UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

"Desarrollo de una aplicación para la detección y codificación MIDI de melodías por medio de análisis tiempo-frecuencia"

TRABAJO FINAL DE GRADO

Autor/a: Rodríguez Martínez, Leonardo

Tutor/a: Ferrer Contreras, Miguel

GANDIA, 2020

Resumen

El desarrollo de herramientas para la industria de la música tiene cada vez mayor crecimiento, tanto a nivel de hardware como de software. Así pues, el procesado de efectos de audio por ordenador y la evolución del software de creación musical son ejemplos que emplean el formato digital para el desarrollo de nuevas aplicaciones e ideas en el ámbito de la producción musical.

Con el fin de ofrecer una solución a la transcripción de ideas musicales a un entorno de producción a aquellos usuarios no habituados con la teoría musical, se propone una herramienta en el software MATLAB R2019a para el estudio de eventos musicales donde, a partir de un análisis tiempo-frecuencia, se intenta obtener la información relevante para poder codificar una melodía monofónica principal en formato MIDI y poder almacenarla, manipularla o sintetizarla posteriormente en cualquier sintetizador que soporte dicho formato.

Por otra parte, tras el análisis tiempo-frecuencia, se realizará una síntesis de dicha información para obtener una representación esencial de la señal original. Al mismo tiempo, se añadirán herramientas para la implementación de efectos de cambio de *pitch* y tiempo en la señal analizada.

PALABRAS CLAVE: Análisis tiempo-frecuencia, Procesado por evento musicales, Tratamiento digital de audio, MATLAB, MIDI, Extracción de melodías, Síntesis aditiva, FL Studio.

Abstract

The tool development for music industry is coming under increasing growth, both in hardware and software. Thus, computer audio effects processing and the evolution of music creation software are examples that use the digital format for the development of new applications and ideas in the field of music production.

In order to offer a solution for the ideas transcription in a production environment for those users who are not habitual with music theory, a tool in MATLAB R2019a software is proposed for the study of musical events where, based on an time-frequency analysis, the relevant information to encode a main monophonic melody in MIDI format is attempted to obtain and able to store, manipulate or synthesize it later on any synthesizer that supports the format.

On the other hand, after the time-frequency analysis, a synthesis of this information is performed to obtain an essential representation of the original signal. At the same time, tools for the implementation of *pitch* and time-changing effects has been implemented on the analysed signal.

KEYWORDS: Time-frequency analysis, Musical events processing, Digital audio processing, MATLAB, MIDI, Melody extraction, Additive synthesis, FL Studio.

Índice

Índice de ilustraciones

Índice de ecuaciones

Índice de tablas

1. Introducción

1.1 Presentación

Durante los últimos años, la evolución de la tecnología relacionada con el tratamiento de audio o producción musical, se ha basado mayormente en el desarrollo del *software* en los equipos. Esto ha influido en la evolución de los procesadores de voz (como el conocido *Autotune* [1]), módulos de efectos digitales, instrumentos virtuales o *samplers,* que han ido incorporando nuevos algoritmos de análisis, procesado y síntesis de audio.

A su vez, las herramientas que incluyen procesado de eventos musicales, están cada vez más presentes hoy en día. Desde aplicaciones móviles para identificar canciones sonando en el entorno del usuario, hasta software de síntesis *online* y efectos de voz como cambio de tiempo o de *pitch* (utilizados comúnmente en filtros de redes sociales) [2] [3].

Todo esto ha derivado en un avance tecnológico en el conjunto de las herramientas que son utilizadas durante el proceso de creación musical con el propósito de facilitar la creación, la transmisión, la escucha de música y demás procesos intermedios.

1.2 Motivación

Este proyecto surge con el propósito de que, usuarios con bajo conocimiento del lenguaje musical, puedan interpretar sus ideas melódicas mediante un instrumento (o incluso mediante silbidos) y así transmitirlas al DAW (Digital Audio Workstation) de una forma sencilla para manipularlas y continuar con el proceso de producción musical.

Partiendo de esta motivación y siguiendo la tendencia del software relacionado con la música mencionado anteriormente, se propone una herramienta que facilite la transmisión de ideas desde la mente del intérprete al DAW a través de un archivo en formato MIDI. A su vez, dicha herramienta permitirá implementar efectos de cambio de tiempo y de tono de la señal original o sintetizar una nueva señal seleccionando sólo la información frecuencial dominante que el usuario considere.

1.3 Objetivos

El objetivo principal del proyecto es la implementación de una herramienta funcional en MATLAB de análisis de eventos musicales para su posterior codificación, síntesis y procesado. No obstante, para valorar el cumplimiento del objetivo principal, se han establecido los siguientes objetivos secundarios a llevar a cabo en el proyecto:

- Estudiar el proceso llevado a cabo en un análisis tiempo-frecuencia [4].
- Estudiar la representación musical mediante parciales.
- Estudiar tipos de síntesis simples para resintetizar sólo la información relevante.
- Estudiar posibles premisas sobre la estructura de melodías para su correcta identificación.
- Realizar un análisis tiempo-frecuencia correcto.
- Registrar la información importante de cada evento musical en un segmento temporal del audio.
- Obtener el número de descriptores relevantes para la codificación MIDI.
- Diseñar una interfaz de usuario de la aplicación simple mediante *Matlab App Designer*.
- Optimizar el tiempo de ejecución de algoritmos para un procesado rápido.
- Verificar los resultados obtenidos para extraer conclusiones tanto objetivas como subjetivas de las fases de síntesis, implementación de efectos y extracción de melodía que componen el proyecto.
- Proponer posibles mejoras de la herramienta.

1.4 Metodología

Por lo que respecta al procedimiento llevado a cabo para la realización del trabajo, se ha dividido en tres fases desarrolladas en el capítulo 2.

En primer lugar, se establece una fase de análisis tiempo-frecuencia del audio de entrada. En segundo lugar, se procede a la fase de síntesis de la información correspondiente a los parciales codificados obtenida del análisis y a la implementación de efectos sobre el tiempo o tono de la señal original. Para finalizar, se concluye con una fase de extracción de la melodía principal a partir de la codificación empleada en el análisis para su posterior guardado como un archivo MIDI.

Para llevar a cabo el conjunto de fases, se ha obtenido información bibliográfica previa (tanto de temarios de asignaturas [5] [6] como artículos académicos y páginas web) sobre conceptos clave relacionados con el tratamiento digital de señales así como el conjunto de técnicas que forman el análisis tiempo-frecuencia; la codificación MIDI empleada para representar la información de parciales como notas y la síntesis aditiva como posible solución sencilla a la síntesis de parciales; y la extracción de melodías sobre una polifonía.

A continuación, se ha procedido al establecimiento de los objetivos del proyecto y a la implementación del código del programa.

Para finalizar, tras realizar pruebas e ir avanzando tanto en las fases como en los objetivos del proyecto, se ha optado por el diseño de un entorno gráfico y la integración de las funciones en programas principales.

2. Desarrollo

2.1 Antecedentes y conceptos clave

Para poder hablar sobre el proceso de análisis, codificación y síntesis de eventos musicales es necesario partir de una serie de conceptos clave que son esenciales en el proyecto y en todo el proceso que se ha llevado a cabo.

2.1.1 Digitalización de señal

Partiendo de una señal audible analógica, donde se pueden obtener infinitos valores temporales y de amplitud, se consigue la señal digital gracias al conversor analógico-digital (ADC por sus siglas en inglés *Analog-Digital Converter*) [7], que transforma esta señal, de variación continua, en una señal digital discreta con un número finito de valores de amplitud que permiten tanto un mejor procesado como una mejora en la reconstrucción y la detección de errores.
Para ello, el ADC consta de tres módulos principales que definen las tres fases de la conversión: muestreador, cuantificador y codificador.

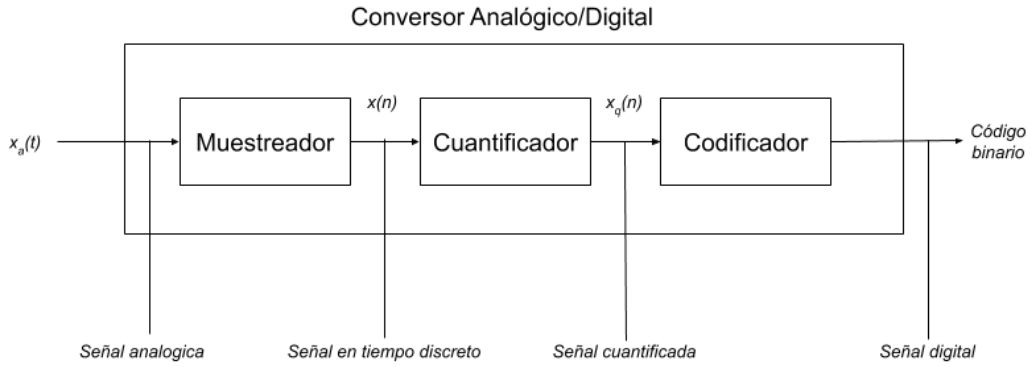


Ilustración 1: Diagrama del Conversor Analógico-Digital [7]

Cada fase consta de las características siguientes:

Muestreo
Se define el muestreo como la toma de valores temporales (equidistantes en caso de muestreo periódico) en una señal. Partiendo de una señal continua (Ilustración 2) matemáticamente, se consigue muestrear usando un generador de impulsos repetidos cada un cierto período T_s *o lo que es lo mismo,* con una frecuencia de repetición determinada f_s (denominada frecuencia de muestreo). Ambos con una relación inversamente proporcional definida por la siguiente fórmula:

$$f_s = \frac{1}{T_s}$$

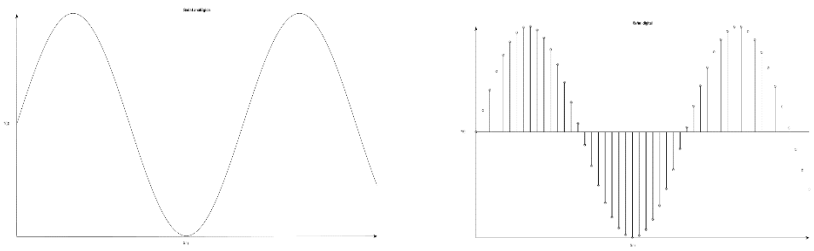


Ilustración 2: Señal analógica

Ilustración 3: Señal muestreada

Las frecuencias de muestreo más comunes en audio son 44,1 kHz (utilizado en CD), 48 kHz (utilizado en grabación y en los formatos audiovisuales digitales como vídeo digital, TDT, etc.) y 96 kHz (utilizado en audio profesional).

Cuantificación

Una vez muestreada, la señal requiere la toma de valores de amplitud en esos instantes seleccionados. Para ello, se establece un conjunto discreto de valores que la señal puede tomar en cada instante de muestreo y se aproxima cada valor de la señal original al valor más cercano de ese conjunto. Esta aproximación, por tanto, definirá el error que se produce en el proceso de cuantificación (llamado error de cuantificación), que puede aumentar o disminuir dependiendo del tipo de señal y de la cantidad de valores disponibles.
Esta totalidad de valores de cuantificación, viene a su vez determinada por 2^n, siendo *n* el número de bits usados para representar los valores en la cuantificación.
Es dicho parámetro *n* al que se le conoce como profundidad de bit o resolución [8], y en los ADC, permite establecer la calidad de la conversión, siendo 16 o 24 bits (con 65536 y 16777216 niveles respectivamente) los valores más comunes.

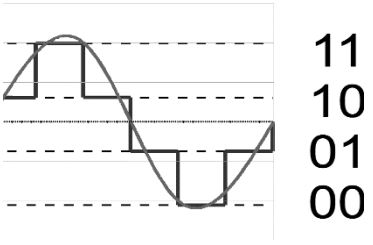


Ilustración 4: Señal cuantificada con 3 bits [9] (izquierda)

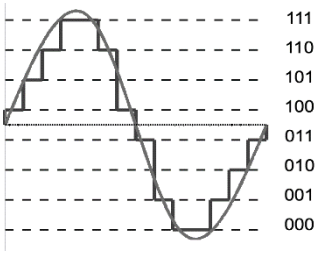


Ilustración 5: Señal cuantificada con 2 bits [9] (derecha)

Codificación

Finalmente, la señal cuantificada pasa por el codificador que, con el fin de convertirla en señal digital, genera cierta sucesión de valores binarios dependiendo del tipo de codificación empleado. Una de las más utilizadas en audio digital es la codificación sin pérdidas PCM, que consiste en una representación directa de los dígitos binarios del valor de cada muestra.

Almacenamiento

Una vez la señal está codificada digitalmente, se procede a almacenarla en un formato de audio. Para ello se utilizan variedad de formatos que se pueden clasificar en autodescriptivos (añaden una cabecera con información del archivo) o sin cabecera (tipo RAW). A su vez, se distinguen aquellos que hacen uso de codificación con o sin pérdidas. El formato utilizado dependerá de la necesidad o aplicación del audio correspondiente.
Se podría destacar el formato de audio WAV (*WAVE form audio file format*). Este es empleado en audio sin compresión, que consiste en la representación directa de los dígitos binarios de cada valor de muestra mediante el uso de la codificación LPCM [10].

MIDI

El formato de codificación de audio puede variar dependiendo de las aplicaciones y el estándar que se quiera llevar a cabo. En este proyecto es relevante hablar sobre el estándar MIDI, ya que define el formato utilizado para la variable de salida más importante de la herramienta.
El estándar MIDI proviene de *Musical Instrument Digital Interface* y se define como un protocolo de comunicación entre instrumentos por medio de mensajes sobre eventos musicales. Su propio formato de almacenamiento de eventos se denomina Formato MIDI Estándar (o SMF) y se puede identificar por su extensión ".mid" [11].
El formato de cada mensaje MIDI consiste en un primer byte de estado seguido por un máximo de dos bytes de datos. Los más relevantes para este proyecto son los siguientes:

Byte de estado		Byte de datos	Byte de datos
Note Off	Canal	Nota	*Velocity*
1000	0000-1111	00000000-01111111	00000000-01111111
Note On			
1001			

Tabla 1: Formato de un mensaje MIDI básico

2.1.2 Transformación al dominio frecuencial

Partiendo de una señal en el dominio temporal, si se requiere analizar sus componentes frecuenciales e identificar aquellas frecuencias que puedan representar posibles notas, es necesaria una transformación al dominio frecuencial. Para ello, se utiliza la Transformada de Fourier [12], que define que toda onda compleja en el tiempo está compuesta por una suma de ondas (o tonos) simples matemáticamente representadas por senos y cosenos a diferentes frecuencias. Estos tonos simples que componen un sonido complejo, son conocidos como parciales.
Dicha transformada, se ve expresada en la siguiente fórmula, siendo $f(t)$ la función que representa la señal en tiempo continuo y $G(f)$ su correspondiente transformada:

$$G(f) = \int_{-\infty}^{+\infty} f(t)\, e^{-i2\pi f t} dt$$

Ecuación 1: Transformada de Fourier

donde:

f corresponde a la frecuencia.
t corresponde a la variable tiempo.
$f(t)$ corresponde a la función en dominio temporal continuo.

En este proyecto se hará uso de la aplicación particular de la Transformada de Fourier Discreta (DFT), ya que se parte de una señal de audio digital previamente muestreada con valores temporales discretos y duración finita. Se define la DFT como la transformada de una secuencia $x[n]$ discreta de N muestras:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi}{N}kn} \qquad k = 0, \ldots, N-1$$

Ecuación 2: Transformada de Fourier Discreta

donde:

k corresponde a la variable frecuencial discreta de la transformada (*bins*).
N corresponde al total de muestras de la señal o trama.
$x[n]$ corresponde a la función en dominio temporal discreto.

siempre que:

$$x[n] = 0 \qquad n < 0 \; y \; n > N-1$$

Al trabajar en el entorno de MATLAB, se utilizará la Transformada Rápida de Fourier (FFT) [13] mediante la función *fft(),* que consiste en una implementación algorítmica rápida y eficiente de la DFT a la hora de trabajar con el tratamiento digital de la señal.

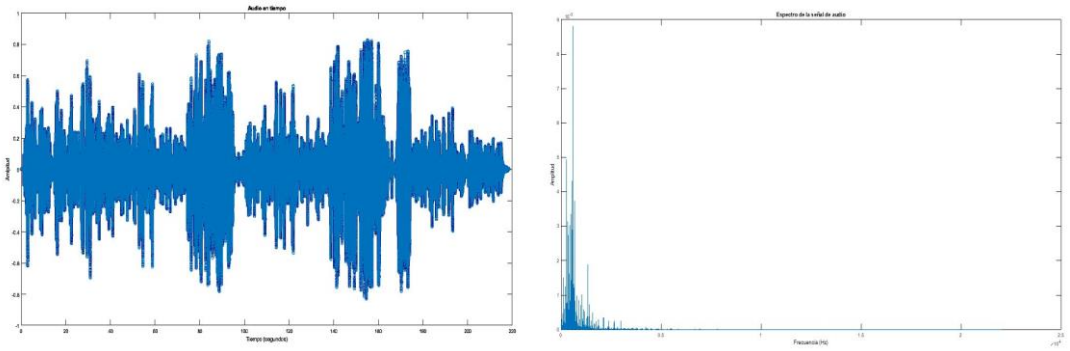


Ilustración 6: Audio en dominio temporal discreto

Ilustración 7: Audio en dominio frecuencial

Por otra parte, si se analiza la Transformada de Fourier de una señal discreta, se puede observar que la equivalencia de frecuencia digital (f_d) respecto a frecuencia analógica (f_a) es la siguiente:

$$f_d = \frac{f_a}{fs}$$

Ecuación 3 Frecuencia analógica y frecuencia digital [5]

Esta equivalencia será útil tanto para la representación de la FFT, como para la detección de parciales que se utilizarán en la codificación de las notas MIDI correspondientes.

Enventanado

Para un correcto cálculo de la DFT de una secuencia discreta, se debe partir de una señal limitada en tiempo cuyo valor sea nulo para instantes menores de *0* y mayores de *N-1.* Operacionalmente, este objetivo se consigue mediante el enventanado de la señal, que consiste en una multiplicación de la señal $x[n]$ por una secuencia $W[n]$ denominada ventana [14]. Existen diferentes tipos de funciones ventanas, las más utilizadas son:

Ventana	Ancho del lóbulo principal	Amplitud del lóbulo secundario
Rectangular	2/N	-13 dB
Triangular	4/N	-27 dB
Hanning	4/N	-32 dB
Hamming	6/N	-43 dB
Blackman	6/N	-58 dB

Tabla 2: Tipo de ventanas [5]

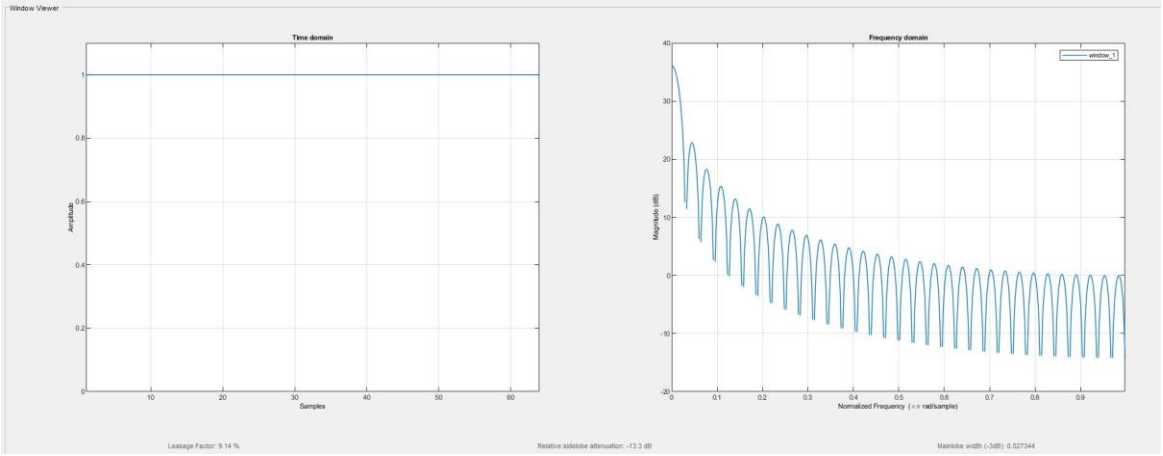


Ilustración 8: Ejemplo de ventana rectangular de 64 muestras (izquierda) y su representación frecuencial (derecha), generada a partir de la herramienta Window Designer de MATLAB.

Considerando que una multiplicación en el dominio temporal se transforma en una convolución de ambas transformadas en frecuencia [5], dicho enventanado implicará una alteración del módulo y la fase de la señal original en el dominio frecuencial, por lo que se debe tener en cuenta las características de la ventana dependiendo del resultado buscado.
Para solucionar el problema, en el módulo de cada componente frecuencial de la señal, como mínimo, es necesario eliminar la dependencia de la ventana empleada en el resultado. Esto se llevará a cabo mediante una normalización de su energía (que coincide con el valor del módulo de la transformada de la ventana evaluado en la frecuencia cero) con el fin de evitar alterar la información original de los parciales.

2.1.3 Síntesis

La síntesis es un proceso de generación de sonido mediante el procesado de señales procedentes de elementos no acústicos. Se pueden distinguir dos grandes categorías dependiendo de dichos elementos:

- **Síntesis analógica**: A partir del procesado mediante elementos y señales analógicas.
- **Síntesis digital**: A partir del procesado mediante elementos y señales digitales.

Para este proyecto se utilizará la síntesis de audio digital, y en concreto, la síntesis en frecuencia aditiva [15]. Esta técnica con fundamento en el desarrollo en serie de Fourier de señales periódicas, permite, a partir de la superposición de señales simples (en este proyecto señales cosenos) con las frecuencias de los parciales seleccionados, una reconstrucción de la información relevante de la señal original.

Se define por tanto la síntesis aditiva empleada como:

$$X(n) = \sum_{k=0}^{N} A_k \cos\left(\frac{2\pi f_k}{f_s} n + \theta_k\right) \qquad k = 0, \ldots, N$$

Ecuación 4: Síntesis aditiva digital [16]

donde:

A_k corresponde a la amplitud del parcial k.
f_k corresponde a la frecuencia instantánea del parcial k.
θ_k corresponde a la fase del parcial k (algunos modelos simplificados, como el considerado en este trabajo, no consideran las fases iniciales de cada parcial).

2.2 Procedimiento para el análisis

El proyecto está implementado sobre la herramienta de Mathworks "MATLAB" que permite la manipulación de datos en forma matricial. Para ello, se utilizarán matrices como representación de un archivo de audio de donde se obtendrá la información de entrada para realizar otros procesos entre los que se incluye la síntesis o la extracción de la melodía.

A partir de dicha información, se realizará el entramado (división temporal del audio en tramas) con un desplazamiento (o tamaño de trama) establecido previamente por el usuario como variable que configura el proceso. A cada trama o sección del audio de entrada se le hará una transformada discreta de Fourier haciendo uso de la FFT, con el fin de obtener la información que represente sus componentes frecuenciales.

Una vez la señal entramada esté en el dominio frecuencial, se extraerán las frecuencias correspondientes a sus parciales predominantes y sus correspondientes amplitudes (se ha optado por no considerar la información de fase para simplificar el procesado). Con las frecuencias representativas de cada parcial y mediante un mapeo a su correspondiente nota en la codificación MIDI, se almacenará como registro de nota encontrada.

A continuación, se volverá a sintetizar la señal con las notas encontradas de manera secuencial, obteniendo nuevamente la secuencia de notas predominantes inicial y con ello una representación de la información relevante del audio original. Además, a fin de comprobar la simplicidad de la aplicación de efectos sobre el tiempo y el *pitch* de la señal, se añadirá a la herramienta la posibilidad de realizar un procesado de dichos efectos formado por tres métodos diferentes de cambio de *pitch* y dos métodos de cambio de tiempo.

Para finalizar, una vez obtenida la información de los parciales, se procederá a la fase de extracción de la melodía mediante el procesado de la matriz de notas encontradas bajo ciertos supuestos sobre la composición de una melodía monofónica. Con el resultado de dicha extracción, se procede a la posterior codificación del archivo MIDI que contendrá la melodía principal del audio.

Para obtener una idea clara del proceso llevado a cabo, se han subdividido el proyecto en cinco fases correspondientes a sus relativas funciones.

2.2.1 Análisis del audio

Representación matricial del archivo de audio

A partir de un archivo de audio, se extrae la matriz correspondiente a sus niveles muestreados y una variable correspondiente a su frecuencia de muestreo f (muestras tomadas por segundo). Dicha matriz representará el audio muestreado en *n* filas, siendo *n* el número de muestras de la señal (instantes temporales), y *m* columnas, dependiendo del número de canales del audio a analizar. Si bien se podría realizar a cada uno de los canales de forma independiente, se ha optado por reducir el análisis al de una señal monocanal por lo que, en caso de partir de una señal multicanal se trabajará sobre la suma de sus canales.

	Amplitud Canal 1	Amplitud canal 2
1		
2		
3		
4		
5		
6		
7		
.	.	.
.	.	.
.	.	.
n		

Ilustración 9: Representación de la matriz de audio de n muestras con dos canales.

MATLAB, por defecto, representa la amplitud del audio con valores reales de tipo *double* [17] entre -1,0 y 1,0.

Representación temporal del archivo de audio

Para representar los valores de dicha matriz de audio respecto al eje temporal *t* (en segundos), se ha de definir un vector de 0 a *n-1* muestras. Partiendo de la toma de fs muestras en un segundo (o lo que es lo mismo, una muestra cada T_s segundos), si se requiere representar dicho vector temporal en segundos, será necesario dividir sus muestras por la frecuencia de muestreo fs (en hercios), obteniéndose la siguiente expresión:

$$t = (0{:}n-1) \cdot \frac{1}{f_s} = (0{:}n-1) \cdot T_s$$

Ecuación 5: Vector de tiempo

Una vez obtenido el vector temporal y el vector de amplitudes, se procede a representar el audio en un gráfico de líneas 2D utilizando la función *plot* de MATLAB [18] para visualizar su forma de onda.

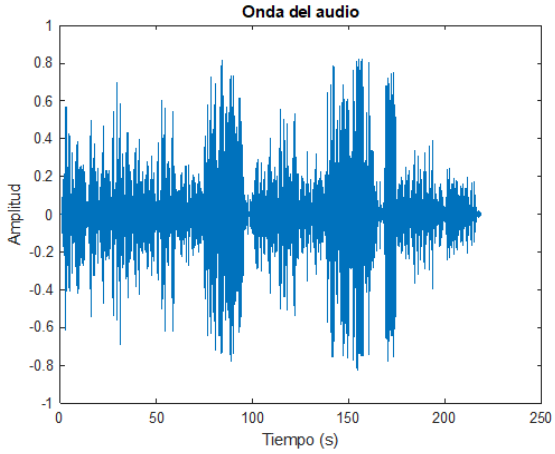


Ilustración 10: Representación del audio monofónico en el tiempo (para este ejemplo se ha representado el audio "ParaElisa.mp3" que se puede encontrar en "Ficheros Anexos/Melodias/ParaElisa.mp3")

Análisis en tiempo

Con el fin de realizar una detección de notas musicales en el tiempo, se ha optado por analizar el audio por tramas de una duración permanente. Estas tramas, además de permitir un análisis tiempo-frecuencia de una señal en tiempo cuasi periódica, definirán la máxima resolución temporal de la detección de notas. Si una nota es de duración menor al desplazamiento de *d* muestras (una trama), será detectada como la misma nota pero con una duración *d* o, en su defecto, será enmascarada por otra con mayor amplitud que suene en el mismo intervalo de tiempo.
Por tanto, dentro de una trama, solo se tendrán en cuenta las notas predominante para la detección de una melodía monofónica (parámetro establecido por el usuario) y, así mismo, si su duración es igual o inferior a *d*, se verá representada en la codificación como una nota de duración *d*.

Ilustración 11: Desplazamiento temporal de d muestras

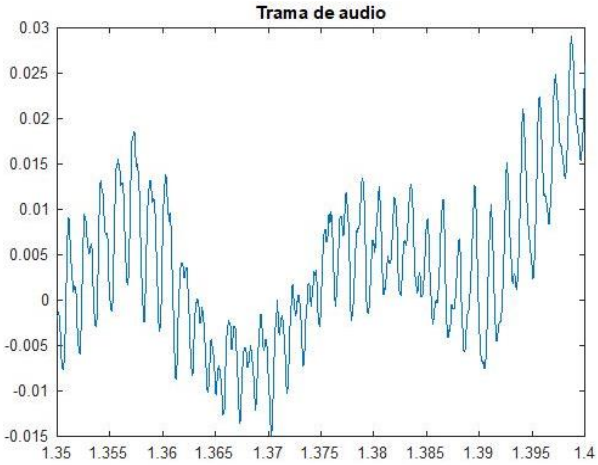


Ilustración 12: Trama de 0.05 segundos

Se define *d* como el redondeo al menor número de muestras del producto del desplazamiento en segundos Δt (especificado previamente como parámetro de entrada) y la frecuencia de muestreo f_s. Quedando de la siguiente manera:

$$d\ (en\ muestras) = \Delta t\ \cdot f_s$$

Ecuación 6: Desplazamiento en muestras

En este proyecto se ha optado por especificar Δt como un parámetro resultante del *tempo (Te)* de la señal musical en *beats por minuto* (*bpm*) y el valor de tiempo relativo de la mínima figura musical a detectar (T_f). Se procede a calcular Δt como:

$$\Delta t\ (en\ segundos) = \frac{Te}{60} \cdot T_f$$

Ecuación 7: Desplazamiento en segundos

A partir de las variables anteriores, se puede definir el número de desplazamientos o eventos ($N\_eventos$) en una trama de duración *n,* mediante el redondeo del resultado de la siguiente fórmula:

$$N\_eventos = \frac{n}{d}$$

Ecuación 8: Número de desplazamientos totales en un audio

Este número de eventos, junto con el parámetro de entrada del programa *N_notas* que establece la cantidad de notas a detectar en una polifonía dentro de una misma trama, definirá la matriz de notas detectadas en todo el audio para su posterior procesado con el fin de detectar una melodía.

	Nota dominante	Segunda nota	...	N_notas
1			...	
2			...	
3			...	
4			...	
5			...	
6			...	
7			...	
.	.	.	...	.
.	.	.	...	.
.	.	.	...	.
N_eventos			...	

Ilustración 13: Matriz de notas detectadas

Análisis en frecuencia

Una vez definida cada trama de audio a analizar a partir de la expresión de la Transformada de Fourier Discreta de una señal, se procede a calcular dicha transformada en cada trama que compone el conjunto de la matriz de la señal.
Si se parte de la definición para una señal de duración *N*:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \ldots, N-1$$

Ecuación 9: Transformada de Fourier Discreta

Extrapolando a cada trama se obtiene:

$$X_k = \sum_{n=Evento}^{(Evento+d)-1} x_n e^{-\frac{2\pi i}{d} kn} \quad k = 0, \ldots, d-1$$

Ecuación 10: Transformada de Fourier Discreta de cada trama

Donde:

k corresponde a la variable frecuencial discreta de la transformada (*bins*).
$Evento$ corresponde a la muestra inicial de la trama a analizar.
d corresponde al tamaño de la trama (desplazamiento).
x_n corresponde al valor de amplitud de cada muestra *n*.

Asimismo, se ha de considerar que, para obtener una correcta representación frecuencial mediante la DFT, es necesario disponer de una señal acotada que cumpla las condiciones siguientes:

$$X(n) = 0 \; para \; n < 0 \text{ y } X(n) = 0 \; para \; n > N-1$$

Para ello, se hará uso de un enventanado en cada trama con el modelo de ventana de Hamming [19], cuya duración está definida como parámetro de entrada del programa. Dicho parámetro, influirá en el proceso de análisis, ya que una duración menor a *d* definirá una nueva duración de dicho desplazamiento igual a la de la ventana y, por tanto, cambiará la duración mínima detectable de una nota.
Se puede ver la representación de la ventana utilizada en la ilustración siguiente:

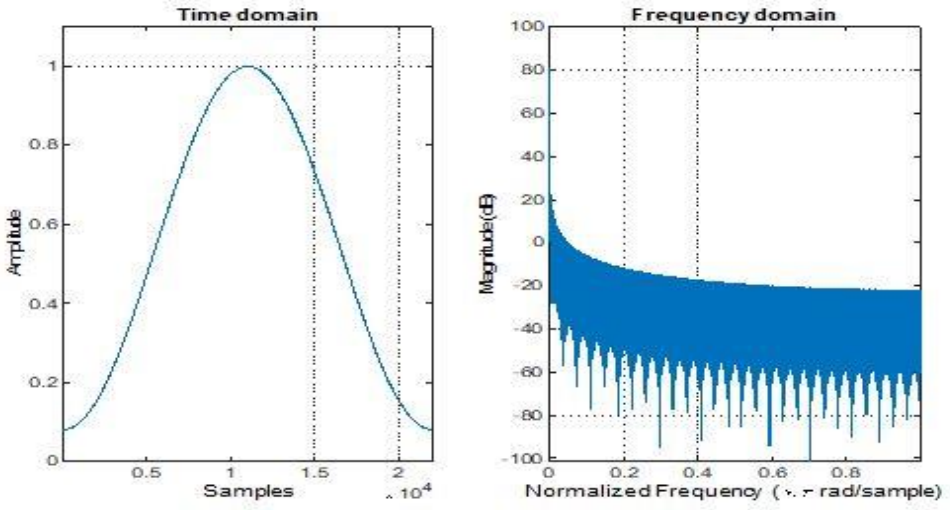


Ilustración 14: Ventana de Hamming de 0.5 segundos y frecuencia de muestreo de 44100 Hz sobre el dominio temporal en muestras (izquierda). Ventana de Hamming sobre el dominio frecuencial en radianes por muestra (derecha).

Cabe destacar que en el algoritmo utilizado para el análisis tiempo-frecuencia de este proyecto, se ha optado por realizar una FFT con un número total de puntos (*bins*) de 2 veces la duración de la ventana (en muestras) [19]. Esto implica una resolución frecuencial por *bin* Δf de:

$$\Delta f = \frac{F_s}{NFFT} = \frac{F_s}{2 \cdot d}$$

Ecuación 11: Resolución frecuencial

Donde:

F_s corresponde a la frecuencia de muestreo de la señal (en hercios).
$NFFT$ corresponde al tamaño de la transformada (en muestras).
d corresponde al tamaño de la trama o desplazamiento (en muestras).

Por lo tanto, la duración de la ventana o en su caso del desplazamiento, tendrá como resultado un compromiso inversamente proporcional entre la resolución frecuencial para una mejor estimación de la frecuencia de los parciales (con tramas de mayor duración) y la resolución temporal para la detección de notas cortas (con tramas de menor duración) [19].

Tras la representación frecuencial de cada trama mediante enventanado, se obtiene el siguiente espectro:

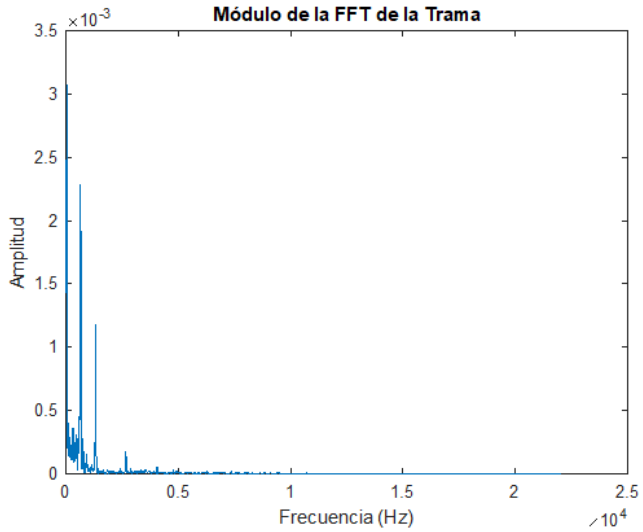


Ilustración 15: Transformada de Fourier de la trama comprendida entre 1.35 y 1.4 segundos.

Detección de parciales

A partir de la representación en frecuencia de la trama, se procede a identificar los picos correspondientes a los parciales de mayor amplitud. El algoritmo diseñado se encarga de detectar los *Num_notas* mayores picos, siendo *Num_notas* un parámetro de entrada del programa que define el número de parciales a detectar en una polifonía. Es decir, se parte de la hipótesis donde los picos detectados pueden ser interpretados como posibles frecuencias fundamentales de notas simultáneas o armónicos de una misma nota.

Con el fin de filtrar la cantidad de picos detectados debido a la sensibilidad de la función *findpeaks* [20] de MATLAB, el algoritmo discrimina todos los picos por debajo del valor cuadrático medio de la amplitud de todos sus parciales en dicha trama. Esto reduce significativamente el tiempo de procesado de la matriz que contiene los picos dentro de la trama, ya que se descartan los picos considerados irrelevantes, como componentes no tonales o de poca energía en la FFT.

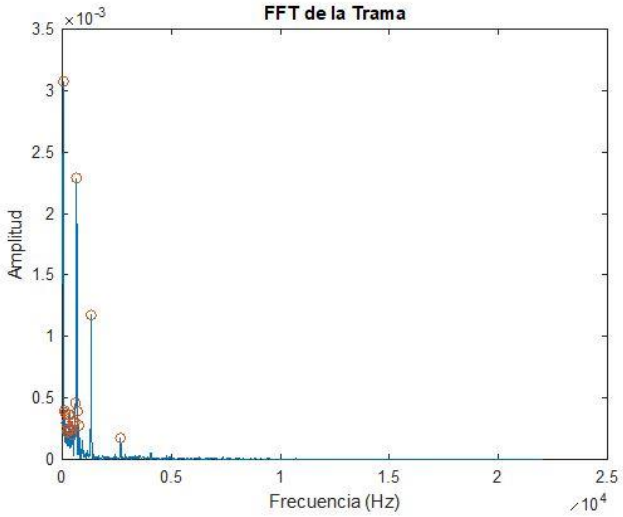


Ilustración 16: Picos detectados en la transformada de Fourier de la trama comprendida entre 1.35 y 1.4 segundos del audio "ParaElisa.mp3".

Mapeo de frecuencias a codificación MIDI

Para manipular los parciales detectados, se utilizará la representación de frecuencias utilizadas en la codificación MIDI. Esto implica que cada parcial, con una frecuencia determinada, será transformado en una nota MIDI con valor entre 0 y 127. La siguiente fórmula que relacionará las variables:

$$Nota\_MIDI = 21 + (12 \cdot log_2(\frac{f}{27{,}5}))$$

Ecuación 12: Mapeo de notas MIDI [21]

Tras el mapeo de frecuencias a codificación MIDI, se obtiene una matriz *N_eventos* x *Num_notas,* cuyas filas representarán los eventos (número de trama) concretos y cuyas columnas las *Num_notas* notas MIDI dominantes en dicha trama.

N_eventos	Nota dominante	Segunda nota	Tercera nota
1	0	0	0
2	0	0	0
3	115	95	116
4	62	113	27
5	112	104	107
6	108	110	110
7	111	110	106
8	113	109	112
9	114	93	77
10	111	112	110
11	110	115	110
12	111	98	114
13	65	107	109
14	110	112	109
15	114	105	107
16	112	113	114
17	116	112	111
18	106	110	114
19	107	112	109
20	109	110	114

Ilustración 17: 20 primeras tramas con sus respectivas 60 notas detectadas en el audio 'ParaElisa.mp3' con N_notas=3

Por otra parte, en cuanto a la extracción de la amplitud de cada parcial, se ha optado por no manipular la variable y extraerla como el valor del módulo de la FFT en su correspondiente frecuencia, quedando una matriz de *N_eventos* x *Num_notas* con el contenido de la amplitud de cada parcial.

Cabe destacar que, aunque el contenido de la fase de la FFT podría ser utilizado para el proceso posterior de síntesis, en el caso de la detección y codificación de la melodía, se considerará irrelevante y por tanto no será procesado.

N_eventos	Nota dominante	Segunda nota	Tercera nota
1	0	0	0
2	0	0	0
3	5,47E-07	5,45E-07	4,65E-07
4	8,98E-07	8,88E-07	8,55E-07
5	1,43E-06	1,25E-06	1,05E-06
6	1,19E-06	1,10E-06	1,02E-06
7	2,17E-06	2,12E-06	2,11E-06
8	1,15E-06	1,14E-06	1,13E-06
9	9,63E-07	8,46E-07	7,26E-07
10	1,43E-06	1,36E-06	1,34E-06
11	1,22E-06	1,18E-06	1,14E-06
12	1,18E-06	1,14E-06	1,07E-06
13	1,22E-06	1,17E-06	1,16E-06
14	1,23E-06	1,12E-06	1,09E-06
15	1,40E-06	1,05E-06	1,05E-06
16	1,29E-06	1,29E-06	1,04E-06
17	9,73E-07	9,08E-07	8,00E-07
18	1,06E-06	1,03E-06	8,81E-07
19	1,02E-06	1,02E-06	1,01E-06
20	1,85E-06	1,37E-06	1,29E-06

Ilustración 18: Matriz con las 60 primeras amplitudes de los parciales en el audio 'ParaElisa.mp3' con N_notas=3

2.2.2 Síntesis

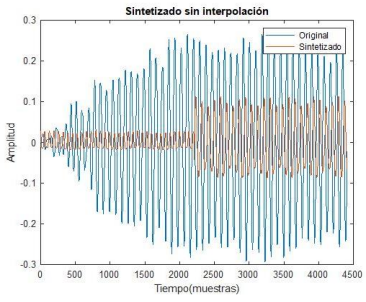
Con la información de los *N_Notas* parciales obtenida en la fase de análisis, se procede a sintetizar la nueva señal, esta vez habiendo discriminado el resto de la información frecuencial y temporal de la señal original. Para ello, se parte de una matriz de parciales codificados como notas MIDI y sus niveles de amplitud correspondientes, un tiempo de duración de trama utilizado en la fase de análisis y la frecuencia de muestreo del audio.

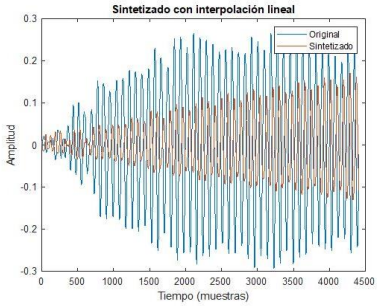
Posteriormente, se procede a sintetizar el audio cada *N_eventos* mediante síntesis aditiva de cosenos a frecuencias correspondientes a las notas MIDI sonando en una misma trama. Cabe aclarar que, aunque la señal musical no sea periódica, se considerará así a la hora de trabajar con el análisis tiempo-frecuencia.
Con el fin de evitar saltos de fase entre tramas, se ha optado por reorganizar la matriz de notas encontradas a la forma *matrizOrdenada*$_{(evento\ x\ nota\_MIDI)}$. Esta matriz contiene cada valor de amplitud del parcial *nota_MIDI (columnas)* en el *evento (filas)* correspondiente.
Quedando la matriz resultante de la siguiente forma:

N_eventos	108	109	110	111	112	113	114	115	116
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	5,47E-7	4,65E-7
4	0	0	0	0	0	8,88E-7	0	0	0
5	0	0	0	0	1,43E-6	0	0	0	0
6	1,19E-6	0	1,02E-6	0	0	0	0	0	0
7	0	0	2,12E-06	2,17E-06	0	0	0	0	0
8	0	1,14E-6	0	0	1,13E-6	1,15E-6	0	0	0
9	0	0	0	0	0	0	9,63E-7	0	0
10	0	0	1,34E-6	1,43E-6	1,36E-6	0	0	0	0
11	0	0	1,14E-6	0	0	0	0	1,18E-6	0
12	0	0	0	1,18E-6	0	0	1,07E-6	0	0
13	0	1,16E-6	0	0	0	0	0	0	0
14	0	1,09E-6	1,23E-6	0	1,12E-6	0	0	0	0
15	0	0	0	0	0	0	1,40E-6	0	0
16	0	0	0	0	1,29E-6	1,29E-6	1,04E-6	0	0
17	0	0	0	8,00E-07	9,08E-7	0	0	0	9,73E-7
18	0	0	1,03E-6	0	0	0	8,81E-7	0	0
19	0	1,01E-6	0	0	1,02E-6	0	0	0	0
20	0	1,85E-6	1,37E-6	0	0	0	1,29E-6	0	0

Ilustración 19: Matriz con las 180 primeras amplitudes formada por filas correspondientes al número de eventos y columnas correspondientes a las notas MIDI (desde 108 a 116) sonando en 'ParaElisa.mp3' con N_notas=3.

Por otra parte, en cuanto a la amplitud de la señal a sintetizar, se ha optado por suavizar los saltos entre tramas mediante una interpolación de los niveles de eventos consecutivos. Para ello se ha hecho pruebas con métodos de interpolación lineal y polinómica entre los extremos de tres tramas consecutivas. A continuación, se presenta la comparación del resultado de los diferentes tipos de algoritmos de interpolación utilizados frente a la señal original:





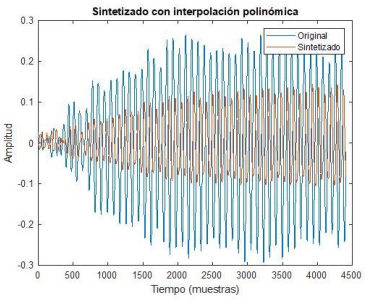


Ilustración 20: Trama número 49 a 51 del audio "ParaElisa.mp3" sin interpolación (izquierda).

Ilustración 21: Trama número 49 a 51 del audio "ParaElisa.mp3" con interpolación lineal (centro).

Ilustración 22: Trama número 49 a 51 del audio "ParaElisa.mp3" con interpolación polinómica de grado 2 (derecha)

Al no tener en cuenta la fase de la señal original en la síntesis, el error objetivo no se puede considerar como parámetro fiable para medir el error en el resultado, por lo que se ha optado por una interpolación lineal ya que presenta un mejor resultado a nivel subjetivo. Se puede verificar el resultado del estudio subjetivo adjunto en las conclusiones del proyecto (capítulo 3.3).

2.2.3 Aplicación de efectos en la síntesis

A partir de la extracción de parciales en el dominio frecuencial se procede, mediante una transformación no lineal de sus componentes frecuenciales, a la aplicación de efectos de audio. Este apartado pretende demostrar la facilidad de la aplicación de efectos tras la fase de análisis como pueden ser el cambio de *pitch* en una melodía o el cambio del tiempo de la señal.
A continuación, se proponen tres métodos diferentes de cambio de *pitch* y dos métodos de cambio de duración de la señal.

Modificación del *pitch*
Método 1: La forma más sencilla de modificar el *pitch* de una señal por un factor β tras el análisis inicial, consiste en *resamplear* [22] la señal mediante un cambio de frecuencia de muestreo. No obstante, este método produce modificaciones en el tiempo de la señal original, alargando o acortando la señal de forma inversamente proporcional al cambio de *pitch*.
Para paliar este efecto, se ha optado por utilizar un método de modificación temporal previa con el uso de un *phase vocoder* [19], que permite mantener a grandes rasgos el espectro original, seguido de un *resampleo* que cambiará el *pitch* de la señal por dicho factor β y restaurará su duración. Se define β como:

$$\beta = \frac{T_{final}}{T_{inicial}}$$

Ecuación 13: Factor de cambio de pitch mediante phase vocoder

Donde:

T_{final}corresponderá con la duración de la señal resultante antes de resamplear.
$T_{inicial}$corresponderá con la duración de la señal original.

Se ha optado por utilizar el algoritmo que describe el *phase vocoder,* ya que permite estimar y mantener la variación de fase de un parcial entre dos tramas consecutivas, a fin de evitar modificar la frecuencia del parcial al estrechar o ensanchar una señal en el tiempo.

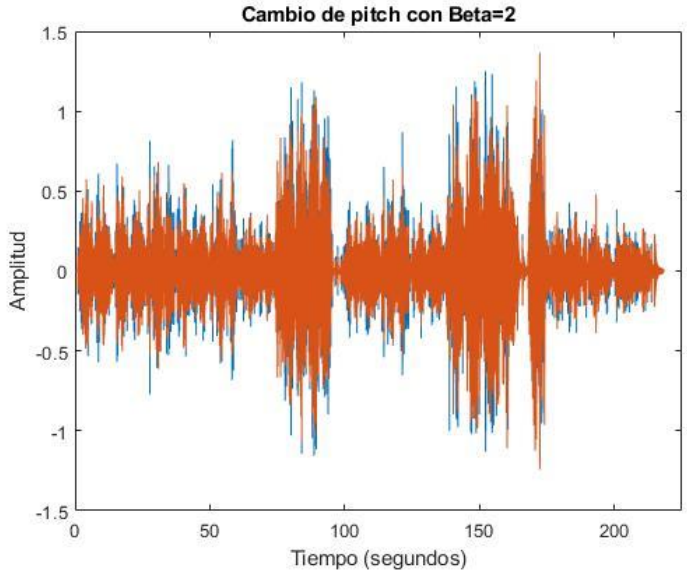


Ilustración 23: Audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch con β=2 (naranja).

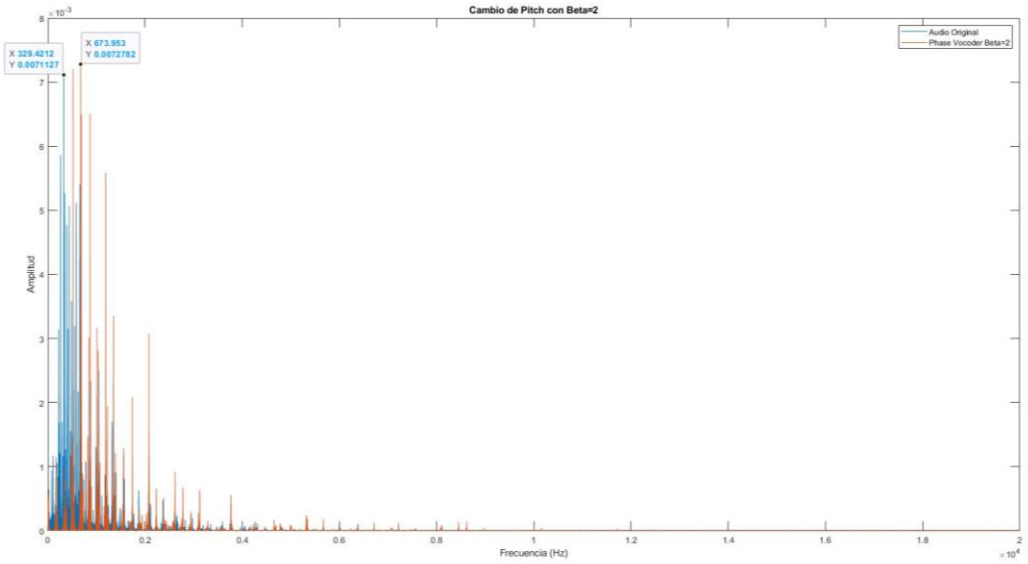


Ilustración 24: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch mediante el método 1 con β=2 (naranja).

Método 2: En segundo lugar, se propone un método alternativo de cambio de *pitch* mediante el desplazamiento de los parciales que componen la señal a una nueva frecuencia.
La nueva frecuencia de cada parcial será definida como:

$$f_{k\,final} = R \cdot f_{k\,inicial}$$

Ecuación 14: Factor de cambio de pitch mediante desplazamiento de parciales

Donde:

$f_{k\,final}$ corresponderá con la frecuencia del parcial *k* de la señal resultante.
$f_{k\,inicial}$ corresponderá con la frecuencia del parcial *k* de la señal original.

Además, se hará uso de la estimación de la fase de cada parcial mediante el algoritmo del *phase vocoder* para no inducir saltos de fase bruscos al realizar el desplazamiento [19].

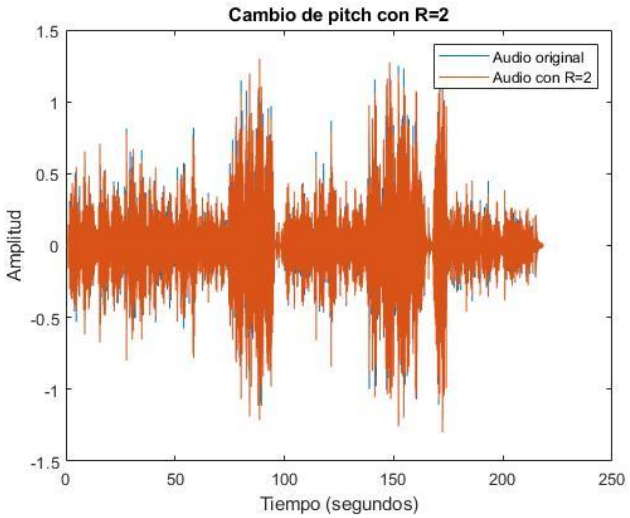


Ilustración 25: Audio "ParaElisa.mp3" original (azul) y tras un cambio de pitch con R=2 (naranja).

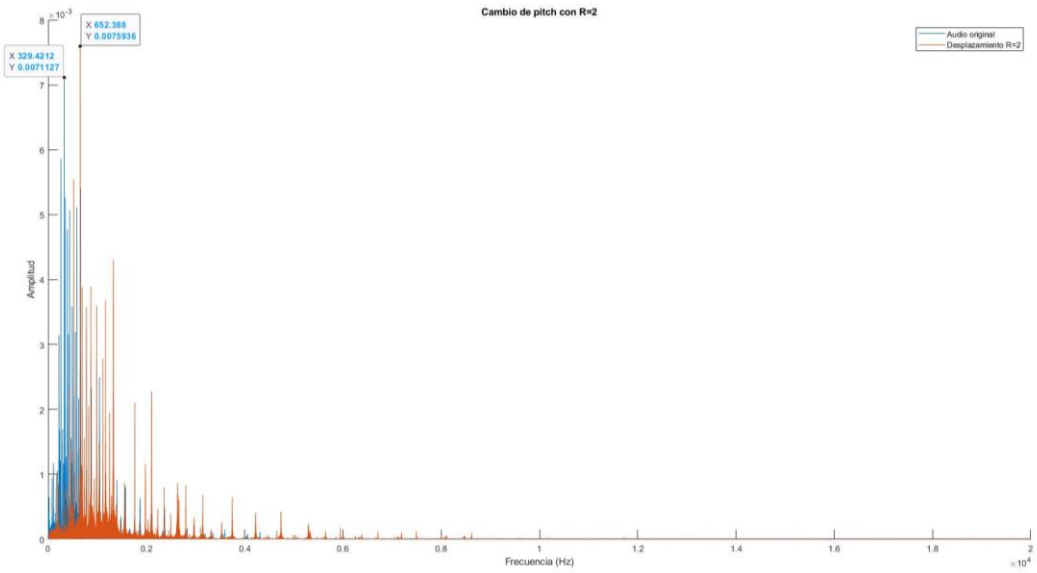


Ilustración 26: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch mediante el método 2 con R=2 (naranja).

Método 3: Para finalizar, se puede considerar el método de alteración de la codificación MIDI establecida para cada parcial predominante en escala musical, añadiendo un desplazamiento de $\pm D$ semitonos en la propia codificación. Esto implica una pérdida de información relevante a la hora de sintetizar la señal, ya que será necesario ejecutar previamente la etapa de análisis y obtener la matriz de *Num_Notas* notas MIDI. No obstante, se puede apreciar un cambio de *pitch* en el audio original si se realiza un análisis con el suficiente número de notas a codificar.
La nueva nota será codificada como:

$$N_{k\,final} = N_{k\,inicial} + D$$

Ecuación 15: Factor de cambio de pitch mediante suma de semitonos en la codificación.

Donde:

$N_{k\,final}$ corresponderá con la nota MIDI del parcial *k* de la señal resultante.
$N_{k\,inicial}$ corresponderá con la nota MIDI del parcial *k* de la señal original.

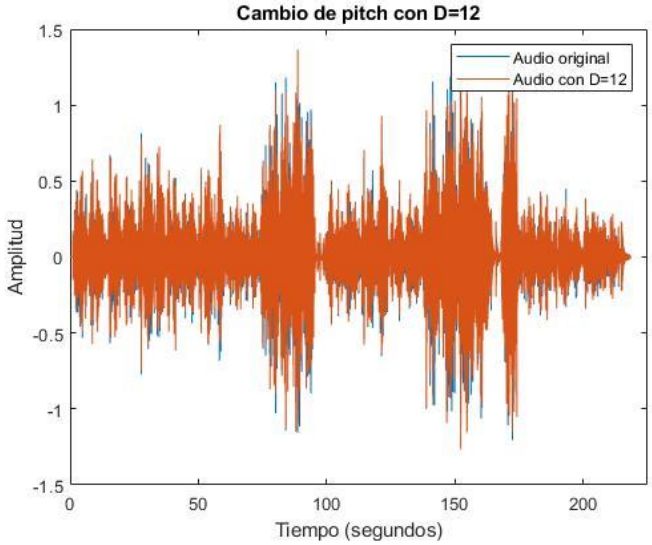


Ilustración 27: Audio "ParaElisa.mp3" original (azul) y tras un cambio de pitch con D=12 (naranja).

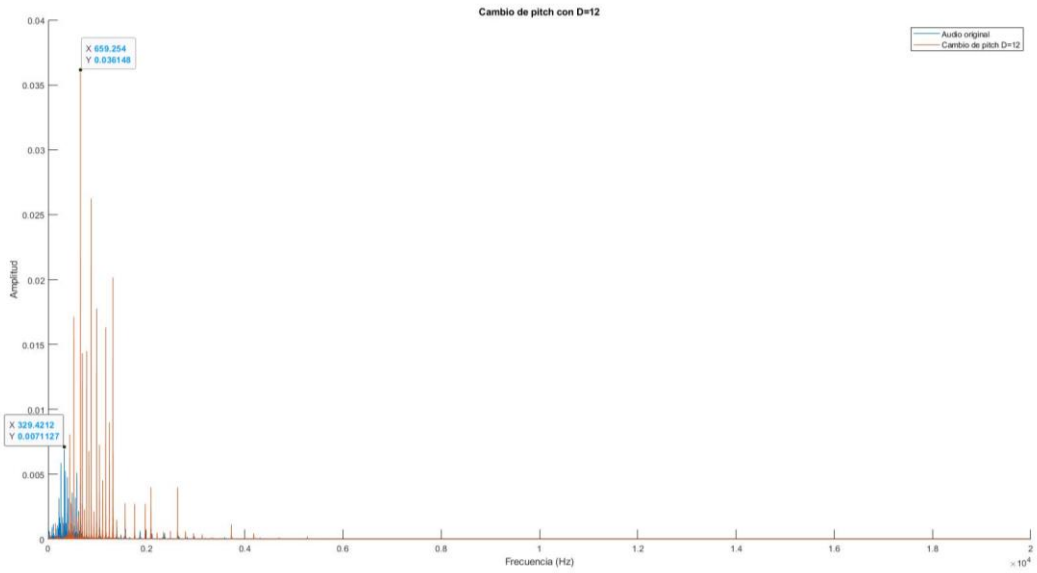


Ilustración 28: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch mediante el método 3 a una octava superior con D=12 (naranja) y Num_Notas=3.

Modificación del tiempo

Método 1: El primer método consiste en modificar el tiempo sin modificar el *pitch* de la señal original. Para ello se ha optado por emplear, al igual que en el caso del cambio de *pitch* el phase vocoder descrito en el apartado anterior, que permite ajustar la duración de la señal sin afectar a su contenido frecuencial [19].

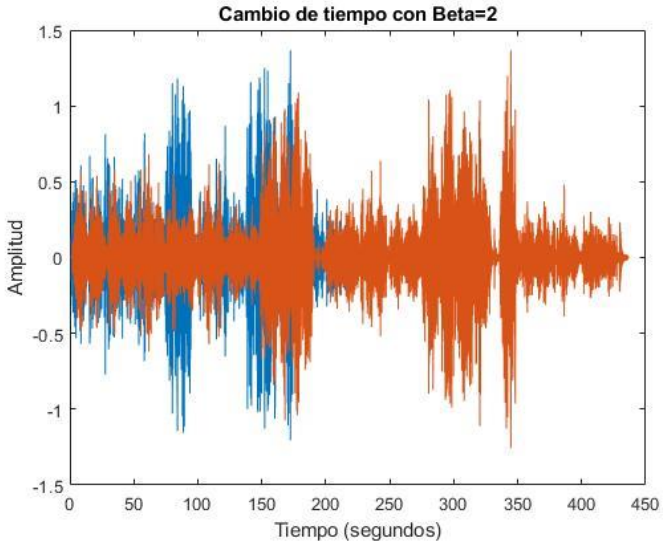


Ilustración 29: Audio "ParaElisa.mp3" original (azul) y tras un cambio de tiempo con β=2 (naranja).

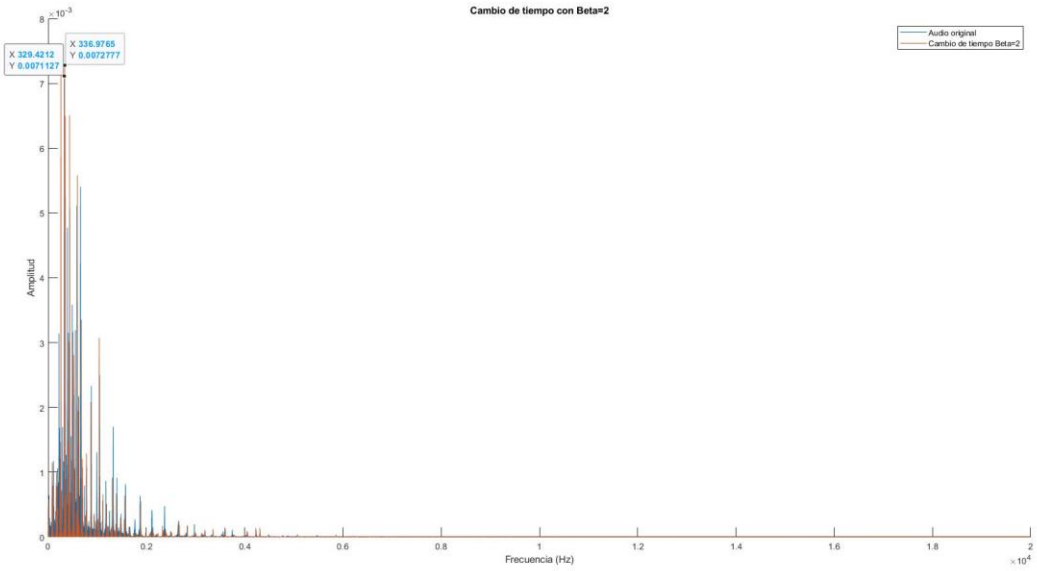


Ilustración 30: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del tiempo con β=2 (naranja).

Método 2: En segundo lugar, para modificar la duración de la señal por un factor R, se propone realizar una fase de análisis con tramas de duración *deltaTAnalisis* y *L* parciales a detectar (donde *L* corresponde la duración en muestras de la ventana de Hamming), para posteriormente volver a sintetizar la señal con una nueva ventana *deltaTAnalisisCambiada* definida como:

$$deltaTAnalisisCambiada = R \cdot deltaTAnalisis$$

Ecuación 16: Cálculo de la ventana de síntesis

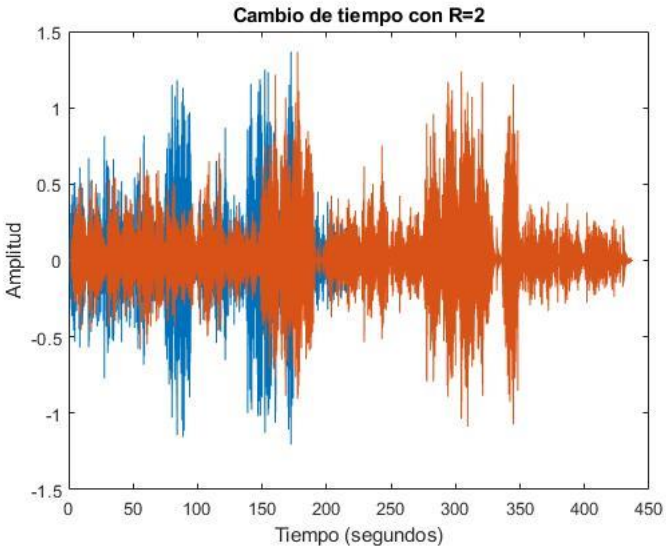


Ilustración 31: Audio "ParaElisa.mp3" original (azul) y tras un cambio de tiempo con R=2 (naranja).

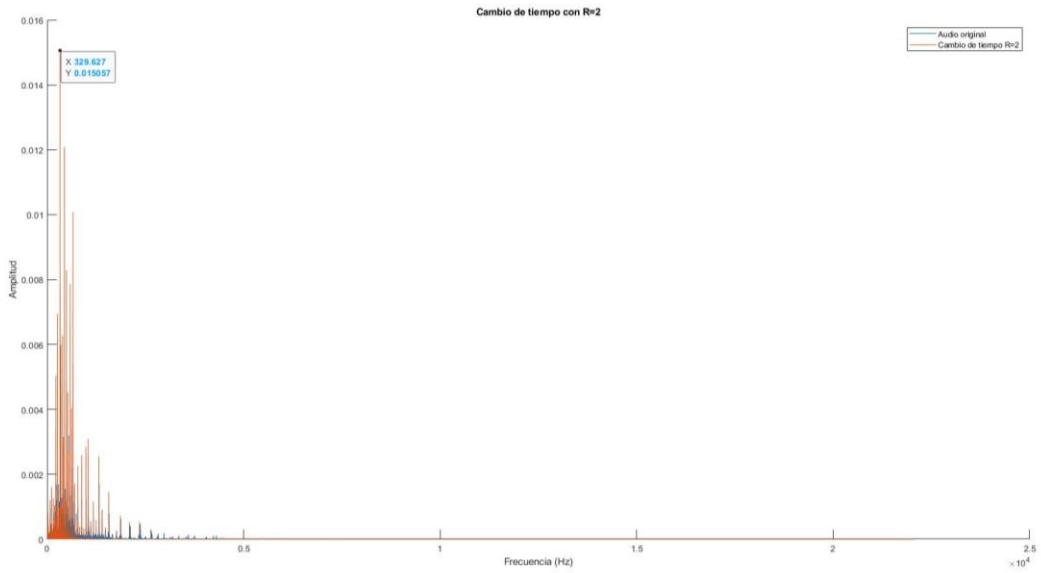


Ilustración 32: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del tiempo con R=2 (naranja).

2.2.4 Extracción de melodía

Tras haber obtenido las variables de salida de la fase de análisis, se procede a la fase de extracción de la melodía. Para ello se parte de las siguientes premisas que definen una melodía según [23]:

1. Mientras que el rango dinámico de un vibrato cantado es de ± 60~200 cents [24], es solo de ± 20~30 cents para instrumentos.
2. Las transiciones de una melodía están normalmente limitadas a una sola octava [25].
3. En general, un descanso de la voz principal tiene una duración mayor de 50 ms.

Dado que en este proyecto se ha utilizado una codificación de parciales en particular, se procede a transcribir dichas premisas a la codificación de notas MIDI. Para ello se hará uso de la fórmula utilizada anteriormente para el mapeo de frecuencias (Fórmula 1.21):

$$Nota\_MIDI = 21 + (12 \cdot log_2(\frac{f}{27{,}5}))$$

Ecuación 12: Mapeo de notas MIDI

Obteniendo las siguientes premisas que se adaptan correctamente a la codificación de este proyecto:

1. Mientras que el rango dinámico de un vibrato de voz es de ± 0.5~2 notas MIDI, es solo ± 0.2~0.3 notas MIDI para instrumentos. Lo que implica que, al utilizar un formato de codificación de notas MIDI entero, el vibrato de un instrumento será descartado, ya que estará incluido dentro de una misma nota MIDI.
2. Las transiciones entre melodías son típicamente limitadas a una octava o lo que es lo mismo ± 12 notas MIDI. Por tanto, se establece un rango máximo de variación de una melodía en 12 semitonos superiores y 12 semitonos inferiores a partir de la nota más repetitiva en la melodía que, por consiguiente, establecerá la octava principal.
3. En general, un descanso de la voz principal tiene una duración mayor de 50 ms, lo que implica que, al definir una duración de ventana de análisis de Δt, dicho descanso corresponderá al redondeo de $NtramasSilencio$ eventos:

$$NtramasSilencio = (50/1000)/\Delta t$$

Ecuación 17: Cálculo de tramas de silencio.

2.2.5 Codificación a mensajes MIDI

Mapeo de amplitud a codificación MIDI

En cuanto a la representación de la amplitud de las notas que componen la melodía, se ha optado por realizar un mapeo de la energía de cada parcial detectado sobre valores binarios entre 0 y 127 (valores de *velocity* en el estándar de codificación MIDI).

Para ello se considerará que el valor máximo de amplitud en toda la melodía (entre 0 y 1) corresponderá con el valor de 127. Por tanto, aunque en la codificación MIDI el parámetro *velocity* represente la sensación de fuerza con la que una nota es pulsada, a efectos prácticos para la herramienta, se define dicho parámetro como:

$$velocity = 127 \times \frac{A(n)}{max(A(n))} \qquad n = 1, \ldots, N\_eventos$$

Ecuación 18: Cálculo del parámetro velocity.

Donde:

$A(n)$, con valores entre 0 y 1, corresponde a la amplitud del parcial *n.*

n corresponde a la nota MIDI o evento particular que compone la melodía.

Creación del archivo MIDI

A partir de un mapeo de amplitud para ajustar la señal a la variable *velocity* de la codificación, se crea un archivo MIDI, cuya información, ordenada de forma secuencial, corresponde a mensajes con cada nota detectada. Cabe destacar que, al ser el estándar MIDI una codificación basada en eventos, el mensaje de estado de una nota no será emitido si esta no varía alguno de sus parámetros *velocity* u *ON/OFF.*

TRACK	CHANNEL	NOTE ON|OFF	NOTE	VELOCITY

Ilustración 33: Estructura del mensaje MIDI utilizada en la codificación. Los parámetros track y channel se han establecido por defecto con un valor de 1.

Además, para identificar el inicio y el final de cada nota, será necesario realizar un procesado de la matriz que representa la melodía antes de proceder a la codificación. Para ello, se ha observado que, si una nota es pulsada, el nivel de amplitud en dicho evento y parcial suele ser mayor al de sus vecinos debido a la propia naturaleza de la envolvente [26].

Se puede observar una representación de notas detectadas utilizado este procesado sobre la matriz de la melodía en la siguiente ilustración:

melodia

4370x2 double

	1	2	3
16	0	0	
17	0	0	
18	0	0	SILENCIO
19	0	0	
20	0	0	
21	0	0	
22	67	2.0710e-04	NOTA 1
23	76	0.0611	
24	76	0.0493	
25	76	0.0335	NOTA 2
26	76	0.0176	
27	76	0.0066	
28	76	0.0023	
29	76	0.0057	NOTA 3
30	75	0.0372	NOTA 4
31	75	0.0332	
32	75	0.0399	
33	75	0.0360	NOTA 4
34	75	0.0260	
35	75	0.0188	

Ilustración 34: Parte de la matriz representante de la melodía extraída del audio "ParaElisa.mp3" con una ventana de análisis de duración 0.05 segundos.

Para finalizar, se procede a crear el fichero MIDI con el contenido de la melodía a fin de transferirla a otro software o instrumento cómodamente. Para llevar a cabo esta codificación, se hará uso de la librería de scripts propuesta por Ken Schutte [27] en la que se define una función para dicho propósito.

2.3 Funciones utilizadas

FiltradoPrevio.m

La función *FiltradoPrevio* realiza el filtrado de la señal de entrada mediante un filtro de Butterworth Paso Alto de 5to orden. Su frecuencia de corte vendrá definida por el valor de la variable de entrada "f" (en hercios), que tendrá un valor por defecto de 150 Hz con el fin de eliminar bajas frecuencias (*Mid, Low* y *Sub bass* [28]) y así facilitar la detección de la melodía principal en la mayoría de los casos.

Ilustración 35: Diseño de la función FiltradoPrevio.m

Variables de entrada:

audioOriginal: Señal de audio de entrada.
FS: Frecuencia de muestreo del audio original.
f: Frecuencia de corte del filtro en hercios.

Variables de salida:

audioFiltrado: Señal de audio filtrado.

Analisis.m

La función *Analisis* es la función principal, que permitirá extraer los parámetros significativos de una señal de audio a partir de un análisis tiempo-frecuencia sobre el archivo de audio original. Para ello consta del siguiente diseño:

Ilustración 36: Diseño de la función Analisis.m

Variables de entrada:

archivoAudio: Señal de audio de entrada.
deltaT: Fracción temporal de análisis (en segundos). Viene determinada por la nota de menor duración a detectar.
durVentanaFFT: Duración de ventana de Hamming utilizada para el enventanado.
Num_Notas: Define el número de notas a detectar que suenan simultáneamente como polifonía.

Variables de salida:

Notas: Vector de notas MIDI organizado por orden de aparición temporal.
Nivel: Vector de niveles de cada nota ordenado por aparición temporal.
deltaTAnalisis*:* Fracción temporal de análisis utilizada dependiendo del valor de *deltaT* y *durVentanaFFT*.
FS: Frecuencia de muestreo del audio original.

Sintetiza.m

Esta función *padre* permitirá volver a sintetizar una señal similar a la original a partir los parámetros extraídos de una señal de audio en la fase de análisis. Para ello consta del siguiente diseño:

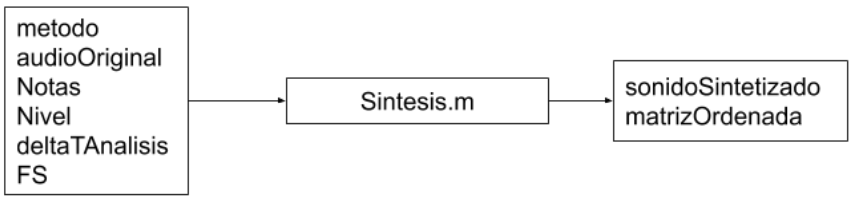


Ilustración 37: Diseño de la función Sintetiza.m

Se utiliza para llamar a la función de síntesis particular dependiendo del método elegido (síntesis sin interpolación entre tramas y con interpolación entre tramas). Dichas funciones son las siguientes:

Ilustración 38: Diseño de la función SintetizaSinInterpolar.m

Ilustración 39: Diseño de la función SintetizaInterpola.m

Variables de entrada:

audioOriginal: Señal de audio de entrada.
metodo*:* Método con o sin interpolación (valor entero de 1 o 2).
Notas*:* Matriz de notas MIDI obtenidas en el análisis.
Nivel: Matriz de niveles obtenidos en el análisis.
deltaTAnalisis*:* Fracción temporal de análisis utilizada.
FS: Frecuencia de muestreo del audio original.

Variables de salida:

sonidoSintetizado: Matriz de amplitudes que representan el sonido resultante de la síntesis.
matrizOrdenada: Matriz de *evento(fila), notas midi (columna), amplitud (valor).*

cambioDePitch.m

La herramienta de *cambioDePitch* es la función *padre* que permitirá aplicar modificaciones en el espectro de la señal para realizar un cambio de tono de la misma. Para ello consta del siguiente diseño:

Ilustración 40: Diseño de la función cambioDePitch.m

Se utiliza para procesar la matriz *Notas*, desplazando los semitonos correspondientes, o llamar a la función de cambio de *pitch* particular dependiendo del método elegido (mediante *phase vocoder y resampling o* mediante desplazamiento de parciales). Dichas funciones son las siguientes:

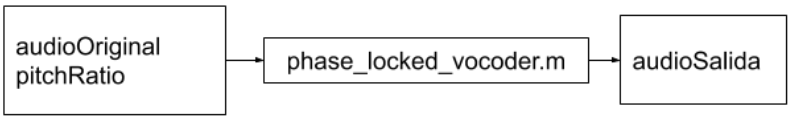


Ilustración 41: Diseño de la función phase_locked_vocoder.m

Ilustración 42: Diseño de la función desplazarFrec.m

Variables de entrada:

***audioOriginal*:** Señal de audio de entrada.
***FS**: Frecuencia de muestreo del audio original.*
metodo: Método mediante phase vocoder, desplazamiento de parciales o alteración de la codificación MIDI (1,2 y 3).
B (beta): Factor de estrechamiento/*pitch* del phase vocoder ($F_{final}=B*F_{inicial}$).
ratioEscalaFrec: Factor de desplazamiento de parciales ($F_{final}=ratioEscalaFrec*F_{inicial}$).
pitchEscalaMidi: Desplazamiento de $\pm pitchEscalaMidi$ semitonos sobre la codificación de la nota MIDI.
***deltaTAnalisis**:* Fracción temporal de análisis utilizada.

Variables de salida:

sonidoSintetizado: Matriz de amplitudes que representan el sonido resultante de la síntesis con efecto.
matrizOrdenada: Matriz de evento(fila), notas midi (columna), amplitud (valor).

cambioDeTiempo.m

La herramienta de cambio de tiempo es la función *padre* que permitirá volver a sintetizar una señal en tono similar a la original, pero con una duración definida por el parámetro R (tanto si se realiza mediante una síntesis con una nueva ventana temporal o mediante un *phase vocoder*).
Para ello consta del siguiente diseño:

Ilustración 43: Diseño de la función cambioDeTiempo.m

Se utiliza para llamar a la función de cambio de tiempo particular, dependiendo del método elegido (ventana temporal o phase vocoder). Dichas funciones son las siguientes:

Ilustración 44: Diseño de la función cambioDeTiempoVentana.m

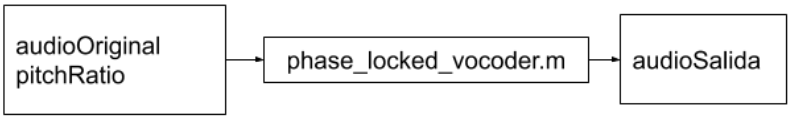


Ilustración 41: Diseño de la función phase_locked_vocoder.m

Variables de entrada:

audioOriginal: Señal de audio de entrada.
metodo*:* Método mediante cambio de ventana o phase vocoder (2 o 1)
deltaTAnalisis*:* Fracción temporal de análisis utilizada.
durVentanaFFT: Duración de ventana de Hamming utilizada para el enventanado.
R: Ratio estrechamiento de la señal a sintetizar. Definido como: $0 < R < \infty$. Modifica la duración de la señal de entrada de la forma: $T_{final}=R*T_{inicial}$.
FS: Frecuencia de muestreo del audio original.

Variables de salida:

sonidoSintetizadoTiempo: Matriz de amplitudes que representan el sonido resultante de la síntesis tras aplicarle el cambio de duración.

extraerMelodia.m

La función *extraerMelodia* es la herramienta que permitirá extraer la melodía principal de la matriz de parciales resultante de la fase de análisis cumpliendo con las premisas que definen una melodía expuestas en el apartado 2.2.4 Extracción de melodía.
Para ello consta del siguiente diseño:

Ilustración 45: Diseño de la función extraerMelodia.m

Variables de entrada:

***Notas**:* Matriz de notas MIDI obtenidas en el análisis.
Nivel: Matriz de niveles obtenidos en el análisis.
***deltaTAnalisis**:* Fracción temporal de análisis utilizada.
FS: Frecuencia de muestreo del audio original.

Variables de salida:

melodia: Matriz compuesta por las notas MIDI que componen la melodía detectada en la primera columna y sus respectivas amplitudes en la segunda columna.
rangoOctava: Rango máximo de la melodía ($\pm$12 semitonos a partir de la nota predominante).

Codifica_MIDI.m

La función de codificación MIDI es la herramienta que permitirá convertir la matriz ***matrizOrdenada*** en mensajes de eventos MIDI en función de los cambios en cada nota. Para ello consta del siguiente diseño:

Ilustración 46: Diseño de la función Codifica_MIDI.m

Cabe destacar, que su función principal es detectar el inicio y el final de cada nota para poder llamar correctamente a las funciones *matlab-midi* de Ken Schutte y codificar el archivo MIDI resultante.

Variables de entrada:

matrizOrdenada: Vector de niveles de cada nota ordenado por aparición temporal y frecuencia obtenida en la fase de síntesis.
***deltaTAnalisis**:* Fracción temporal de análisis utilizada.

Variables de salida:

rawbytes: Bytes de codificación que componen el archivo MIDI.

Funciones externas utilizadas

Tanto para llevar a cabo la creación de un archivo MIDI como para iniciar el desarrollo de un *phase vocoder*, se han utilizado funciones externas. Estas son:

- *matlab-midi* por Ken Schutte [27].
- *phase_locked_vocoder* (modificada) por T. Dutoit, J. Laroche [19].

Funciones de MATLAB utilizadas

Por otra parte, cabe destacar algunas de las funciones incluidas en MATLAB, que han sido de gran utilidad para el correcto funcionamiento del programa siendo estas:

- Nativas
  - audioread(): Leer archivo de audio. [29]
  - fft(): Transformada Rápida de Fourier. [30]
- Herramientas del complemento *Signal Processing Toolbox*
  - butter(): Diseño del filtro Butterworth. [31]

- filtfilt(): Filtrado digital de fase cero. [32]
- hamming():Ventana de Hamming. [33]
- findpeaks(): Encuentra máximos locales (picos) en un vector. [20]

2.4 Instalación y ejecución del programa

El proyecto ha sido programado en el entorno MATLAB, por lo que, si se opta por ejecutarlo sobre esta plataforma, se requerirá:

1. Instalar la versión de MATLAB *2019a* en la que se ha desarrollado el proyecto.
2. Descargar el *Toolbox* de MATLAB "*Signal Processing Toolbox*" para el procesado de señales.
3. Descargar la carpeta "Ficheros anexos" con las funciones necesarias, los ficheros de audio de prueba y sus respectivos archivos ".mid" resultantes.
4. Seleccionar como carpeta de trabajo la carpeta de ficheros descargada previamente.
5. Ejecutar la aplicación *TFG.mlapp*.

No obstante, para facilitar el uso de la herramienta a usuarios no familiarizados con el entorno MATLAB, se ha empaquetado la aplicación en un instalable independiente disponible como "Fitxeros Anexos/Aplicacion/Aplicacion Compilada/*TFG Installer.exe",* que permitirá su uso sin necesidad de realizar los pasos citados anteriormente.
Una vez instalada la aplicación, se podrá iniciar la herramienta ejecutando el archivo que se encontrará en "…*application/TFG.exe*" (dentro del directorio de instalación que se ha seleccionado).

2.5 Interfaz gráfico

Por lo que respecta al interfaz gráfico de la aplicación generado mediante MATLAB App Designer [34], se ha optado por incluir todas las funciones del proyecto en una misma ventana que se ha divido en tres secciones:

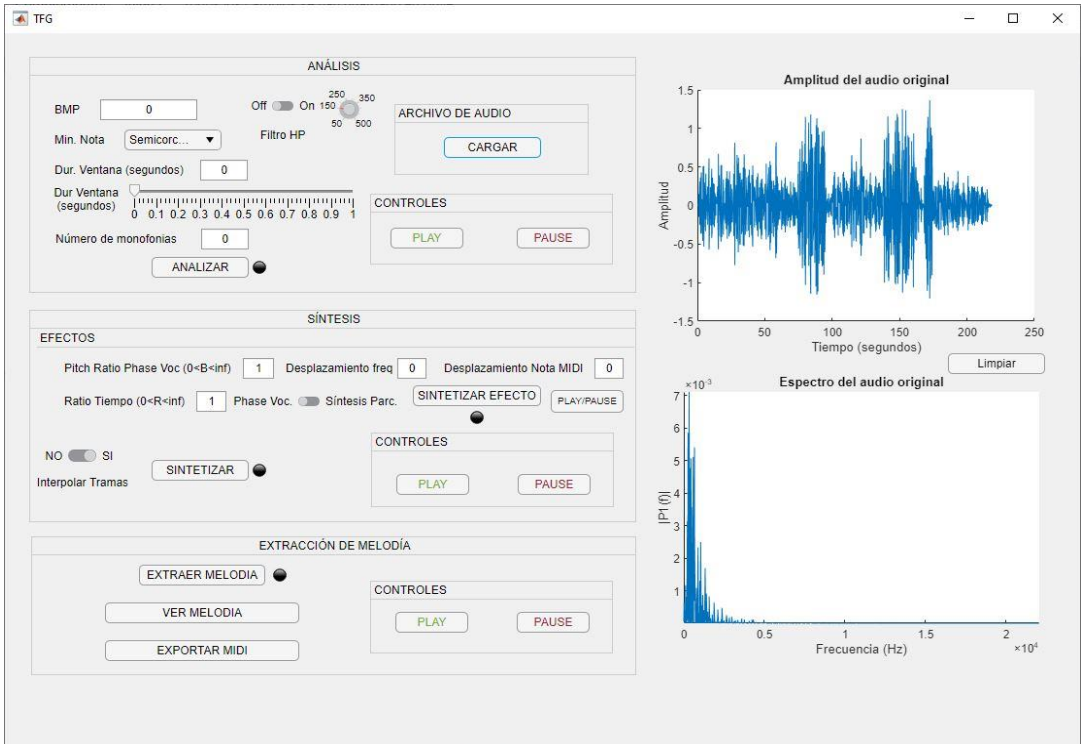


Ilustración 47: Interfaz principal del programa.

1. **ANÁLISIS**: Contiene los objetos de entrada necesarios para la fase de análisis. Está compuesta por los siguientes elementos:

    a. Botón CARGAR: Permite abrir el archivo de audio a analizar con formato WAV o MP3.

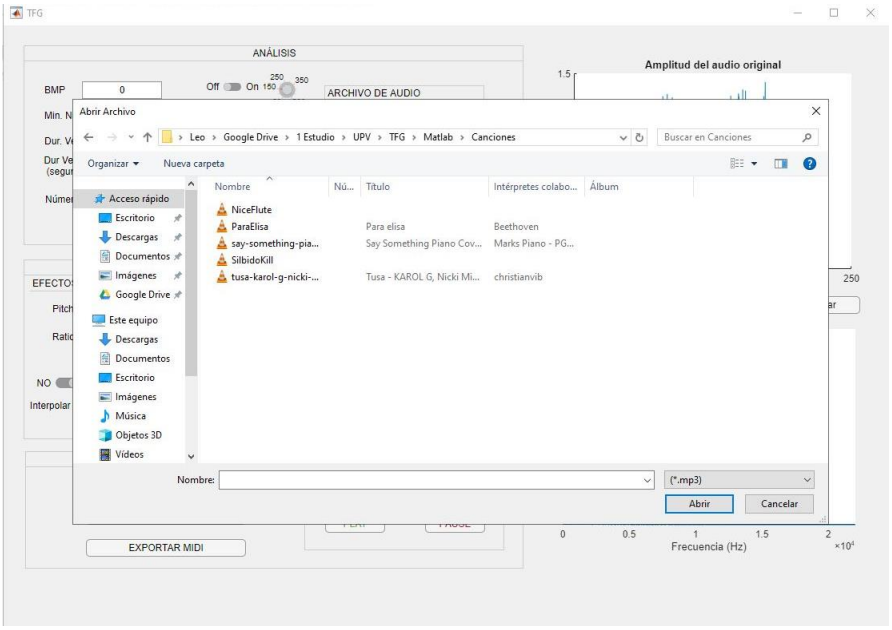


Ilustración 48: Cargar un nuevo archivo de audio en el programa.

    b. *Switch* Filtro *HP* On/Off: Activa o desactiva el filtro paso alto previo al análisis.
    c. *Knob* de frecuencia de corte: Establece la frecuencia de corte del filtro en hercios.
    d. *Input* BPM: *Beats* por minuto de la canción/audio a analizar.
    e. Min. Nota: Mínima figura musical a detectar en la melodía.
    f. *Input* Dur. Ventana: Entrada del valor de duración de la ventana para la FFT.
    g. Slider Dur. Ventana: Entrada gráfica del valor de duración de la ventana para la FFT como sugerencia de posibles valores.
    h. *Input* Número de monofonías: Número de parciales detectables en una trama.
    i. Controles: Sección de reproducción del audio original (filtrado en caso de tener el filtro paso alto activado).
    j. Botón *ANALIZAR*: Ejecutar la síntesis.

2. **SÍNTESIS**: Contiene los objetos de entrada necesarios para la fase de síntesis. Está compuesta por los siguientes elementos:

    a. Efectos
        i. *Input* Pitch Ratio Phase Voc.: Entrada numérica del valor β para el cambio de *pitch* mediante el *phase vocoder*.
        ii. *Input* Desplazamiento frec.: Entrada numérica del valor R para el cambio de *pitch* mediante el desplazamiento de parciales.
        iii. *Input* Desplazamiento Nota MIDI: Entrada numérica del valor D para el cambio de *pitch* mediante el desplazamiento de semitonos en la codificación MIDI.
        iv. *Input* Ratio Tiempo: Entrada numérica del valor β para el cambio de tiempo mediante el *phase vocoder* o cambio de ventana de síntesis.
        v. *Switch* Phase Voc. / Síntesis Parc.: Elección del método para el efecto de cambio de tiempo.
        vi. Botón *SINTETIZAR EFECTO*: Ejecutar la síntesis de los efectos e imprime el resultado y su espectro en los a su derecha.
        vii. Botón Play/Pause: Reproducción del audio tras los efectos.
    b. *Switch* Interpolar Tramas: Elección entre síntesis con o sin interpolación entre tramas.

c. Botón *SINTETIZAR*: Ejecutar la síntesis.
d. Controles: Sección de reproducción del audio sintetizado.

3. **EXTRACCIÓN DE MELODÍA**: Contiene los botones necesarios para la fase de extracción de melodía. Está compuesta por los siguientes elementos:

a. Botón *EXTRAER MELODÍA*: Ejecutar la extracción de la melodía principal.
b. Botón *VER MELODIA*: Imprimir el espectrograma de la melodía extraída.
c. Botón EXPORTAR MIDI: Abrir la interfaz para exportar la melodía extraída a un archivo *.mid.*

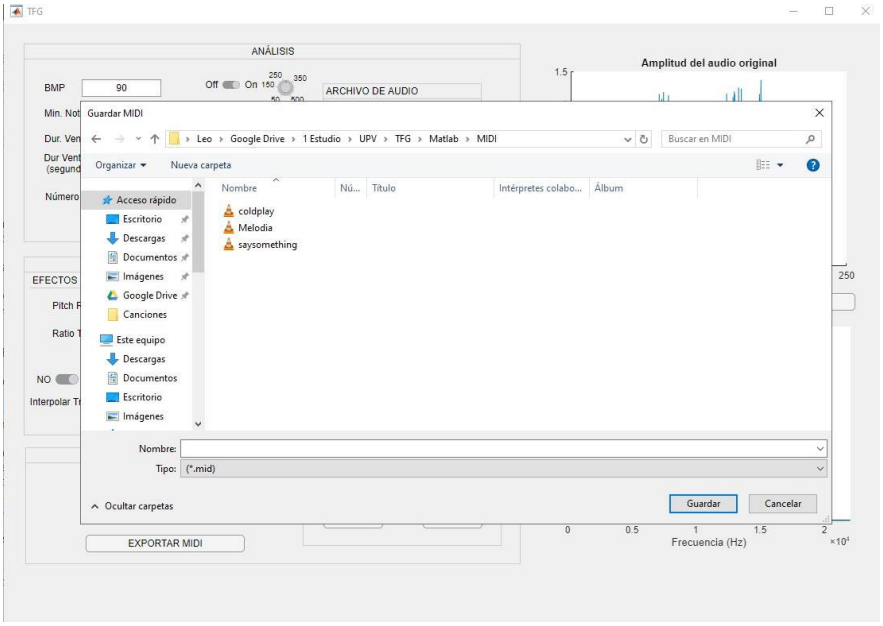


Ilustración 49: Guardar un nuevo archivo MIDI ".mid".

3. Conclusiones

Este trabajo ha sido de gran utilidad para comprender el proceso llevado a cabo en un análisis tiempo-frecuencia básico. A su vez, ha resultado apropiado para comprender la relación entre la representación temporal/frecuencial del audio y los eventos musicales útiles en la codificación MIDI empleada, así como también la síntesis aditiva a partir de dichos eventos.
Por otra parte, se ha comprobado a través de bibliografía utilizada y mediante su verificación con pruebas en la aplicación, el correcto funcionamiento de los supuestos sobre la detección de una melodía monofónica. No obstante, es posible mejorar el resultado como se comentará en el punto *3.6 Posibles mejoras* de este capítulo.

En cuanto al desarrollo de la herramienta, se puede concluir que, utilizando algoritmos simples, es posible llegar a obtener un resultado muy fiable cuando la composición musical es monofónica, por lo que es apropiada para el propósito que se creó (transcribir música desde un instrumento o cantando, o tarareando o silbando la melodía principal). Sin embargo, dadas las múltiples opciones y variaciones que se pueden obtener en una composición de melodías, el análisis y la detección serán sensibles al audio a analizar.
Además, el resultado se verá afectado negativamente tanto por el número de instrumentos sonando simultáneamente como por la amplitud de sus melodías o voces secundarias. Sobre todo, si están comprendidas en el rango frecuencial de la melodía principal como puede ser la voz humana.
Por tanto, si bien en condiciones ideales el funcionamiento de una herramienta sencilla sería óptimo, en la práctica requerirá de una mayor complejidad.

Una posible implementación futura de este proyecto podría ser generar una herramienta (o incluso un VST [35]) cuyo funcionamiento a *tiempo real* permita generar música sin necesidad de conocer el lenguaje musical. De tal forma que, tanto la interpretación vocal o instrumental de cada pista como el conjunto de melodías generadas, puedan ser sintetizada con el instrumento elegido por el usuario en su respectivo DAW.

A continuación, se procederá a especificar las conclusiones específicas de cada fase del proyecto, los problemas que han surgido durante la realización de este y posibles mejoras que podrían llevarse a cabo.

3.1 Análisis

En cuanto a la fase de análisis, se puede concluir que funciona de forma eficaz, detectando (aunque con cierto error implícito) las notas correctas en cada trama. No obstante, este error dependerá, tanto de los parámetros de entrada como del audio a analizar, por lo que puede ser optimizado trabajando sobre dichos parámetros. A su vez, una correcta elección de los parámetros, permitirá optimizar el compromiso resolución temporal/resolución frecuencial explicado en el capítulo 2.2.1 Análisis del audio.

Para evaluar su funcionamiento, se ha analizado un audio a modo de test, de un piano interpretando una melodía de ocho corcheas que incrementan un semitono desde C5 a G5 a 100BPM. Se puede encontrar en **"***Ficheros Anexos/Test/Test Analisis C5 Corch 100BPM.wav***"**.

En el caso del *test*, se sabe que con 8 notas sonando en 2,4 segundos, se obtiene que cada nota dura 0,3 segundos. Por tanto, se establecen los siguientes parámetros de entrada:

deltaT = 0.3 segundos.
durVentanaFFT = 0.3 segundos.
Num_Notas = 1 nota.

NOTAS REALES MIDI	RESULTADO	OBSERVACIÓN
0	0	Relleno con ceros de duración 1 trama
60	60	CORRECTO
61	61	CORRECTO
62	62	CORRECTO
63	62	INCORRECTO
64	64	CORRECTO
65	65	CORRECTO
66	66	CORRECTO
67	67	CORRECTO
67	67	Cola (*Fade out* de la última nota)
67	67	
67	67	
67	67	
67	67	

Tabla 3 Resultado del análisis con un audio de test.

El uso de ventanas de una duración menor podría mejorar el resultado ya que permitirá captar mejor el cambio de algunas notas. Sin embargo, se necesitaría un procesado posterior para identificar el inicio y el final de cada nota.

Se puede concluir que con el análisis se ha conseguido obtener de forma eficaz los parámetros relevantes para la codificación empleada, como pueden ser la amplitud o la nota que cada parcial seleccionado representa. Por otra parte, cabe destacar que, con mayor número de parciales, se podría obtener una mejor representación de la señal original con la consecuencia de una menor compresión de la información.

A continuación, se muestra: la información de la matriz del audio original "Test Sintesis Tusa 95 BPM.wav" (se puede encontrar en "*Ficheros Anexos/Test*"), de la matriz del audio codificado a partir de un análisis considerando sólo los 3 parciales más relevantes y de los parámetros necesarios para la posterior síntesis. Se puede verificar que el ratio de compresión [36] en este caso particular es de:

$$RC = \frac{Bytes\ sin\ comprimir}{Bytes\ comprimido} \qquad RC = \frac{14173056}{46096 + 8 + 8} = \frac{307.36}{1} \approx 307:1$$

Ecuación 19: Ratio de compresión del audio

```
K>> whos data
  Name            Size                Bytes  Class     Attributes

  data         885816x2             14173056  double

K>> whos matrizOrdenada
  Name                  Size             Bytes  Class     Attributes

  matrizOrdenada       67x86             46096  double

K>> whos deltaTAnalisis
  Name                  Size             Bytes  Class     Attributes

  deltaTAnalisis        1x1                  8  double

K>> whos FS
  Name       Size             Bytes  Class     Attributes

  FS         1x1                  8  double
```

Ilustración 50: Matrices de audio original vs. audio codificado (con Num_Notas=3) matrizOrdenada, ventana de análisis deltaTAnalisis y frecuencia de muestreo FS.

En cuanto al tiempo de ejecución del análisis, se representa a continuación su respuesta a la variación de los parámetros de entrada *Δt y Num_Notas,* sobre el audio de test "Test Tiempo Tusa 95 BPM.wav". Para ello, se ha medido el tiempo de ejecución del algoritmo con los siguientes vectores de prueba:

- *Δt* representado con un incremento de 0.01 segundos, desde 0.01 a 1 segundos (desde 441 a 44100 muestras en el caso del audio de test).
- *Num_Notas* representadas con un incremento de un parcial, desde 1 a 100 parciales a detectar por ejecución.

A efectos prácticos del test, se ha optado por fijar el tamaño de la FFT (*durVentanaFFT*) en 4096 muestras.

Se puede observar que, con un tamaño de la FFT fijo, el tiempo de ejecución del algoritmo es, en general, bastante uniforme (con cierta sensibilidad al número de notas a detectar). En el caso del audio de test, se vería un incremento en el tiempo de duración del análisis con un número de parciales a detectar inferior a diez, debido al procesado que realiza la función *findpeaks* de MATLAB para encontrar picos en la FFT.

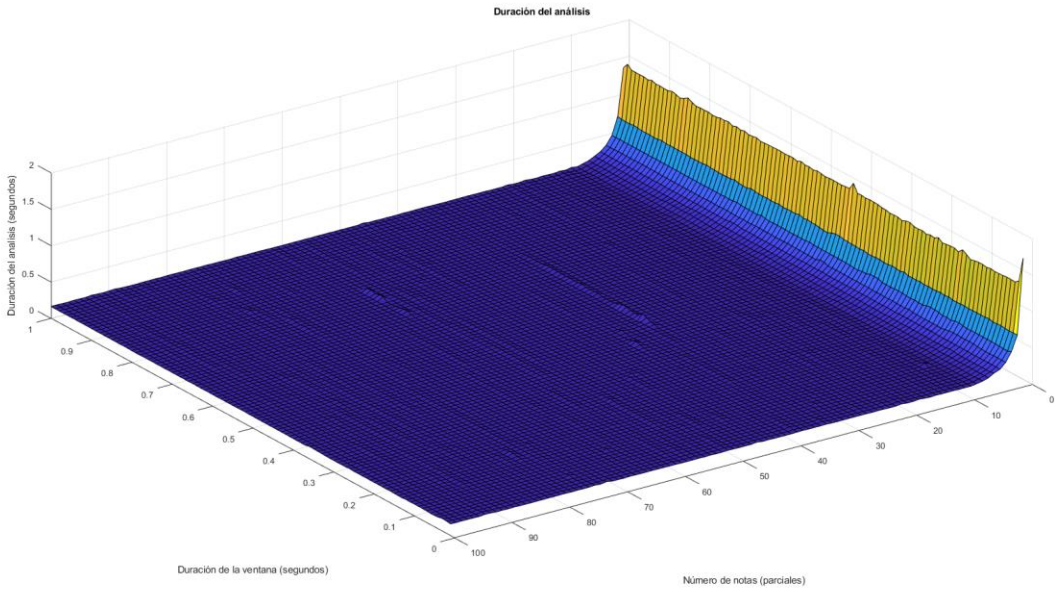


Ilustración 51: Tiempo de análisis dependiendo del tamaño de la ventana y el número de parciales a considerar.

3.2 Síntesis

Por lo que respecta a la síntesis, se puede comprobar que el modelo de extracción de los parciales con mayor amplitud, llevado a cabo en la fase de análisis, consigue representar la información relevante y comprimir así el audio original.
No obstante, conlleva un coste de tiempo de procesado y resulta en una pérdida de calidad del archivo original. Esto es debido a que el procesado en frecuencia empleado y la posterior síntesis, descartan información frecuencial y de fase, derivando en una reconstrucción acústicamente similar de cada trama de la señal original pero no igual.

Por otra parte, dado que se ha propuesto tres soluciones para suavizar los saltos de amplitud entre tramas durante la síntesis (capítulo 2.2.2 Síntesis), se ha realizado un estudio sobre los resultados comparando dichos métodos de forma subjetiva. Para ello se ha sintetizado tres versiones del audio "Test Síntesis Tusa 95 BPM.wav", a partir de un análisis de 10 parciales y con ventanas de 0.0625 segundos:

- **Versión 1**: Sin interpolación entre tramas (se puede encontrar en "*Ficheros Anexos/Test/Test Sintesis 1.wav*").
- **Versión 2**: Con interpolación lineal entre tramas (se puede encontrar en "*Ficheros Anexos/Test/Test Sintesis 2.wav*")
- **Versión 3**: Con interpolación polinómica de grado 2 entre tramas (se puede encontrar en "*Ficheros Anexos/Test/Test Sintesis 3.wav*")

Para seleccionar el mejor modelo de síntesis, se ha entrevistado a 20 usuarios anónimos a quienes se le ha permitido comparar y puntuar el parecido al audio original con una escala entre **1** (Nada parecido) y **5** (Totalmente parecido), aportando los siguientes resultados finales:

USUARIO	MÉTODO 1	MÉTODO 2	MÉTODO 3
Usuario 1	2	4	3
Usuario 2	2	4	3
Usuario 3	2	4	3
Usuario 4	1	3	4
Usuario 5	2	3	4
Usuario 6	4	2	2
Usuario 7	4	3	3
Usuario 8	4	2	2
Usuario 9	3	4	3
Usuario 10	4	5	3
Usuario 11	3	2	4
Usuario 12	4	3	4
Usuario 13	3	3	4
Usuario 14	4	3	3

Usuario 15	3	2	4
Usuario 16	3	3	4
Usuario 17	2	4	4
Usuario 18	3	4	3
Usuario 19	1	3	2
Usuario 20	3	4	2
Media de valoraciones	2.85	3.25	3.2

Tabla 4: Encuesta de valoración sobre los métodos de interpolación de amplitud propuestos. Se puede encontrar en: "Ficheros Anexos/Encuesta sobre interpolación entre tramas"

Por tanto, se podría concluir que el método de síntesis mejor valorado es el método 2, con interpolación lineal por lo que, hacer uso de interpolación de la amplitud entre tramas, influye de forma positiva en el resultado de la síntesis.

3.3 Efectos

Por lo que respecta a la implementación de efectos, se puede concluir que cada método de cambio de *pitch* conlleva su particularidad:

Como se aprecia en la Ilustración 24 e Ilustración 26, tanto el método 1 (cambio de *pitch* mediante *phase vocoder* y posterior *resampleo* de la señal) como el método 2 (cambio de *pitch* mediante el desplazamiento de parciales) tienen un contenido espectral similar al del audio original y mantienen en gran medida la amplitud del espectro. No obstante, al basarse en una estimación de la frecuencia de cada parcial y con una resolución frecuencial que dependerá del tamaño de ventana de detección utilizada, el resultado de un cambio de *pitch* con factor 2 en los picos del espectro no es exacto. Se puede observar en el primer caso, que al cambiar el *pitch* en una octava, el parcial indicado en la figura con frecuencia de 329.4 Hz (nota E4 del piano), pasaría a desplazarse a 673,9 Hz (entre las notas del piano E5 a 659.2 Hz y F5 a 698.4 Hz).

En cuanto al segundo método, el efecto en la frecuencia es similar, aunque con una desviación menor sobre la frecuencia de la octava superior. Esto puede producir un efecto de inarmonía en la combinación de acordes de la composición.

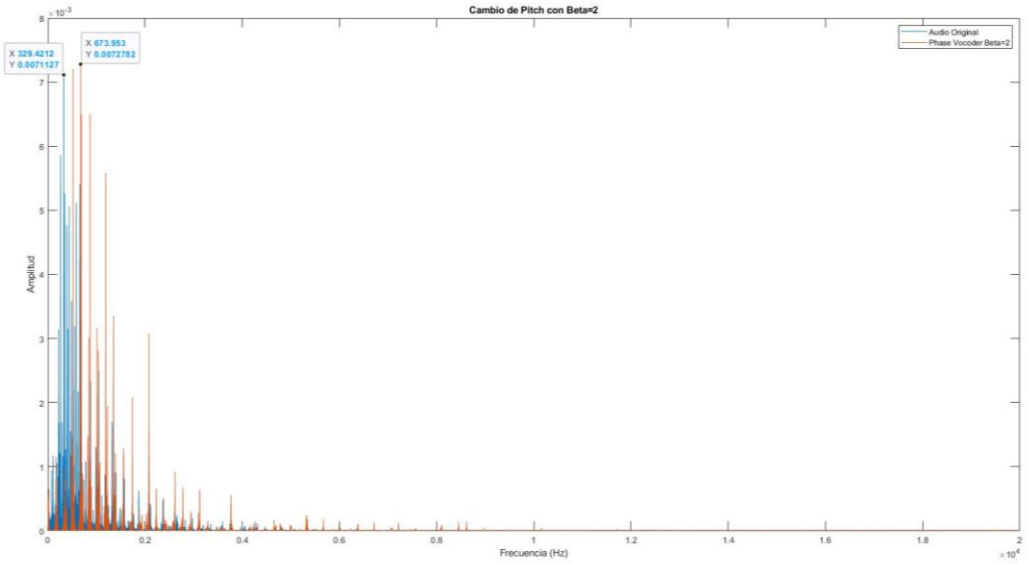


Ilustración 24: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch mediante el método 1 con β=2 (naranja).

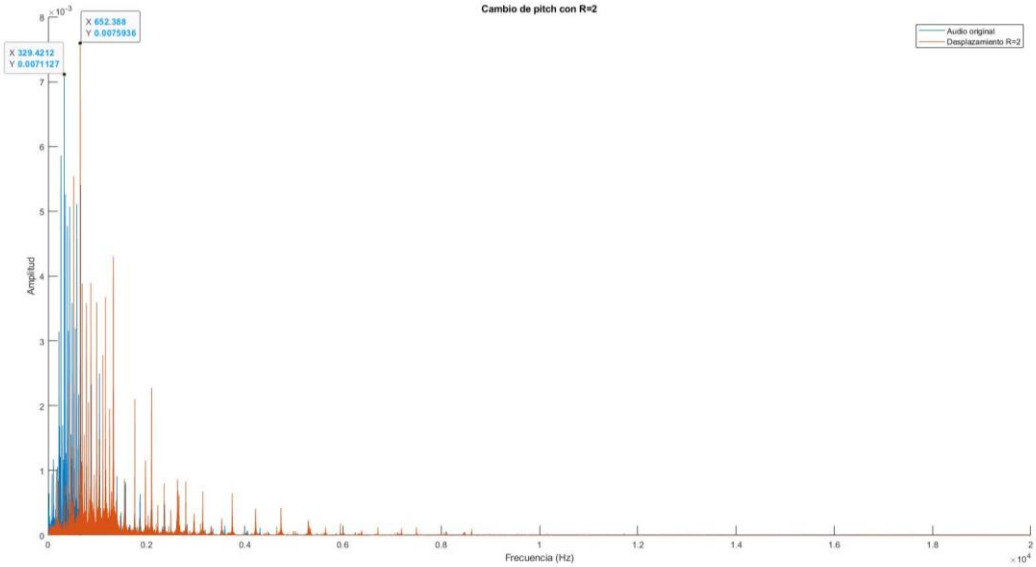


Ilustración 26: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch mediante el método 2 con R=2 (naranja).

Por otra parte, el método 3 (cambio de *pitch* mediante codificación MIDI) tendrá un contenido espectral más o menos similar al del audio original dependiendo del número de parciales que se consideren en el análisis. Se puede observar en la ilustración a continuación que, al cambiar el *pitch* en una octava, el parcial indicado en la figura con frecuencia de 329.36 Hz (nota E4 del piano) pasaría a desplazarse a 659,2 Hz (nota E5 del piano), por lo que se podría concluir que el método 3 es el más fiable para desplazar los parciales exactamente a la frecuencia correspondiente a un instrumento afinado. No obstante, si bien el cambio de *pitch* a una octava superior se puede considerar matemáticamente correcto, este método genera modificaciones notables la riqueza espectral original de los sonidos.

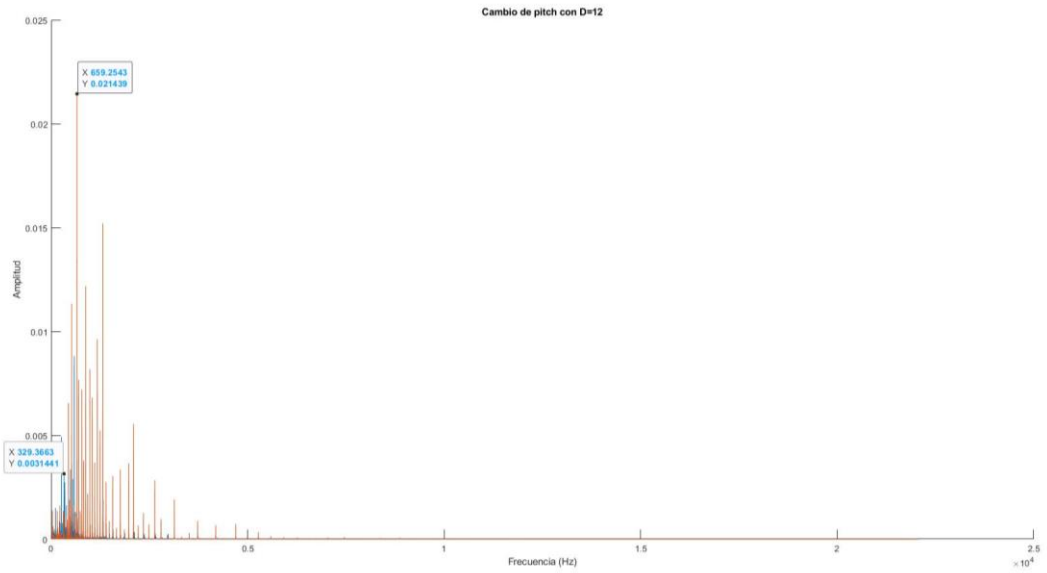


Ilustración 28: Espectro del audio "ParaElisa.mp3" original (azul) y tras un cambio del pitch mediante el método 3 a una octava superior con D=12 (naranja) y Num_Notas=3.

3.4 Extracción de la melodía

En cuanto a la extracción de la melodía, se puede concluir que, al ser esta fase dependiente del análisis, se ve directamente afectada por la elección de los parámetros de configuración de dicho análisis. Mayormente, es la correcta elección de la duración de las ventanas lo que permite reducir el error en la detección de la melodía.
A continuación, se muestra una comparación general de los diferentes resultados sobre un mismo audio ("ParaElisa.mp3") variando sus parámetros de entrada:

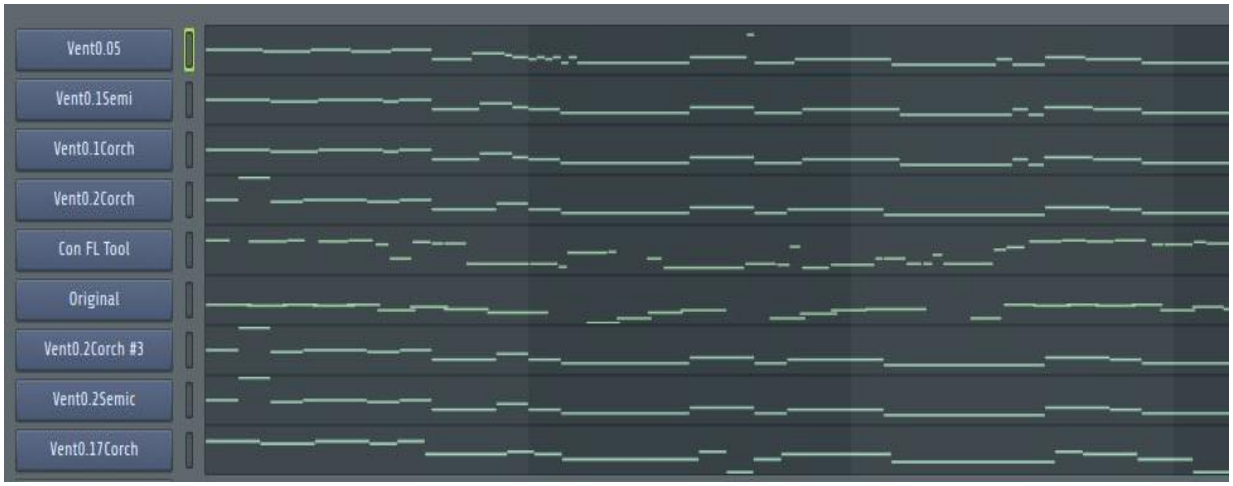


Ilustración 52: Melodías MIDI detectadas sobre un mismo audio al realizar variaciones en los parámetros de entrada del análisis.

Si se consideran unos parámetros de entrada lo suficientemente ajustados, se puede concluir que el resultado podría ser mejor que el obtenido por la herramienta del DAW FL Studio "*Convert to score and dump to piano roll*", cuya función transforma un audio en una melodía MIDI.
En las ilustraciones Ilustración 54 e Ilustración 55, se puede observar la melodía del audio "ParaElisa.mp3" detectada en este proyecto respecto al resultado obtenido con dicho *software*.

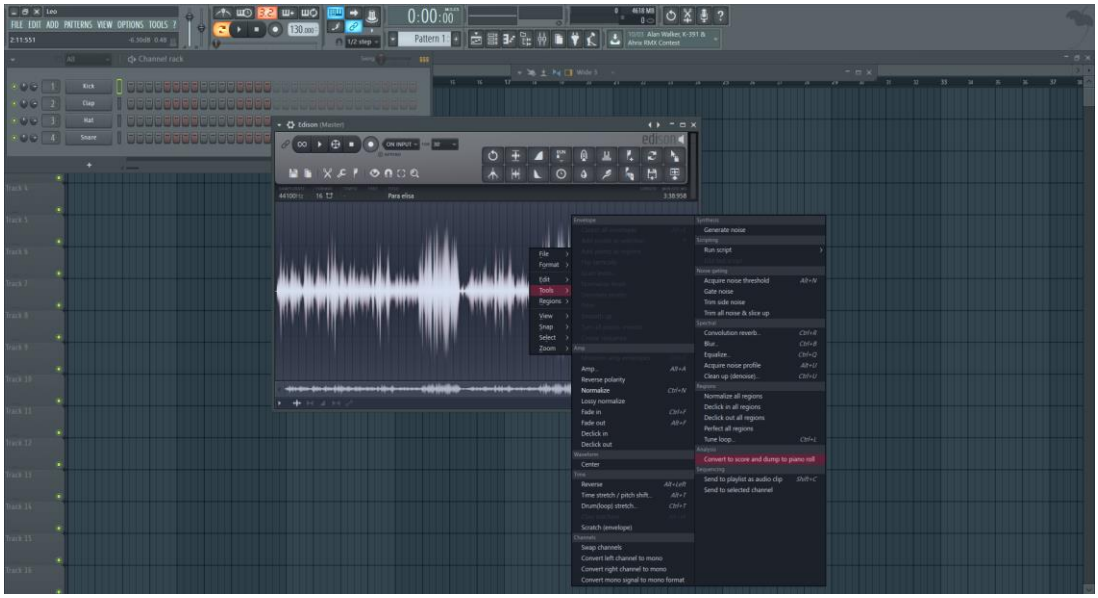


Ilustración 53: Herramienta "Convert to score and dump to piano roll"

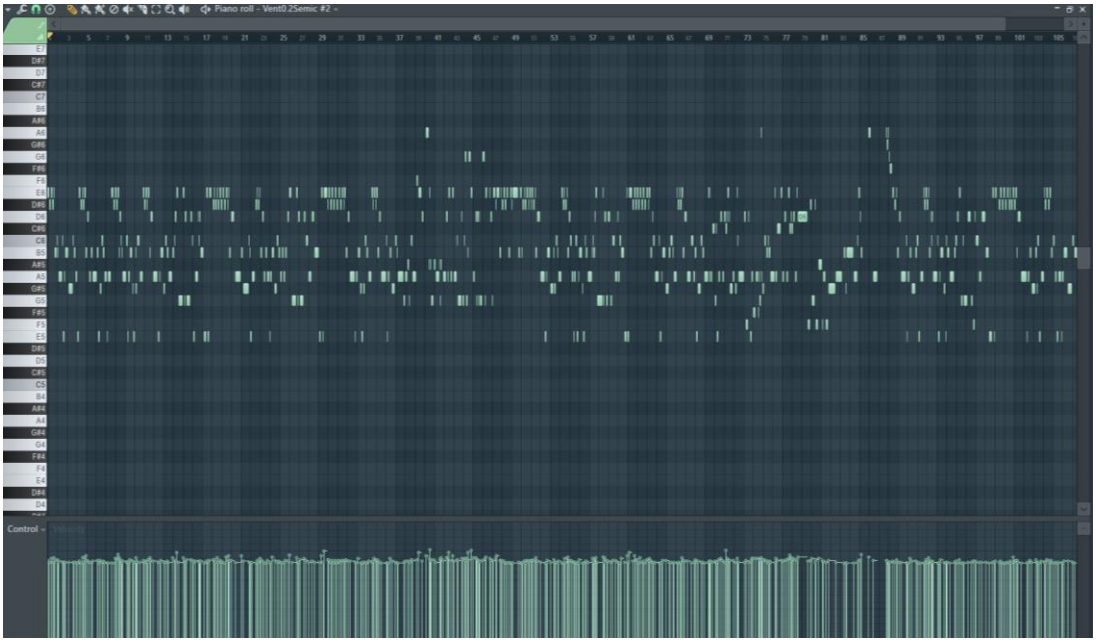


Ilustración 54: Resultado MIDI mediante la herramienta propuesta en este proyecto. Se ha realizado un análisis con BPM de 90; corchea como mínima nota a detectar; tamaño de ventana de 0.17 segundos; 2 monofonías y un filtro paso alto a 350 Hz.

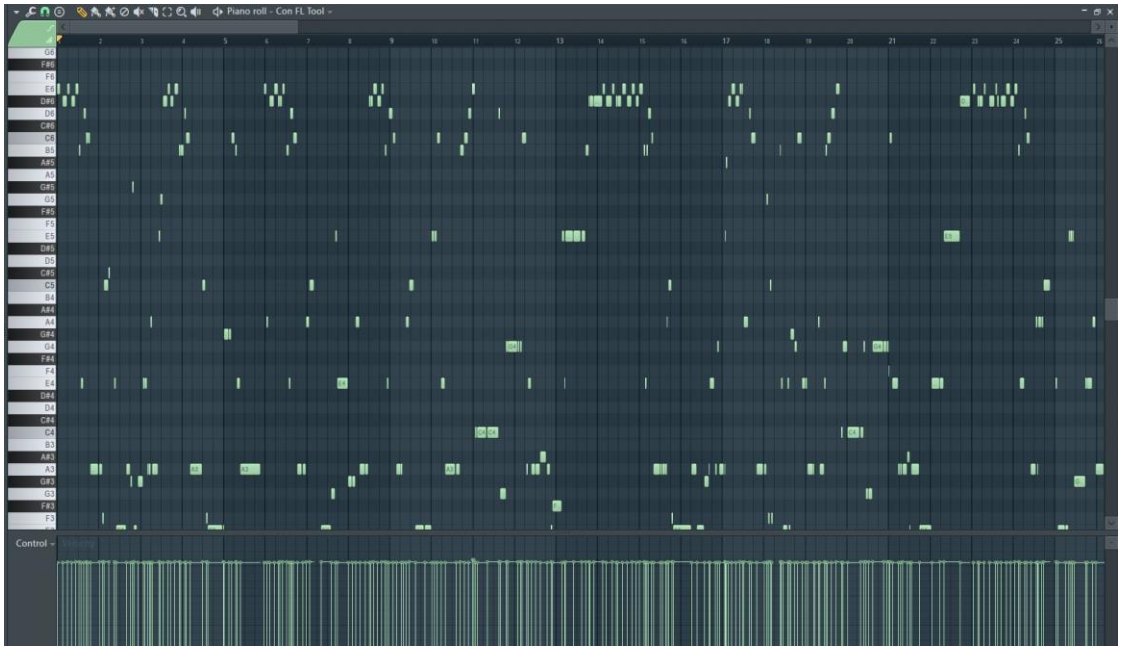


Ilustración 55: Resultado MIDI mediante la herramienta "Convert to score and dump to piano roll". Se puede apreciar que esta herramienta no realiza una correcta detección de la melodía principal ya que es probable que no tenga en consideración conceptos

Tras esta comparación, se puede concluir finalmente, que la herramienta realiza una correcta extracción de la melodía si su análisis también se realiza correctamente (se proponen mejoras para optimizar su resultado en el punto *3.6 Posibles mejoras* de este capítulo).

3.5 Problemas encontrados

Durante la realización del proyecto se han encontrado algunos problemas relacionados con la detección de la melodía.
En primer lugar, como se puede comprobar en la ilustración Ilustración 57, los parciales de baja frecuencia cuya amplitud puede ser mayor a la media, pueden inducir a errores si la melodía está comprendida en las bandas musicales superiores, dificultando la fase de extracción de la melodía principal. Para solucionar este problema, se ha añadido a la herramienta el filtro previo de Butterworth Paso Alto de 5to orden, que mejora el resultado eliminando aquellas frecuencias inferiores a la frecuencia de corte elegida por el usuario (entre 50 y 500 Hz).

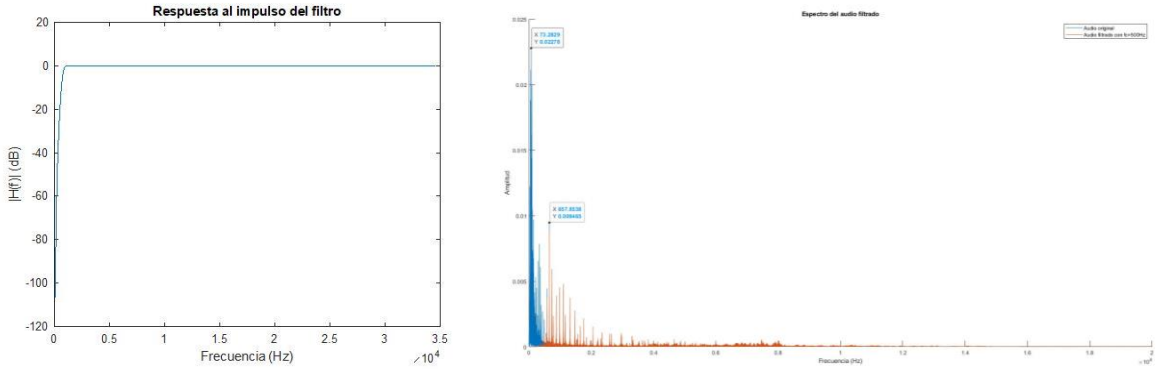


Ilustración 56: Respuesta en frecuencia del filtro de Butterworth paso alto con frecuencia de corte de 150 Hz (izquierda).

Ilustración 57: Espectro del audio "Test Sintesis Tusa 95 BPM.wav" (azul) respecto al mismo audio filtrado en 500 Hz (derecha).

A continuación, se muestra el resultado en la detección de la melodía (con y sin filtro) sobre un audio de ejemplo:

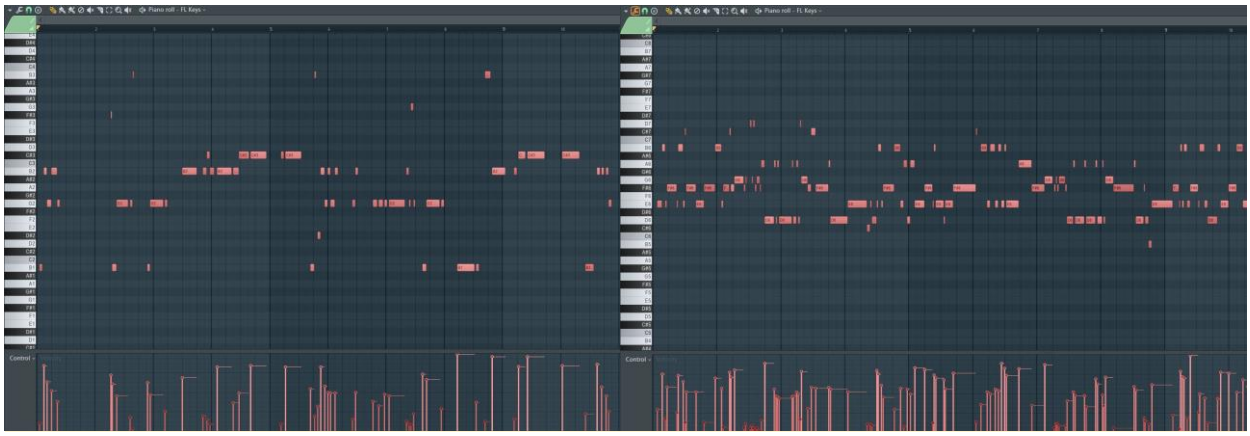


Ilustración 58: Melodía (sobre la segunda octava) detectada del audio "Test Sintesis Tusa 95 BPM.wav" sin filtro (izquierda).

Ilustración 59: Melodía principal (sobre la sexta octava del piano) detectada del audio "Test Sintesis Tusa 95 BPM.wav" con filtro paso alto en 500 Hz (derecha).

Por otra parte, otro problema que dificulta la detección se observa con audios compuestos por más de un instrumento sobre la banda de la voz principal. Al contener parciales que pueden enmascarar a la melodía en la fase de análisis, el resultado de la detección puede no ser el esperado. Para intentar resolver este problema se ha presentado una posible mejora en el capítulo a continuación.

3.6 Posibles mejoras

En cuanto a posibles mejoras de la herramienta, se propone una versión en la que se incorporen funcionalidades con grabación de audio y el desarrollo de algoritmos más avanzados que permitan:

1. Análisis con desplazamiento (ventana de detección) de duración variable, ajustado automáticamente al extraer los golpes y ataques de instrumentos en el tiempo. A su vez, esto se podría conseguir analizando la variación temporal de secciones del espectro, obtenidas mediante correlación con modelos armónicos de instrumentos o identificación de timbres mediante inteligencia artificial.
2. Síntesis con interpolación de frecuencias entre dos tramas y teniendo en cuenta la información de fase para suavizar los cambios de nota bruscos.
3. Síntesis por modelado físico [37] en la escucha de la melodía extraída, a fin de escucharla en otro instrumento generado mediante un modelo matemático.
4. Detección de la melodía principal de instrumentos específicos por medio de inteligencia artificial, utilizando modelos entrenados a partir de melodías en formato de audio y sus respectivas versiones MIDI monofónicas.

5. Bibliografía

[1] Z. Crockett, «The Mathematical Genius of Auto-Tune,» Priceonomics.com, 26 Septiembre 2016. [En línea]. Available: https://priceonomics.com/the-inventor-of-auto-tune/.

[2] Computer Music, «Early DAWs: the software that changed music production forever,» Musicradar.com, 20 Febrero 2020. [En línea]. Available: https://www.musicradar.com/news/early-daws-the-software-that-changed-music-production-forever.

[3] G. Arrizabalaga, Tecnología digital y evolución del producto musical, UPV/EHU, 2009.

[4] Wikipedia, «Time–frequency analysis for music signals,» Wikipedia, 14 Agosto 2019. [En línea]. Available: https://en.wikipedia.org/wiki/Time%E2%80%93frequency_analysis_for_music_signals.

[5] M. D. Girona Coma, Tratamiento digital de la señal [Material del aula], Gandía: Universitat Politècnica de València.

[6] M. F. Contreras, Tratamiento digital del audio [Material de aula], Gandía: Universitat Politècnica de València.

[7] Wikipedia, «Conversión analógica-digital,» Wikipedia, 1 Diciembre 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Conversi%C3%B3n_anal%C3%B3gica-digital.

[8] Wikipedia, «Resolución digital,» Wikipedia, 19 Diciembre 2019. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Resoluci%C3%B3n_digital&oldid=122137764.

[9] J. Hodgson, Understanding Records, Continuum Inter. Publis., 2010.

[10] Library of Congress, «Linear Pulse Code Modulated Audio (LPCM),» Library of Congress, 23 Diciembre 2015. [En línea]. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000011.shtml.

[11] The MIDI Manufacturers Association, The Complete MIDI 1.0 Detailed Specification, The MIDI Manufacturers Association, 2014.

[12] W. A. Sethares, «5. Transforms,» de *Rhythm and Transforms*, Springer, 2007.

[13] Wikipedia, «Transformada Rápida de Fourier,» Wikipedia, 30 Agosto 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Transformada_r%C3%A1pida_de_Fourier .

[14] National Instruments, «Comprender FFTs y Funciones Ventana,» 5 Marzo 2019. [En línea]. Available: https://www.ni.com/es-es/innovations/white-papers/06/understanding-ffts-and-windowing.html.

[15] M. Russ, «3.4 Additive synthesis,» de *Sound Synthesis and Sampling*, Focal Press, 2008.

[16] Wikipedia, «Additive synthesis,» Wikipedia, 2020 Abril 16. [En línea]. Available: https://en.wikipedia.org/wiki/Additive_synthesis.

[17] MathWorks, «Double-precision arrays,» MathWorks, [En línea]. Available: https://es.mathworks.com/help/matlab/ref/double.html.

[18] MathWorks, «2-D line plot,» MathWorks, [En línea]. Available: https://es.mathworks.com/help/matlab/ref/plot.html.

[19] J. L. T. Dutoit, «How does an audio effects processor perform pitch-shifting?,» de *Applied Signal Processing*, Springer, 2009.

[20] Mathworks, «Find local maxima,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/signal/ref/findpeaks.html.

[21] J. Wolfe, «Note names, MIDI numbers and frequencies,» Universidad de Nueva Gales del Sur (UNSW), [En línea]. Available: https://newt.phys.unsw.edu.au/jw/notes.html.

[22] Hydrogenaudio, «Resampling,» Hydrogenaudio, 6 Enero 2019. [En línea]. Available: https://wiki.hydrogenaud.io/index.php?title=Resampling.

[23] S. J. C. D. Y. Sihyun Joo, Melody extraction from poliphonic audio signal, EECS, Korea Advanced Institute of Science and Technology, 2009.

[24] Wikipedia, «Cent,» Wikipedia, 13 Septiembre 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Cent.

[25] Wikipedia, «Octava,» Wikipedia, 9 Abril 2020. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Especial:Citar&page=Octava&id=125026325&wpFormIdentifier=titleform.

[26] Á. Doménech, «Introducción a la adquisición, síntesis y procesamiento del audio - Envolventes,» Departament d'Informàtica de Sistemes i Computadors, UPV, [En línea]. Available: http://www.disca.upv.es/adomenec/IASPA/tema5/Embolicants.html.

[27] K. Schutte, «MATLAB and MIDI,» KenSchutte.com, [En línea]. Available: https://kenschutte.com/midi.

[28] Teach me audio , «Audio Spectrum,» 25 Abril 2020. [En línea]. Available: https://www.teachmeaudio.com/mixing/techniques/audio-spectrum.

[29] Mathworks, «Read audio files,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/matlab/ref/audioread.html.

[30] Mathworks, «Fast Fourier transform,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/matlab/ref/fft.html.

[31] Mathworks, «Butterworth filter design,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/signal/ref/butter.html.

[32] Mathworks, «Zero-phase digital filtering,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/signal/ref/filtfilt.html.

[33] Mathworks, «Hamming window,» Mathworks, [En línea]. Available: https://es.mathworks.com/help/signal/ref/hamming.html.

[34] Mathworks, «MATLAB App Designer,» Mathworks, [En línea]. Available: https://es.mathworks.com/products/matlab/app-designer.html.

[35] Wikipedia, «Virtual Studio Technology,» Wikipedia, Marzo 20 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Virtual_Studio_Technology.

[36] Wikipedia, «Data compression ratio,» Wikipedia, 2019 Diciembre 1. [En línea]. Available: https://en.wikipedia.org/wiki/Data_compression_ratio.

[37] Wikipedia, «Síntesis por modelado físico,» Wikipedia, 2 Marzo 2020. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=S%C3%ADntesis_por_modelado_f%C3%ADsico&oldid=123966847.