabo algunas instancias con señales de este tipo solo a modo de corroborar las hipótesis teóricas y verificar los artefactos de saltos de fase.

Luego podemos comenzar a hacer pruebas con distintos audios percusivos, de distintas índoles y timbres. Por ejemplo: baterías acústicas, platillos, tambores, shakers, mallets, baterias electrónicas, entre otros.

El parámetro más importante a tener en cuenta es el tamaño de los frames. Así que se deberían efectuar pruebas cambiando el largo de estos marcos temporales hasta obtener el mejor resultado, en términos de percepción auditiva (considerando las recomendaciones de utilizar aproximadamente 10 ms).

Otro método de verificación será utilizando una señal ideal compuesta por una delta en algún tiempo determinado y comparar el resultado del algoritmo contra el resultado ideal de la señal con el pulso desplazado el tiempo especificado. Si bien sabemos que en la práctica ninguna señal tendrá estas características tan extremas consideramos que tener una medida objetiva nos servirá para contrastar con la verificación experimental subjetiva descrita anteriormente.

2.3. PV-TSM: 'Phase-Vocoder'

El TSM phase vocoder es un algoritmo que realiza modificaciones temporales en el dominio de la frecuencia. La idea de este algoritmo es buscar las frecuencias instantáneas de x_m y transicionar a las frecuencias del siguiente frame contemplando un cambio de fase, llamado propagación de fase, que garantice la continuidad de la señal y así evitar artefactos y discontinuidades.

El primer paso en el análisis frecuencial es realizar la transformada STFT a la señal de interés. En este paso es importante definir los parámetros de la STFT. En [1] se recomienda utilizar valores de ventana de 100 ms de duración para tener buena resolución frecuencial. En principio se utilizará ese valor de ventana como referencia pero se realizarán pruebas con ventanas de menor y mayor tamaño. En principio se utilizará una ventana Hamming por su balance entre ancho de lóbulo principal y piso de ruido, perp se harán pruebas con otras ventanas como Blackman o Hann.

Sabemos que al aumentar la ventana se tendrá una mayor resolución frecuencial que mejora el rendimiento del algoritmo, pero a coste de afectar la resolución temporal, por lo cual en señales con cambios rápidos de frecuencia este cambio puede significar el perder información de los cambios tonales. En [1] se recomienda un valor de hopsize chico para la STFT, pero no se establece un valor específico así que se probarán diferentes valores como 512, 240, 120, 60 y se determinará en las pruebas cuál es el más efectivo.En general se tomarán los valores de referencia de [2].

El segundo paso es calcular la frecuencia instantánea de x_m, esto se realiza calculando la predicción de la fase del siguiente frame a partir del tiempo entre frames (7). Y luego comparar esta predicción con el valor real del frame siguiente x_{m+1} (8). Para luego poder obtener un valor de frecuencia instantánea que contemple este error de fase entre frames (9).

$$\varphi^{pred} = \varphi_1 + \omega\Delta t \tag{7}$$

$$\varphi^{Err} = \psi(\varphi_2 - \varphi^{Pred}) \tag{8}$$

$$IF(\omega) = \omega + \frac{\varphi^{Err}}{\Delta t} \tag{9}$$

Hay que aclarar que este análisis asume que la frecuencia instantánea y la frecuencia que determina la STFT en ese frame no difieren por más de medio periodo. Por este motivo es que la resolución frecuencial es importante al elegir el ventaneo de la señal, y es un factor a tener en cuenta si se analizan señales con variaciones tonales muy rápidas (menores al largo de ventana elegido).

Este procedimiento se realiza para cada bin tiempo-frecuencia de la STFT para obtener una versión con mayor precisión (10).

$$F^{IF}_{coef}(m,k) = IF(\omega) = \omega + \frac{\psi(\varphi_2 - (\varphi_1 + \omega\Delta t))}{\Delta t} \tag{10}$$

El último paso es comparar dos frames consecutivos y modificarlos para que la transición entre estos sea continua y evitar saltos de fase (11). Para esto se necesita calcular la modificación de fase adecuada, la cual se obtiene de forma iterativa en un proceso llamado propagación de fase.

$$X^{Mod}(m,k) = |X(m,k)|exp(2\pi\varphi^{Mod}(m,k)) \tag{11}$$

La fase modificada se calcula teniendo la STFT con las frecuencias instantáneas calculadas en el paso anterior y prediciendo el desplazamiento ideal de fase para el tiempo determinado por el hop size de análisis (12). Este proceso se realiza para cada par de frames consecutivos garantizando así la continuidad de fase de toda la señal de síntesis.

$$\varphi^{Mod}(m+1,k) = \varphi^{Mod}(m,k) + F^{IF}_{coef}(m,k)\frac{H_s}{F_s} \tag{12}$$

Este algoritmo funciona bien para señales con contenido armónico pero en el proceso de la propagación de fase se ve afectada la coherencia vertical de la fase. Esto resulta en la pérdida de transientes en la señal. Otro efecto negativo es que la propagación de fase si bien logra mantener coherencia no logra un resultado final tan natural, este efecto en la coloración de la señal se conoce como *phasiness*. Por el contrario, **este algoritmo no funciona bien con sonidos percusivos**, en la aplicación el algoritmo termina eliminando o atenuando las transientes.

Para evitar el fenómeno de *phasiness* mediante TSM PV se utiliza una técnica llamada 'phase locking', la cual en vez de modificar la fase de forma independiente para cada bin frecuencial se hace de manera conjunta teniendo en cuenta los picos o frecuencias fundamentales en la señal [3].

Lo más importante a destacar al implementar esta mejora es el algoritmo de detección de picos [4], ya que hay distintos caminos para resolver esta tarea con sus ventajas y desventajas, pero lo más relevante es determinar las características de la señal a analizar. Si la señal contiene principalmente componentes armónicos, el espectro va a tener una forma lobular donde el algoritmo de detección de picos tendrá menos errores, y el proceso de phase locking será más sencillo. Por ende, la implementación de esta mejora depende significativamente del desarrollo de la sección de separación de la señal desarrollada en la

sección 2.4. Según la eficacia de la separación de señal, proponemos utilizar el algoritmo de peak detection implementado en Scipy y comparar los resultados del TSM PV con y sin phase locking.

Para la etapa de validación se utilizarán diferentes señales para testear los puntos mas críticos del algoritmo. Se mencionó que las señales con una variación armónica rápida puede generar una estimación errónea de la frecuencia instantánea, y, por ende, no reconstruirse de manera adecuada en el proceso de modificación de la STFT en mi frame de síntesis. Para evaluar la severidad de estos problemas se evaluarán dos señales específicas. Una señal de audio real de una pieza músical de piano donde se realicen cambios de notas muy rápido, y otra será una señal generada por tonos puros alternados muy rápidamente. Consideramos que estos casos extremos ponen a prueba la robustez del modelo antes los casos más críticos. El caso real será analizado de forma auditiva y el caso ideal será analizado por medio de la métrica de similitud.

En el caso de la implementación del PV con phase locking sé verificará con señales reales que sean tonales pero con ruido agregado para ver cómo responde el algoritmo de peak detection utilizado ante una señal que tenga una respuesta frecuencial más caótica.

2.4. HPS: 'Harmonic-Percussive Separation'

Como ya fue detallado anteriormente, el algoritmo de OLA alcanza los mejores resultados para señales percusivas y el el PV-TSM en armónicas. De aquí surge la utilidad de separar el contenido estas estructuras de una señal, para poder procesarlos de forma independiente y obtener las mejores cualidades de cada algoritmo de TSM. Este proceso de separación se conoce como Harmonic-Percussive Separation (HPS) y tambien se utiliza para realizar remezclas ya que permite cambiar el balance de estas dos componentes. Para este trabajo nos centramos en la implementación HPS para obtener buenos resultados en una TSM.

Será implementado el algoritmo simple y efectivo propuesto por Fitzgerald [5], cuyo funcionamiento se basa en filtros de media móvil. El primer paso para aplicar este método es calcular la STFT X , luego a partir de estas transformadas se obtiene el espectrograma como la magnitud de X. A la hora de definir esta transformación deberá ser considerado el largo de muestras el cual se busca ajustar un valor óptimo para obtener un buen balance entre resolución entre frecuencia y tiempo.

En principio serán implementados los valores de los ejemplos dados en [5], los cuales fueron ejecutados con una señal con una frecuencia de muestreo de 44.1 kHz, un largo de FFT de 4096 muestras y un hopsize de 1024.

Si observamos espectrogramas de señales musicales podemos encontrar que la parte percusiva y armónica se distribuyen con estructuras definidas y con dirección contraria. Como las transientes ocurren durante un corto periodo de tiempo, al visualizar el espectrograma se distribuyen de forma vertical y su contenido armónico es homogéneo. Teóricamente esto es evidente dado que la respuesta en frecuencia de un impulso es una constante. Por el contrario, el contenido armónico se distribuye de forma horizontal, la cual se compone por una fundamental y sus armónicos que se repiten al aumentar la frecuencia.

El siguiente paso para separar estas estructuras es procesar el espectrograma con un filtro de media móvil en dos configuraciones diferentes (13). La primera será con desplazamiento horizontal a través del tiempo, del cual se obtienen las componentes armónicas dado que el contenido percusivo tendrá mayor frecuencia. La segunda configuración se aplica verticalmente a través de la frecuencia y ocurre lo contrario. Para este proceso se programará una función que calcule el filtro de media móvil, en el cual inicialmente se utilizará un largo de muestras de entre 15 a 30, tal como es propuesto en [5].

$$\begin{aligned}\tilde{Y}_{\mathrm{h}}(m,k) &= \operatorname{prom}\left(Y\left(m-\ell_{\mathrm{h}},k\right),\ldots,Y\left(m+\ell_{\mathrm{h}},k\right)\right)\\ \tilde{Y}_{\mathrm{p}}(m,k) &= \operatorname{prom}\left(Y\left(m,k-\ell_{\mathrm{p}}\right),\ldots,Y\left(m,k+\ell_{\mathrm{p}}\right)\right)\end{aligned} \tag{13}$$

Una vez obtenidos estos dos espectrogramas filtrados se utilizan para definir un par de 'máscaras binarias', las cuales seleccionan qué partes de la señal de entrada son percusivas y cuales armónicas. (14). Esto se consigue comparando la magnitud de los dos espectrogramas ya filtrados en cada segmento y frecuencia. En base a esta comparación se le asigna un valor de 1 a la máscara que tenga la magnitud predominante. Notar que la suma de las dos máscaras debería formar un pasa todo.

$$\begin{aligned}\mathcal{M}_{\mathrm{h}}(m,k) &= \begin{cases}1, & \text{si } \tilde{Y}_{\mathrm{h}}(m,k) > \tilde{Y}_{\mathrm{p}}(m,k),\\ 0, & \text{otro caso,}\end{cases}\\ \mathcal{M}_{\mathrm{p}}(m,k) &= \begin{cases}1, & \text{si } \tilde{Y}_{\mathrm{p}}(m,k) \geq \tilde{Y}_{\mathrm{h}}(m,k),\\ 0, & \text{otro caso.}\end{cases}\end{aligned} \tag{14}$$

Finalmente con las máscaras se separa el contenido haciendo el producto con la señal ya previamente trasformada, y se puede obtener las señales en función del tiempo antitransformando los resultados (15).

$$\begin{aligned}X_{\mathrm{h}}(m,k) &= X(m,k)\mathcal{M}_{\mathrm{h}}(m,k)\\ X_{\mathrm{p}}(m,k) &= X(m,k)\mathcal{M}_{\mathrm{p}}(m,k).\end{aligned} \tag{15}$$

Tal como fue propuesto anteriormente se buscará evaluar el algoritmo con señales ideales generadas. El HPS en particular podrá ser evaluado independientemente procesando una señal de entrada que sea la suma de otras dos

señales, una percusiva y otra armónica, y luego evaluando mediante correlación la capacidad del método en volver a obtener estas señales ideales de entradas separadas. Luego se buscará evaluar el sistema para aplicar distintos algoritmos de TSM en cada señal separada, haciendo el máximo aprovechamiento de cada uno de estos. Una vez se verifique el correcto funcionamiento para señales simples, se pondrá a prueba el sistema con señales musicales.

2.5. Estructura del código (tentativa)

- Carpeta TSM:

  En esta carpeta va estar la funcionalidad principal de los algoritmos a implementar. (OLA.py - PV.py - HPS.py).

- Carpeta Test:

  En esta carpeta va a estar el código que contenga la funcionalidad de testeo de los distintos algoritmos. (plotting.py - signals.py - test_objetive.py - test_subjetive.py). Entre otros.

- Carpeta Utils:

  En esta carpeta va a estar el código auxiliar que sea redundante y necesario en los distintos módulos. (read_wav.py - peak_detection.py - windows.py). Entre otros.

- Directorio Principal:

  En este directorio va a estar el script principal que ofrezca todas las funcionalidades de forma compacta y sencilla. Se incluirá un archivo de *Jupyter Notebook* donde se mostrará de forma escrita y gráfica los resultados más relevantes del proyecto. (main_tsm.py - resultados_y_conclusiones_tsm.ipynb)

REFERENCIAS

[1] Jonathan Driedger y Meinard Muller. «A review of time-scale modification of music signals». En: (2016).

[2] Jonathan Driedger y Meinard Muller. «TSM Toolbox: Matlab implementations of time-scale modification algorithms». En: (2014).

[3] M Laroche J.; Dolson. «Improved phase vocoder time-scale modification of audio». En: (1999).

[4] Hamid Satar-Boroujeni y Bahram Shafai. «Peak Tracking and Partial Formation of Music Signals». En: (2015).

[5] Derry FitzGerald. «Harmonic/Percusive Separation using Median Filtering». En: (2010).