

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA



TESIS DE GRADO

**ALGORITMO PARA LA GENERACIÓN DE LA
PARTITURA DE UNA PIEZA MUSICAL BASADO EN EL
ANÁLISIS ARMÓNICO**

**PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: CIENCIAS DE LA COMPUTACIÓN**

POSTULANTE: LUIS ALEJANDRO ILLANES PINTO

TUTOR: Lic. BRÍGIDA ALEXANDRA CARVAJAL BLANCO

LA PAZ – BOLIVIA

2021



**UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA**



LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON EstrictAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

- a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
- b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
- c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

TODOS LOS DERECHOS RESERVADOS. EL USO NO AUTORIZADO DE LOS CONTENIDOS PUBLICADOS EN ESTE SITIO DERIVARA EN EL INICIO DE ACCIONES LEGALES CONTEMPLADOS EN LA LEY DE DERECHOS DE AUTOR.

DEDICATORIA

Dedico la presente tesis a mi familia por el cariño, paciencia y apoyo que me dieron todos estos años de estudio.

AGRADECIMIENTOS

A mi familia, por el apoyo incondicional durante todos estos años de estudio en la carrera.

A mi docente de la materia de Taller II Lic. Rodriguez Ramirez Grover por su guía y enseñanza, siempre dispuesto a ayudar para el desarrollo de la presente tesis.

A mi tutora Lic Brigida Carvajal Blanco, por su asesoramiento brindándome consejos, apoyo y conocimiento.

RESUMEN

La transcripción musical es un proceso que fue evolucionando con el pasar de los tiempos. Con el creciente avance de la tecnología se tienen nuevos métodos para la transcripción automática de música. El Análisis armónico es una rama de las matemáticas que se ocupa de las representaciones de funciones o señales, así como el estudio y generalización del análisis de Fourier, hoy en día es un área que aún se sigue investigando.

El presente documento propone un algoritmo que tiene como objetivo crear la partitura de una pieza musical en formato WAV aplicando el análisis armónico mediante el análisis de Fourier y el procesamiento de señales musicales.

Se aplica la metodología arriba-abajo enfocada a la programación modular en sus 6 etapas para la creación de un prototipo eficiente implementado en el lenguaje de programación Python con PySide2.

Palabras clave: Análisis armónico, Señales musicales, Espectro de frecuencias, Transformada de Fourier, Partitura musical, Arriba-abajo con enfoque modular.

ABSTRACT

The musical transcription is a process that evolved with the passing of time. With the increasing advancement of technology, there are new methods for automatic music transcription. Harmonic Analysis is a branch of mathematics that deals with the representations of functions or signals, as well as the study and generalization of Fourier analysis, today it is an area that is still being investigated.

This document proposes an algorithm that aims to create the score of a musical piece in WAV format by applying harmonic analysis through Fourier analysis and musical signal processing.

The top-down methodology focused on modular programming in its 6 stages is applied for the creation of an efficient prototype implemented in the Python programming language with PySide2.

Keywords: Harmonic analysis, Musical signals, Frequency spectrum, Fourier transform, Musical score, Top-bottom with modular approach.

ÍNDICE GENERAL

DEDICATORIA	III
AGRADECIMIENTOS	IV
RESUMEN	V
ABSTRACT.....	VI
ÍNDICE DE FIGURAS.....	X
ÍNDICE DE TABLAS	XII
1 MARCO REFERENCIAL	1
1.1. INTRODUCCIÓN.....	1
1.2. ANTECEDENTES.....	3
1.2.1. ESTADO DEL ARTE	3
1.2.1.1. KEATON MUSIC TYPEWRITER	3
1.2.1.2. OPTICAL MUSIC RECOGNITION	4
1.2.1.3. DOREMIR SCORECLOUD	4
1.2.2. TRABAJOS SIMILARES	5
1.3. PLANTEAMIENTO DEL PROBLEMA	7
1.3.1. PROBLEMA CENTRAL	9
1.3.2. PROBLEMAS SECUNDARIOS	9
1.4. DEFINICIÓN DE OBJETIVOS	10
1.4.1. OBJETIVO GENERAL	10
1.4.2. OBJETIVOS ESPECÍFICOS	10
1.5. HIPÓTESIS.....	11
1.5.1. OPERACIONALIZACIÓN DE VARIABLES	11
1.6. JUSTIFICACIÓN.....	11
1.6.1. JUSTIFICACIÓN ECONÓMICA	11
1.6.2. JUSTIFICACIÓN SOCIAL.....	11
1.6.3. JUSTIFICACIÓN CIENTÍFICA	12
1.7.1. ALCANCES.....	12
1.7.2. LÍMITES	13
1.8. APORTES	13
1.8.1. PRÁCTICO	13

1.8.2.	TEÓRICO.....	14
1.9.	METODOLOGÍA	14
2	MARCO TEÓRICO	16
2.1.	LA MÚSICA.....	16
2.2.	TEORÍA MUSICAL	16
2.2.1.	TONO Y NOTAS MUSICALES	16
2.2.2.	RITMO	19
2.2.3.	ESCALAS MUSICALES.....	20
2.2.4.	ACORDES MUSICALES, MELODÍA Y ARMONÍA	21
2.3.	TRANSCRIPCIÓN MUSICAL.....	22
2.3.1.	TRANSCRIPCIÓN TRADICIONAL DE MÚSICA	23
2.3.1.1.	OBJETIVOS DE LA TRANSCRIPCIÓN MUSICAL	24
2.3.1.2.	PROCEDIMIENTO	25
2.3.2.	TRANSCRIPCIÓN AUTOMÁTICA DE MÚSICA	26
2.4.	ANÁLISIS DE FRECUENCIA.....	28
2.4.1.	FRECUENCIA FUNDAMENTAL Y TONO	30
2.5.	SEGMENTACIÓN TEMPORAL	31
2.5.1.	FUNCIÓN DE DETECCIÓN DE INICIO	32
2.5.1.1.	CONTENIDO DE ALTA FRECUENCIA (HFC)	32
2.6.	SEÑALES DE AUDIO	32
2.7.	ANÁLISIS ARMÓNICO	36
2.7.1.	TRANSFORMADA DE FOURIER.....	36
2.7.2.	TRANSFORMADA DISCRETA DE FOURIER.....	38
2.7.3.	TRANSFORMADA RÁPIDA DE FOURIER	40
2.7.4.	TRANSFORMADA DE FOURIER DE TIEMPO REDUCIDO	42
2.8.	AMPLITUDES ARMÓNICAS	45
2.8.1.	BLANQUEAMIENTO ESPECTRAL.....	45
2.8.2.	FUNCION DE PROMINENCIA.....	46
2.9.	PRECISIÓN Y EXHAUSTIVIDAD	47
2.10.	NOTACIÓN BIG O	47
2.11.	KIT DE HERRAMIENTAS MULTIPLAFORMA	49

3 MARCO APLICATIVO	50
3.1. INTRODUCCIÓN.....	50
3.2. DEFINICIÓN DEL PROBLEMA	51
3.3. ANÁLISIS DEL PROBLEMA.....	52
3.4. ANÁLISIS DE DATOS DE ENTRADA Y SALIDA	52
3.4.1. DATOS DE ENTRADA	52
3.4.2. DATOS DE SALIDA.....	53
3.5. DISEÑO Y ANÁLISIS DEL ALGORITMO	53
3.5.1. EXTRACCIÓN DE DATOS	53
3.5.2. PREPARACIÓN DE DATOS	54
3.5.2.1. TAMAÑO DE VENTANA	54
3.5.2.2. FUNCIÓN VENTANA	54
3.5.2.3. RELLENO CERO	55
3.5.2.4. BLANQUEAMIENTO ESPECTRAL.....	56
3.5.3. ANÁLISIS DE FOURIER.....	57
3.5.4. COMPUTO DE FRECUENCIAS FUNDAMENTALES	61
3.5.5. FUNCIÓN DE DETECCIÓN DE INICIO	62
3.5.6. GENERACIÓN DE LA PARTITURA MUSICAL	65
3.5.7. IMPLEMENTACIÓN DE ARCHIVO MIDI	66
3.6. IMPLEMENTACIÓN	67
 4 RESULTADOS Y ANÁLISIS	 70
4.1. RESULTADOS	70
4.1.1. DETECCIÓN DE INICIO DE NOTA MUSICAL	71
4.1.2. EVALUACIÓN DE TRANSCRIPCIÓN	74
4.2. PRUEBA DE LA HIPÓTESIS	76
4.3. ANÁLISIS A LA COMPLEJIDAD ALGORITMICA	77
 5 CONCLUSIONES Y RECOMENDACIONES	 79
5.1. CONCLUSIONES.....	79
5.2. RECOMENDACIONES	81
 6 BIBLIOGRAFÍA	 82

ÍNDICE DE FIGURAS

Figura 1. Maquina Keaton Music Typewriter	3
Figura 2. Arquitectura de reconocimiento de música óptica por Bainbridge y Bell.....	4
Figura 3. Interfaz gráfica del software ScoreCloud	5
Figura 4. Las dos primeras unidades de compas de Fantasía No. 3 en B menor de Telemann representado como una partitura musical	7
Figura 5. Características de un compás musical.	8
Figura 6: Arquitectura del algoritmo para generar la partitura de una pieza musical.....	13
Figura 7: Esquema Programación por módulos.....	14
Figura 8: Dos octavas y media F3-C6 Representadas como (a) notas, (b) teclas de piano	17
Figura 9: Frecuencias	18
Figura 10: Ejemplo de compas musical.....	20
Figura 11: Escala Mayor C.....	20
Figura 12: Escala Menor C.....	21
Figura 13: Ejemplo de acordes musicales. C es C mayor, Cm es C menor, G7 es G-siete y Asus4 es A suspendida 4.....	21
Figura 14: Un extracto de "Polonesa en sol menor" de J. S. Bach.....	22
Figura 15: 2048 muestras (46 ms) de una nota de violín grabada a 44,1 kHz.....	26
Figura 16: Los dos primeros compases de Telemann's Fantasia No. 3 in B minor representados como (a) una partitura musical y (b) un redoble de piano.	27
Figura 17: Clases de tono según lo dispuesto en un teclado.	29
Figura 18. Definición de Onset	31
Figura 19: Sinusoide que oscila a 3 Hz con una fase de $\pi/2$ y amplitud unitaria. La señal se muestrea a una frecuencia de $f_s = 6$ Hz.	34
Figura 20: Sinusoide oscilante a 3 Hz con una fase de $\pi/2$ y amplitud unitaria. La señal se muestrea a una frecuencia de $f_s = 4$ Hz.	34

Figura 21: Forma de onda de una guitarra acústica que toca la nota G3, junto con un primer plano de la señal de 25 ms a partir de 1 segundo.....	35
Figura 22: Forma de onda de un teclado electrónico que toca la nota G3, junto con un primer plano de la señal de 25 ms a partir de 1 segundo.....	36
Figura 23: Señales de dominio de tiempo (izquierda) y sus correspondientes transformadas de Fourier (derecha).....	37
Figura 24: Espectro de magnitud de una guitarra acústica tocando la nota G3. Hay una energía fuerte en $h = 1$, que corresponde a 196 Hz, menos en $h = 2$, incluso menos en $h = 3$, y una energía pequeña pero notable entre los otros armónicos.	39
Figura 25: Espectro de magnitud de un teclado electrónico que toca la nota G3. Hay una energía significativa pero variable en casi todos los armónicos dentro de lo que se muestra en el DFT, que contiene los primeros diez armónicos.....	40
Figura 26: Una comparación de las funciones de ventanas para un marco de audio de 4096 muestras. La ventana rectangular.	43
Figura 27: Espectrograma de la grabación de un solo de piano. El STFT se genera con tamaño $M = 2048$ y tamaño de salto $R = 512$	45
Figura 28. Comparación de órdenes de complejidad	49
Figura 29. Estructura del proyecto dividido por módulos.....	51
Figura 30. Espectro de frecuencias igualmente espaciadas de la nota A4 de un violín.	52
Figura 31. Comparación de las funciones de ventanas para un marco de audio de 4096 muestras. ...	55
Figura 32. Una trama de 1024 muestras de audio procesada mediante la transformada rápida de Fourier.....	56
Figura 33. Algoritmo: blanqueamiento Espectral	57
Figura 34. Algoritmo: Transformada Discreta de Fourier	58
Figura 35. Algoritmo: Transformada Rápida de Fourier	59
Figura 36. Algoritmo: Transformada de Fourier de Tiempo Reducido	60
Figura 37. Algoritmo: Obtención de la frecuencia fundamental	61
Figura 38. Algoritmo: Detección de Onset.....	62
Figura 39. Pieza musical Perpetual Motion, 5 primeros segundos.	62

Figura 40. Piano roll de los 5 primeros segundos de la pieza musical Perpetual Motion.....	63
Figura 41. Selección de inicio de frecuencia. Primera estimación de Onset	63
Figura 42. Selección de picos de frecuencia. Segunda estimación de Onset	64
Figura 43. Equivalencia Nota Musical - Frecuencia - Nota Midi	65
Figura 44. Diagrama de flujo: Generación de la partitura Musical	66
Figura 45. Prototipo pantalla de operaciones.....	67
Figura 46. Prototipo carga de archivo wav	68
Figura 47. Prototipo, ejecución del botón convertir partitura	68
Figura 48. Prototipo, Resultados	69

ÍNDICE DE TABLAS

Tabla 1. Formato de obtención de datos archivos WAV	54
Tabla 2. Instrumentos para el análisis	71
Tabla 3. Métricas: Recuperación de inicio de nota musical, Violín.....	72
Tabla 4. Métricas: Recuperación de inicio de nota musical, Piano.....	73
Tabla 5. Métricas: Recuperación de inicio de nota musical, Clarinete.	73
Tabla 6. Media de precisión obtenida del inicio de nota musical por instrumento	74
Tabla 7. Métricas: Recuperación de transcripción musical, Violín.....	74
Tabla 8. Métricas: Recuperación de transcripción musical, Piano.....	75
Tabla 9. Métricas: Recuperación de transcripción musical, Clarinete.	76
Tabla 10. Media de precisión obtenida de la transcripción musical por instrumento.....	76
Tabla 11. Tiempos de ejecución de cada módulo	78
Tabla 12. Complejidad Algoritmica de cada módulo	78

CAPÍTULO 1. MARCO REFERENCIAL

1.1. INTRODUCCIÓN

La música es el arte relacionado con la combinación de sonidos vocales o instrumentales para la belleza de la forma o la expresión emocional, generalmente de acuerdo con los estándares culturales de ritmo, melodía y en la mayoría de la música occidental, armonía. Tanto la canción popular simple como la compleja composición electrónica pertenecen a la misma actividad, la música. Ambos están diseñados por humanos; ambos son conceptuales y auditivos, y estos factores han estado presentes en la música de todos los estilos y en todos los períodos de la historia, en todo el mundo. (Epperson, 2019).

Representar una pieza musical gráficamente es necesario para que un intérprete la pueda ejecutar de manera deseada, el sistema de notación más utilizado para esta representación es el sistema gráfico occidental que representa sobre un pentagrama figuras musicales.

La Notación musical, es un registro visual de sonido musical escuchado o imaginado, o un conjunto de instrucciones visuales para la ejecución de la música. Por lo general, toma forma escrita o impresa y es un proceso consciente, relativamente laborioso. Una partitura, tiene todas las instrucciones necesarias para que un músico sea capaz de reproducir el sonido. Donde se incluyen valores de las notas, signatura del compás, la clave que hace referencia a la escala musical en la que se escribe la música. (OKDIARIO, 2018)

La interpretación musical consiste en que un músico especializado decodifica un texto musical de una partitura y lo hace audible en uno o varios instrumentos musicales. (Orlandini, 2012)

Estas interpretaciones musicales generan lo que se llama señales musicales, estas señales contienen de manera audible toda la partitura de una pieza musical.

La noción de señal es bastante amplia y aparece en diferentes situaciones en las cuales ciertas cantidades de una magnitud física o de otra naturaleza varían en el tiempo o en el espacio.

Por lo tanto está ligada al concepto de función. Muchas veces, es muy importante expresar estas señales de tal manera de poder analizarlas y manipularlas para desarrollar sistemas dinámicos que requieren distintas áreas como lo son la de: comunicaciones, ingeniería mecánica y de control, de procesamiento de imágenes, de medicina, entre otras. Un claro ejemplo de estas señales es el sonido, el cual es muy utilizado en todo tipo de artefactos electrónicos, y para ello requiere previamente de un gran desarrollo de técnicas para su manipulación, donde la transformada de Fourier cumple un rol muy importante. (Tomás, 2015)

Todo amante de la "buena" música se ha visto expuesto a alguna variedad de música electrónica, concreta o de computadora. Durante los últimos años, investigadores musicales de los dos Cambridges (el de Inglaterra y el de Massachusetts) y de París, han desarrollado una manera radicalmente nueva de analizar y sintetizar sonidos musicales. El elemento esencial es una computadora, pero no para hacer la chamba del compositor, sino para crear los sonidos que él todavía tendrá que "componer". (Fourier y la Música, 2012)

El modo tradicional de considerar un sonido musical lo caracteriza por su intensidad, su tono o frecuencia dominante y su timbre. Ésta es una descripción demasiado simple y primitiva, en donde todas las complicaciones se barren debajo del significado de timbre. Describir adecuadamente la riqueza de la música con tan pobres elementos es equivalente a tener críticos de pintura que fuesen daltónicos. Para superar la clásica terna intensidad-tono-timbre, que se ha incrustado en el alma de la música occidental, se usa una técnica matemática, desarrollada hace más de siglo y medio por José Fourier y publicada en su libro *Théorie analytique de la chaleur*. (Fourier y la Música, 2012).

El análisis de Fourier de un sonido lo descompone en todas y cada una de las frecuencias que lo forman, y le asigna a cada frecuencia una intensidad o amplitud específica. Al conjunto de frecuencias amplitudes se le llama el espectro del sonido analizado. (Fourier y la Música, 2012).

El análisis armónico es una extensión del análisis de Fourier que se centra en la representación de señales mediante ondas básicas, estas señales se procesan a través de los métodos de la transformada de Fourier.

El presente trabajo se propone un algoritmo basado en el análisis armónico para generar la partitura de una pieza musical de una pieza musical en formato wav.

1.2. ANTECEDENTES

1.2.1. ESTADO DEL ARTE

1.2.1.1. KEATON MUSIC TYPEWRITER

Robert H. Keaton, nacido en San Francisco, California, inventó una máquina de 33 teclas la cual puede imprimir caracteres en el pentagrama con mucha precisión, tal como lo explica Keaton, (1953) “Un teclado está adaptado para escribir una clase de caracteres musicales como las barras de compás y las líneas adicionales que, cuando se repiten, suelen aparecer en las mismas posiciones relativas en relación al pentagrama... y un segundo teclado adaptado para teclear otro tipo de caracteres musicales, como las notas, los silencios, los símbolos de sostenido y bemol, etc. que, al repetirse, pueden aparecer en varias posiciones espaciadas con respecto al pentagrama”

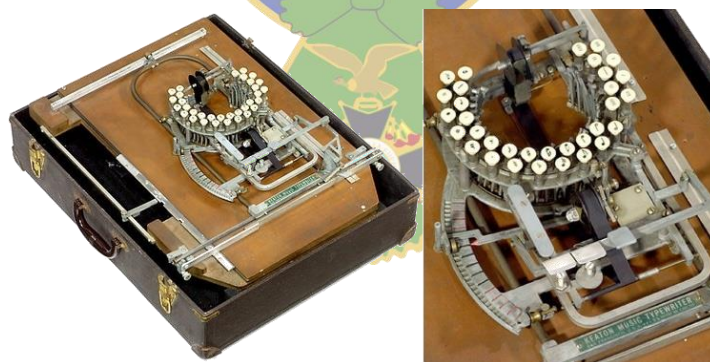


Figura 1. Máquina Keaton Music Typewriter
Fuente: (Keaton, 1953)

1.2.1.2. OPTICAL MUSIC RECOGNITION

En el año 2003, David Bainbridge y Tim Bell publicarían su trabajo sobre los desafíos de OMR, el cual es un sistema que se utiliza para convertir la música escaneada del papel a un formato adecuado para reproducir o editar en una computadora. (Bainbridge y Bell, 2003)

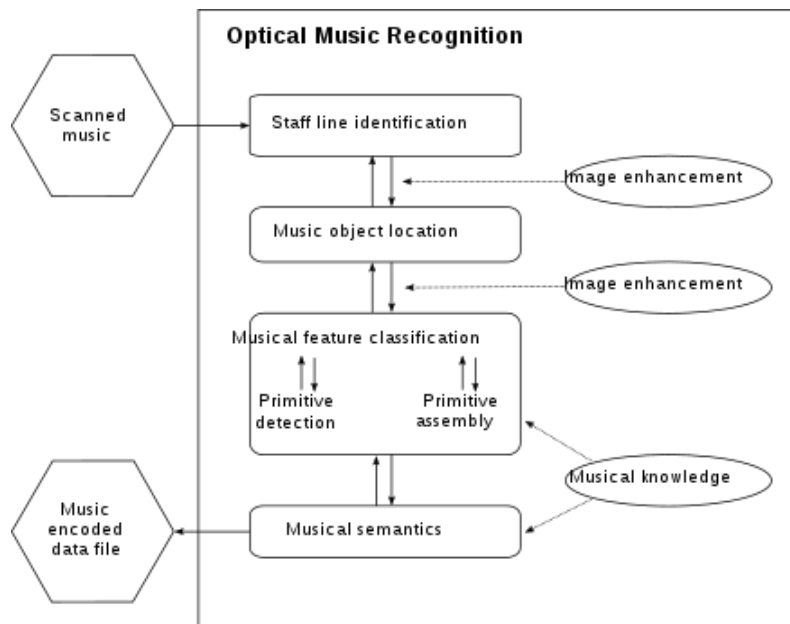


Figura 2. Arquitectura de reconocimiento de música óptica por Bainbridge y Bell
Fuente: (Bainbridge y Bell, 2003)

1.2.1.3. DOREMIR SCORECLOUD

ScoreCloud es un software de notación musical que transcribe cualquier grabación de música polifónica o monofónica de instrumentos en vivo a partitura.

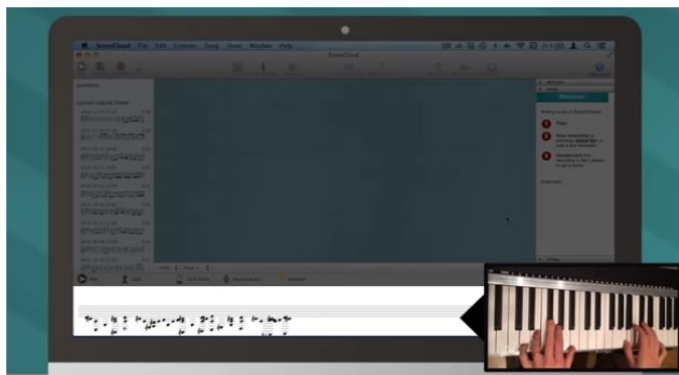


Figura 3. Interfaz gráfica del software ScoreCloud
Fuente: (DOREMIR, 2019)

1.2.2. TRABAJOS SIMILARES

Título: Harmonic analysis for music transcription and characterization

Autor: Alessio Degani

Año: 2014

Institución: *Universita Degli Studi Di Brescia*

Este trabajo plantea diferentes formas de análisis a la música monofónica y polifónica, empieza con el análisis Tiempo-Frecuencia, donde habla de la transformada en tiempo corto de Fourier, plantea una Función de prominencia de tono polifónico, una detección de límite de acordes y pruebas en 2 clases de música. (Degani, 2014)

Título: Automatic music transcription

Autor: Dylan Quenneville

Año: 2018

Institución: Middlebury College

Este trabajo plantea una transcripción musical automática a piezas musicales de tono puro aplicando la transformada de Fourier en tiempo corto utilizando un espectro de audio basado en estimaciones de tono y nociones de armonicidad. (Quenneville. 2018)

Título: End-to-end Music Transcription Using Fine-Tuned Variable-Q Filterbanks

Autor: Frank C. Cwitkowitz Jr

Año: 2019

Institución: Kate Gleason College of Engineering

Esta tesis propone utilizar la transformada Q variable como reemplazo de una arquitectura de transcripción de línea base, este reemplazo aprendido se ajusta conjuntamente con una arquitectura de línea de base para la tarea de transcripción del piano, y los “bancos de filtro” resultantes se visualizan y evalúan contra la transformación estándar. (Cwitkowitz, 2019)

Título: Automatic music transcription software based on constant Q transform

Autor: Robert Alexandru Dobre y Cristian Negrescu

Año: 2016

Institución: Conferencia Internacional sobre Electrónica, Computación e Inteligencia Artificial (ECAI)

Presenta un software de transcripción de música automática que utiliza una transformación Q constante como herramienta de análisis de frecuencia de tiempo y demuestra el por qué esta transformación es más adecuada que la transformada discreta de Fourier para la aplicación presentada. El software puede transcribir estructuras melódicas tocadas en cualquier tecla musical por cualquier instrumento templado. (Dobre y Negrescu, 2016)

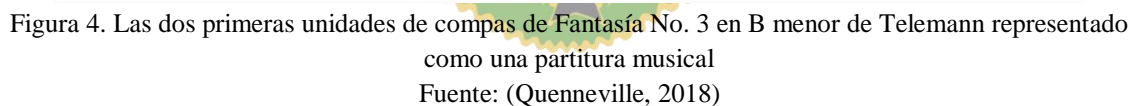
Título: MOMOS-MT: Mobile Monophonic System for Music Transcription.

Autor: Makhmutov, Munir and Brown, Joseph and Mazzara, Manuel and Johard, Leonard

Institución: arXiv – Computer Science

1.3. PLANTEAMIENTO DEL PROBLEMA

En una partitura musical se encuentran distintos elementos que lo componen, por ejemplo las notas musicales, silencios, alteraciones, entre otros.



7

pentagrama de la obra, el numerador indica el número de tiempos que consta el compás y el denominador el valor de cada uno de estos tiempos, seguidamente la tonalidad dicta a que ritmo (o antiguamente una palabra italiana) se tocará la pieza musical. Esta puede tener una generalización de alteraciones llamada armadura.

Las escalas mayores y menores naturales construidas sobre Do y La respectivamente, no tienen ninguna nota alterada. Para poder construir estas escalas partiendo de cualquier otra nota necesitamos alterar una o más notas. Por ejemplo, en la escala de Sol mayor necesitamos alterar con un sostenido la nota Fa. Si deseáramos escribir una melodía en Sol mayor, deberíamos alterar todas las notas Fa. Para evitar tener que escribir tantas alteraciones usamos las armaduras de clave. (Rodríguez, 2016)



Figura 5. Características de un compás musical.
Fuente: (Rodríguez, 2016)

Entendemos por acorde a un conjunto de tres o más notas musicales diferentes (García, 2018) seguidamente del tempo que es la velocidad en la cual se interpreta la pieza musical. (Rivera, 2020)

En la interpretación, los músicos convierten las representaciones de partituras en sonido que se transmite a través del aire como oscilaciones de presión de aire. En esencia, el sonido es simplemente una vibración de aire. Una señal musical es una representación del sonido que representa la fluctuación en la presión del aire causada por la vibración en función del tiempo. A diferencia de las partituras o las representaciones simbólicas, las representaciones de audio codifican todo lo necesario para reproducir una realización acústica de una pieza musical. (MIR, 2018).

El sonido se compone de distintas ondas que viajan de forma simultánea, aunque las personas las escuchen como una sola. El timbre es una de sus cualidades y permite diferenciar sonidos que comparten altura, sonoridad y duración. Además, depende de la cantidad de armónicos que posee una fuente sonora y su intensidad. Las ondas que emite un cuerpo poseen varias frecuencias. Existe una principal y otras secundarias que contribuyen a marcar la personalidad del conjunto.

La transcripción de música se define como el proceso de analizar una señal musical para escribir los parámetros de los sonidos que constituyen a la pieza musical en cuestión. Dicho de otro modo, consiste en obtener una representación simbólica de la pieza que contenga todos los aspectos musicales de la misma, es decir, además de la identificación de la nota, determinar el tono, el ritmo, y la duración de la misma. (Klapuri, 2004)

Este proceso puede considerarse como la operación inversa de interpretación musical, que a menudo involucra a un intérprete que lee notación.

Actualmente, la transcripción es una habilidad que solo poseen expertos con amplios conocimientos musicales o músicos experimentados. Aun así, es un proceso costoso e ineficiente que es propenso a errores humanos y no escala bien. (Cwitkowitz, 2019)

1.3.1. PROBLEMA CENTRAL

¿Cómo generar la partitura de una pieza musical en base a señales musicales?

1.3.2. PROBLEMAS SECUNDARIOS

- Existen piezas musicales cuya partitura original no se encuentra al alcance de todos, optar por transcribir de manera clásica (a oído) es un proceso que puede generar errores y consumir tiempo.
- Los armónicos de una pieza musical alteran de manera significativa los picos de frecuencia y esto conlleva a generar una incorrecta aproximación de la nota musical.

- El ritmo y tempo influyen en las características del compás y la velocidad a la que se interpreta la pieza musical, esto ocasiona que la interpretación de las figuras musicales sea distinta.
- Las señales musicales varían en rango de frecuencias, esto influye en el análisis del espectro de frecuencias y puede ocasionar una generación inexacta de figuras musicales.
- La tonalidad de una pieza musical afecta a la entonación de los sonidos producidos por la pieza musical lo que ocasiona una inexacta detección de accidentes o alteraciones musicales.

1.4. DEFINICIÓN DE OBJETIVOS

1.4.1. OBJETIVO GENERAL

Diseñar un algoritmo basado en el análisis armónico aplicado a señales musicales para obtener un espectro de frecuencias del cual se podrá generar la partitura de la pieza musical.

1.4.2. OBJETIVOS ESPECÍFICOS

- Reducir la complejidad de la implementación algorítmica para obtener la partitura de una pieza musical en tiempos óptimos.
- Realizar un módulo de pre procesamiento al espectro de frecuencias para la detección de frecuencias fundamentales.
- Diseñar un módulo basado en las frecuencias obtenidas que detecte el tempo de una pieza musical.
- Generalizar el rango de frecuencias de la pieza musical mediante índices de Fourier.
- Realizar un módulo de post procesamiento para detectar la tonalidad en la cual se está interpretando la pieza musical.

1.5. HIPÓTESIS

El algoritmo basado en el análisis armónico aplicado en señales musicales obtiene un espectro de frecuencias el cual genera la partitura de la pieza musical con una precisión mayor al 85%.

1.5.1. OPERACIONALIZACIÓN DE VARIABLES

Variable	Tipo
Algoritmo basado en el análisis armónico	Independiente
Espectro de frecuencias el cual genera la partitura de la pieza musical	Dependiente
Señales musicales	Interviniente

1.6. JUSTIFICACIÓN

1.6.1. JUSTIFICACIÓN ECONÓMICA

El algoritmo para generar la partitura de una pieza musical será de gran ayuda al reducir los costos que se genera al contratar un interpretador musical mediante un prototipo de fácil uso.

El algoritmo generará ahorros a los interpretadores musicales ya que podrán partir de una partitura generada como base para seguir transcribiendo la pieza musical deseada.

1.6.2. JUSTIFICACIÓN SOCIAL

El algoritmo de generación de partituras será de gran ayuda en:

La parte música y procesamiento de señales puesto que ayudará en gran medida a un interpretador musical que aplica técnicas tradicionales para generar la partitura de una pieza musical.

Para los músicos que estén afinando su oído para aprender la transcripción musical de modo clásico.

Personas o agencias que se dediquen a crear covers de músicas ya compuestas las cuales necesitan su partitura.

Estudiantes que estén aprendiendo a tocar un instrumento monofónico y quieran interpretar una pieza musical cuya partitura no esté disponible.

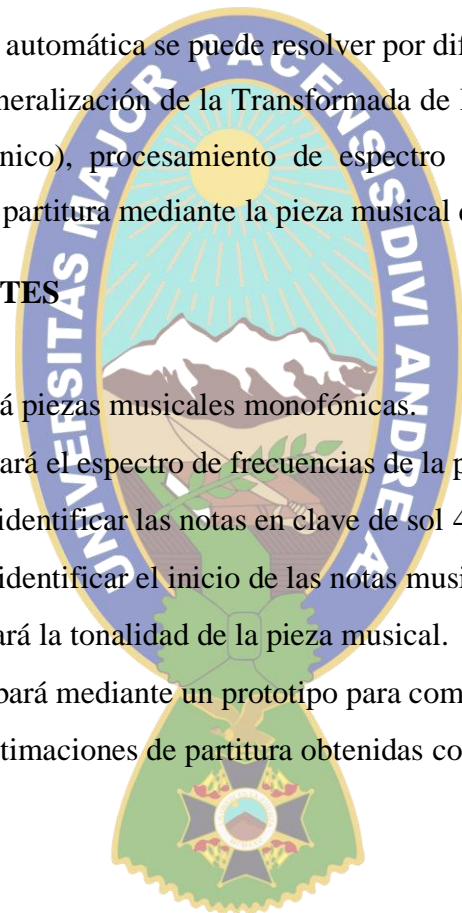
1.6.3. JUSTIFICACIÓN CIENTÍFICA

La generación de partituras automática se puede resolver por diferentes métodos, la presente investigación emplea la generalización de la Transformada de Fourier aplicado a las piezas musicales (Análisis Armónico), procesamiento de espectro de frecuencias para que el algoritmo pueda generar la partitura mediante la pieza musical que se analice.

1.7. ALCANCES Y LÍMITES

1.7.1. ALCANCES

- El algoritmo recibirá piezas musicales monofónicas.
- El algoritmo procesará el espectro de frecuencias de la pieza musical.
- El algoritmo podrá identificar las notas en clave de sol 4/4.
- El algoritmo podrá identificar el inicio de las notas musical.
- El algoritmo detectará la tonalidad de la pieza musical.
- El algoritmo se probará mediante un prototipo para comprobar su eficacia.
- Se comparará las estimaciones de partitura obtenidas con la composición original.



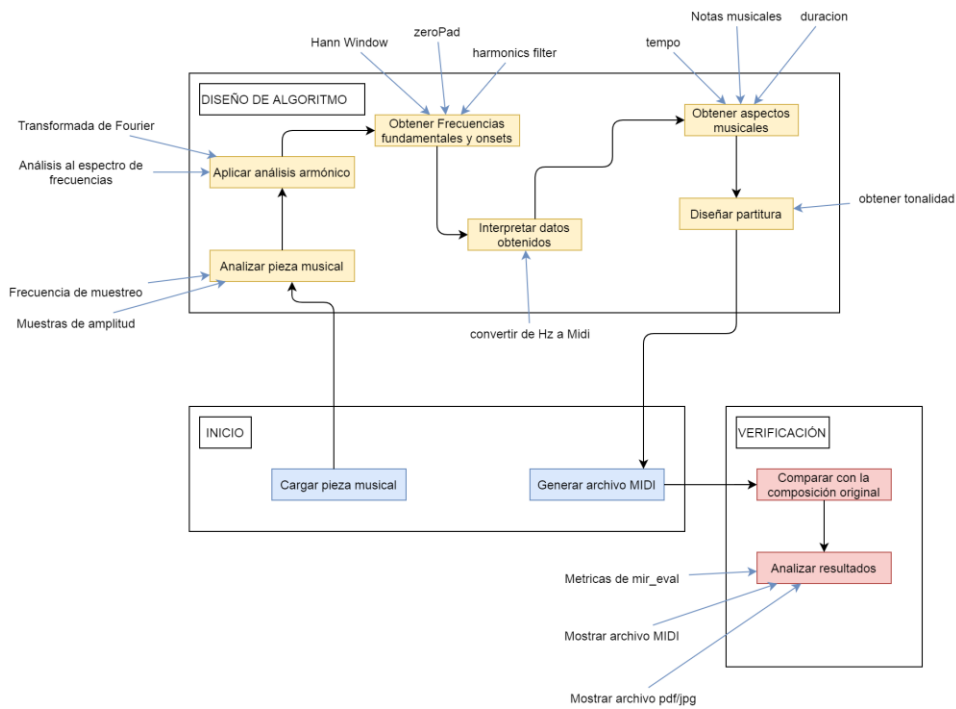


Figura 6: Arquitectura del algoritmo para generar la partitura de una pieza musical.
Fuente: (Elaboración propia)

1.7.2. LÍMITES

- El algoritmo no recibirá piezas musicales que intervengan 2 o más instrumentos
- El algoritmo no procesará piezas musicales cuyo espectro de frecuencias no sea claro.
- El algoritmo no recibirá piezas musicales con múltiples tonalidades.

1.8. APORTES

1.8.1. PRÁCTICO

En el presente trabajo se presentará un algoritmo el cual inicia con la lectura de una pieza musical seguidamente empleará el algoritmo descrito bajo el análisis armónico y devolverá la partitura de la pieza musical en clave de sol.

1.8.2. TEÓRICO

En la presente tesis de investigación se realiza con el propósito de aportar al conocimiento existente sobre el análisis armónico aplicado a las señales musicales de una pieza musical con el fin de que sirva de apoyo para futuras investigaciones en el campo de la música y procesamiento de señales musicales.

1.9. METODOLOGÍA

Se emplea el método científico la cual consiste en la sistematización, formulación y análisis. A su vez tendrá un enfoque cuantitativo la cual consistirá en aplicar modelos matemáticos y técnicos computacionales.

La investigación que se llevará acabo será de tipo, exploratorio descriptivo.

La metodología para el desarrollo del algoritmo será propone la programación modular con enfoque arriba hacia abajo, la cual consiste en dividir el problema principal en sub tareas o módulos

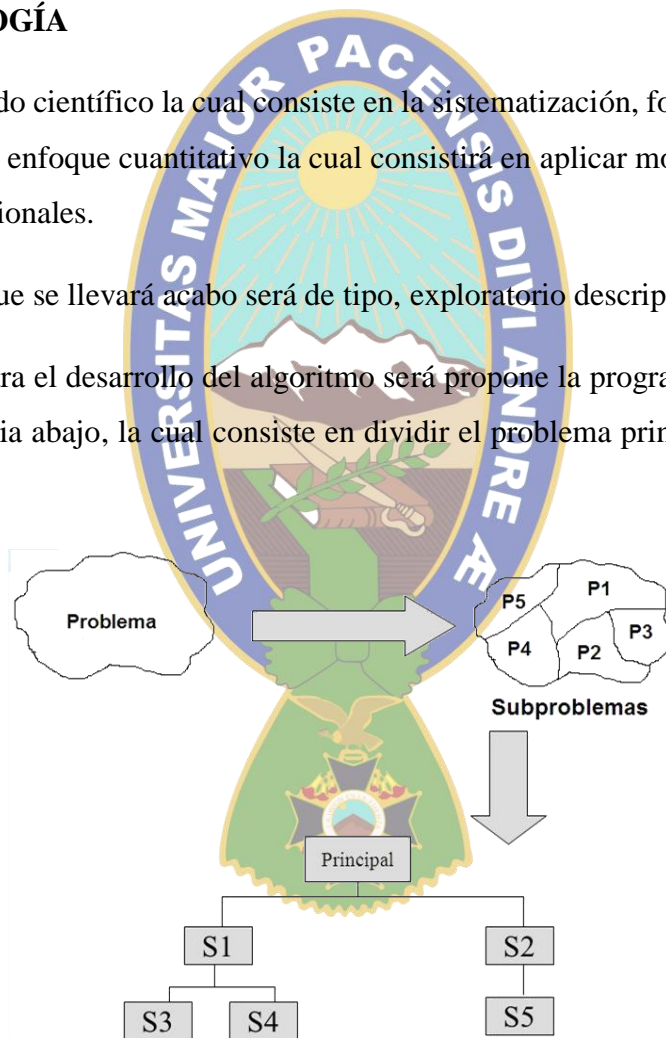


Figura 7: Esquema Programación por módulos.
Fuente: (Elaboración propia)

El proceso de desarrollo sigue las siguientes etapas.

- Introducción.
- Definición del problema.
- Análisis del problema.
- Análisis de datos de entrada y salida.
- Diseño y análisis del algoritmo.
- Implementación.



CAPÍTULO 2. MARCO TEÓRICO

2.1. LA MÚSICA

En su nivel más básico, la música es una serie de tonos agudos que se tocan en un ritmo rítmico particular. Secuencia. Esta secuencia estructurada de sonido de tono generalmente es interpretada por algo que genera vibraciones de ciertas frecuencias, ya sea un instrumento musical que genera vibraciones físicas, cuerdas vocales humanas o un instrumento electrónico que utiliza ya sea componentes eléctricos o electrónica digital para crear formas de onda. Russel Barton define la música como "sonido, estructura e intención artística", y el sonido como "tono, duración, sonoridad, timbre, textura y ubicación espacial". De esta definición, cuatro componentes son particularmente relevantes para la transcripción automática de música. Estos son tono, duración, volumen y timbre. Por el bien de esta tesis, ampliaremos la duración ligeramente a incluir estructura y hablar de ella más ampliamente como ritmo. Estos cuatro componentes fueron elegidos porque finalmente la transcripción de una pieza solo necesita grabar con precisión el tono, ritmo, volumen y timbre, aunque ciertamente la visión artística y estructura a gran escala de la música puede ser determinada o interpretada. (Quenneville, 2018)

2.2. TEORÍA MUSICAL

La teoría de la música es una colección de reglas, definiciones y relaciones que subyacen a la composición musical. (Degani, 2014)

La teoría musical es el estudio de las posibilidades de la música, se ocupa de describir cómo los músicos y compositores hacen música, incluidos los sistemas de afinación y los métodos de composición.

2.2.1. TONO Y NOTAS MUSICALES

El tono es un fenómeno perceptivo que permite percibir lo bajo o lo alto de una tonalidad. Esa propiedad permite organizar las notas musicales en una escala de frecuencia ordenada.

La altura percibida de un tono musical está estrictamente relacionada con la frecuencia fundamental de la nota, pero no es lo mismo. La frecuencia es una propiedad física de un sonido de nota, pero el tono es una sensación auditiva subjetiva inducida por la interacción compleja de cantidades objetivas, como la combinación de frecuencias (contenido armónico) de un tono musical y su amplitud. (Degani, 2014)

Las notas musicales son un conjunto de tonos etiquetados y la diferencia de frecuencia entre dos notas se llama intervalo. El intervalo más importante es la octava que corresponde a duplicar (o reducir a la mitad) la frecuencia. La octava se subdivide en 12 notas (la escala cromática). Cada uno de estos 12 tonos está etiquetado con letras de la A hacia la G y modificadores como \flat (bemol) y \sharp (sostenido) que identifica de forma exclusiva la llamada clase de tono. La octava correspondiente se identifica usando un número. (Degani, 2014)

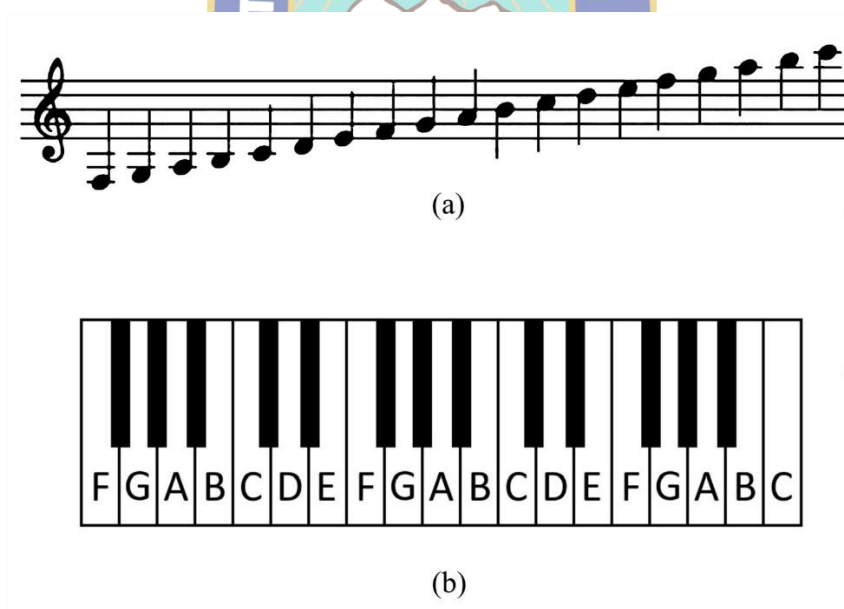


Figura 8: Dos octavas y media F3-C6 Representadas como (a) notas, (b) teclas de piano
Fuente: (Quenneville, 2018)

Por ejemplo, la nota llamada A4 es la nota A en la 4ª octava. Los nombres de las notas asociadas a alguna octava del piano se representan en la Fig. 9.2 (b). El intervalo básico entre notas adyacentes se llama semitono (o medio tono).

Existe una base matemática muy interesante para la octava. Una característica clave del tono musical es que, si bien está directamente relacionado con la frecuencia, no está relacionado linealmente. En cambio, el tono sigue una escala logarítmica. Por definición, una nota una octava por encima de otra nota tendrá el doble de frecuencia. Los tonos musicales modernos se definen según $A4 = 440$ Hz; es decir, tocar un A4 en cualquier instrumento debería causar vibraciones que se repiten 440 veces por segundo. A5, entonces, es 880 Hz, mientras que A3 es 220 Hz. La figura 2.1 muestra dos octavas y media de notas, las teclas de piano correspondientes y sus frecuencias representadas en una escala lineal. (Quenneville, 2018)

Esta relación logarítmica de tonos está profundamente relacionada con la armonía. La armonía se refiere a las relaciones de notas sonadas simultáneamente. Esto se logra mediante un instrumento polifónico, como un piano, que tiene cuerdas dedicadas para cada tono, o mediante múltiples intérpretes que tocan juntos. Debido a las relaciones de las frecuencias de los diferentes tonos, las diferentes combinaciones de tonos tienen diferentes cualidades. Esta colocación de frecuencias a lo largo de una curva logarítmica es importante para comprender. (Quenneville, 2018)

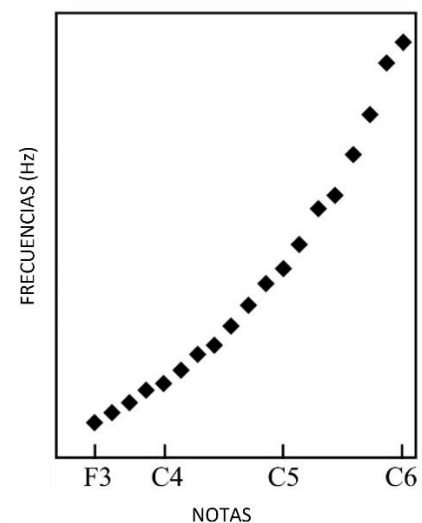


Figura 9: Frecuencias
Fuente: (Quenneville, 2018)

Hay que tener en cuenta las variaciones en el tono también es importante para la transcripción automática de música. La mayoría de los instrumentos requieren cierta cantidad de ajustes menores para tocar perfectamente en sintonía con $A4 = 440$ Hz, pero no hay garantía de que cada tono producido por un intérprete y su instrumento coincidan exactamente con los tonos esperados de cada nota. Además, también hay casos de variación intencional del tono. Vibrato, por ejemplo, se refiere a una variación rítmica de tono hacia arriba y hacia abajo para crear un cierto efecto. Algunos instrumentos también pueden lograr tonos entre los 12 tonos de la octava aplicando tensión adicional a las cuerdas o con un uso cuidadoso del aire. Los tonos intermedios y los cambios graduales en el tono pueden plantear desafíos para la comprensión algorítmica de la música que supone una definición bastante rígida y discreta de los tonos musicales.

2.2.2. RITMO

Ritmo, en la música es la colocación de sonidos en el tiempo. En su sentido más general, el ritmo (ritmos griegos, derivado de *rhein*, "fluir") es una alternancia ordenada de elementos contrastantes. (Crossley, 2002)

El ritmo es la estructura temporal subyacente de la música. En general, la música se compone de notas tocadas en ciertos intervalos de tiempo. La unidad principal de ritmo es el latido, y la velocidad a la que se ejecuta la música generalmente se cuenta en términos de latidos por minuto (lpm). Los latidos generalmente se agrupan en medidas, que pueden tener entre dos y doce latidos de largo, aunque tres o cuatro latidos por medida son los más comunes. (Crossley, 2002)

La medida de tiempo se indica en la apertura de una pieza musical mediante un compás.



Figura 10: Ejemplo de compas musical
Fuente: (Crossley, 2002)

2.2.3. ESCALAS MUSICALES

Las notas se pueden organizar en una variedad de escalas. Una selección de un subconjunto de tonos (generalmente siete, pero son posibles otras combinaciones) del conjunto de 12 tonos de la escala cromática que están dispuestos en patrones de semitonos y tonos completos (dos semitonos) crea las escalas. El primer tono de la escala se llama clave, y los intervalos (llamados grados y denotados en números romanos o usando adjetivos específicos) entre la clave y las otras notas de la escala identifican el modo de la escala y, por lo tanto, la regla de selección de notas. El modo de escala más comúnmente encontrado en la música occidental es la escala mayor y la escala menor representada.

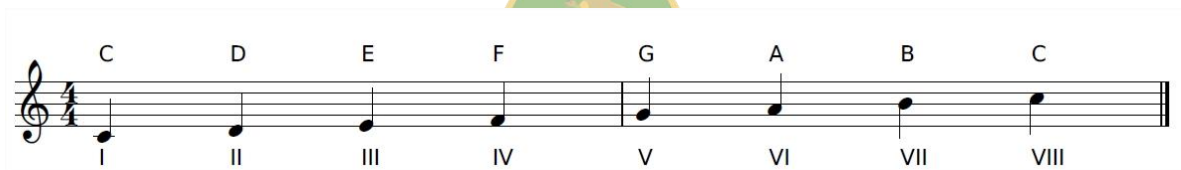


Figura 11: Escala Mayor C
Fuente: (Alessio Degani, 2013)



Figura 12: Escala Menor C
Fuente: (Alessio Degani, 2013)

En la notación occidental tradicional, la escala utilizada para una composición generalmente se indica mediante una firma de clave al principio para designar los tonos que componen esa escala. La música se puede transponer de una escala a otra para diversos fines. La transposición es un cambio positivo o negativo del rango de tono general que cambia la tonalidad, pero conserva las relaciones intervalógicas (el modo) de la escala original.

2.2.4. ACORDES MUSICALES, MELODÍA Y ARMONÍA

Un grupo de 3 o más notas que suenan simultáneamente se llama acorde. Las notas que forman un acorde generalmente (pero no obligatoriamente) se toman del subconjunto de tonos que identifica la escala de la pieza musical. El nombre del acorde depende de los tonos que componen el acorde y, al igual que el nombre de la escala, está formado por un nombre de raíz (el tono principal) y una especificación del tipo de acorde. El tipo de un acorde, al igual que el nombre de una escala, depende de las relaciones intervalógicas entre la nota fundamental y las otras notas del acorde. Los tipos de acordes más utilizados en la música occidental son mayor, menor, séptima, suspendida, aumentada y atenuada. En la Fig 13 se muestran algunos ejemplos.

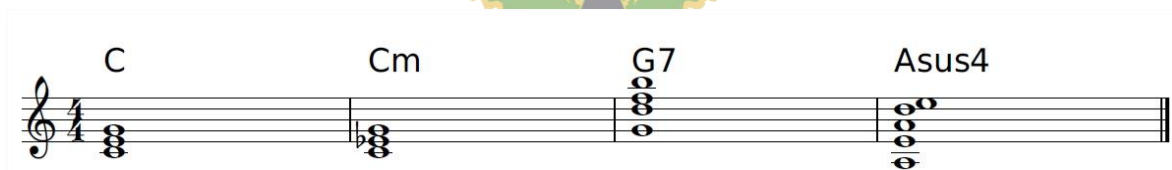


Figura 13: Ejemplo de acordes musicales. C es C mayor, Cm es C menor, G7 es G-siete y Asus4 es A suspendida 4

Fuente: (Alessio Degani, 2013)



Figura 14: Un extracto de "Polonesa en sol menor" de J. S. Bach.

Fuente: (Alessio Degani, 2013)

La melodía es una serie de tonos que suenan en sucesión. Las notas de la melodía se toman generalmente (pero no obligatoriamente) del subconjunto de tonos de la escala de la pieza musical real. Los elementos básicos de la melodía son el tono, la duración de las notas, los patrones de ritmo y el tempo (velocidad, medida en pulsaciones por minuto, BPM). Las relaciones que ocurren simultáneamente entre las notas de la melodía y los acordes se conocen como armonía. La armonía es un punto clave para las tareas MIR como la estimación de acordes o clave, porque explotar la relación armónica que subyace a una pieza musical puede mejorar las actuaciones de precisión de los algoritmos MIR y también el costo computacional. Está claro que el conocimiento de la escala musical que subyace a una composición musical puede reducir el subconjunto de los acordes "válidos" para esa canción determinada.

2.3. TRANSCRIPCIÓN MUSICAL

La transcripción de música se refiere al análisis de una señal musical para escribir el tono, el tiempo de inicio, la duración y la fuente de cada sonido que se produce en él. En la tradición occidental, la música escrita usa símbolos de notas para indicar estos parámetros en una pieza musical. (Klapuri, 2004)

Una pieza musical transcrita es un recurso poderoso. Desafortunadamente, la transcripción puede llevar mucho tiempo y ser propensa a errores, y requiere un cierto nivel de experiencia musical.

2.3.1. TRANSCRIPCIÓN TRADICIONAL DE MÚSICA

Cuando alguien está tocando música, a medida que leen la música, convierten las notas que ven en una posición con los dedos en su instrumento o un tono de su voz y producen el sonido de acuerdo con el ritmo y la dinámica tal como está escrito en la página. La transcripción es esencialmente la inversa de este proceso. El objetivo es producir una representación escrita de una pieza musical con el audio, probablemente una grabación. En general, el propósito de esto es aprender a tocar una pieza musical que no tiene una forma escrita disponible o entenderla musicológicamente.

Desafortunadamente, la transcripción de música puede resultar difícil y llevar mucho tiempo, especialmente cuando la grabación tiene muchos tonos superpuestos (Masatak, 2006).

La dificultad de esta tarea puede entenderse en comparación con la facilidad con la que los humanos pueden leer pasajes de texto y la relativa dificultad de escribir lo que alguien está diciendo. Para aumentar la complejidad del problema, los humanos a menudo procesan el tono relativamente, en lugar de absolutamente. (Quenneville, 2018)

Algunos humanos tienen lo que se conoce como tono perfecto, la capacidad de reconocer tonos de forma aislada. Para aquellos que no tienen un tono perfecto, un método de adivinar y verificar es la mejor opción, pero esto puede llevar mucho tiempo, ya que el transcriptor puede tener que escuchar un pasaje docenas de veces para obtener los tonos y el ritmo completamente precisos.

A pesar de la dificultad y el tedio de la transcripción de música, es útil como herramienta educativa y la pieza musical transcrita resultante puede ser muy valiosa para músicos y musicólogos por igual.

2.3.1.1. OBJETIVOS DE LA TRANSCRIPCIÓN MUSICAL

El objetivo de la transcripción de música es doble. El primer objetivo se relaciona con el proceso de transcripción, mientras que el segundo se relaciona con el producto. (Quenneville, 2018)

Cuando un músico o compositor en formación se sienta a transcribir una pieza musical, a menudo es con el objetivo de comprender mejor el vocabulario y la técnica musical a través del proceso de transcripción. En un nivel básico, la transcripción de música obliga al transcriptor a escuchar cada nota de una grabación y reconocer su tono, tiempo y relación con otras notas a su alrededor. Para un compositor, esto ayuda a construir un vocabulario de patrones de intervalos y elecciones rítmicas que suenan o no atractivos o captan la atención del oyente. También puede haber algún beneficio neurológico para enfocar y procesar melodías; Weinberger (2004) señala que “las células cerebrales "se reasignan" a sí mismas de acuerdo con los tonos "importantes" percibidos. Por lo tanto, la transcripción de música de oído y manual también tiene beneficios para la potencia de procesamiento de música en general. Sin embargo, los beneficios anteriores pueden considerarse más como efectos secundarios.”

La idea de la transcripción musical es más bien producir una forma escrita de una pieza musical no escrita. El análisis de la música en general comienza con la música escrita. Aunque la música es un arte auditivo, está estructurada de acuerdo con patrones de frecuencias y ritmos, todos basados en una cuantificación rígida de tono y ritmo. Como resultado, al buscar la evaluación o la comprensión de una composición, la forma más directa de discutir la composición es en términos de la versión simplificada de la grabación representada en papel

2.3.1.2. PROCEDIMIENTO

En su forma más básica, la transcripción de música tradicional, también llamada transcripción manual, generalmente implica un enfoque de adivinar y verificar para escribir los tonos y ritmos correctos. (Klapuri, 2006)

Antes de intentar escribir los ritmos y las notas que escucha el transcriptor, por lo general, se comienza con la recopilación de meta información sobre la grabación que se va a transcribir. Al escuchar una grabación por primera vez, las personas con experiencia musical suelen captar rápidamente el tempo, el tipo de compás, lo que puede ser útil para dividir el proceso de transcripción en sub tareas medida por medida.

Conocer la firma de la clave puede resultar útil para realizar conjeturas fundamentadas sobre los valores de tono; Si se sabe que una composición está en la clave de D mayor, entonces las notas en una escala de D mayor son mucho más probables que las que están fuera de la escala de D mayor. La firma de clave se puede determinar intentando acompañar la pieza con diferentes notas graves y encontrando cuál encaja mejor. (Quenneville, 2018)

La teoría musical también sugiere que las composiciones a menudo terminan con notas del acorde que forma la firma de clave. También es importante tomar nota de la instrumentación y lo que específicamente el transcriptor espera producir. En el caso de un solo de violín, es probable que haya un grupo de músicos acompañantes que tocan otros instrumentos, aun así el enfoque de la transcripción está solo en el instrumento principal.

Si la transcripción se realiza con objetivos musicológicos en mente, o si se trata de una transcripción de un arreglo completo de una pieza, entonces puede que sea necesario transcribir cada instrumento individual. Una vez que se logra la preparación anterior, el proceso de transcripción generalmente comienza escuchando un breve segmento de la grabación e intentando reproducirlo en un instrumento. A menudo, esto significa sentarse junto a un piano y tocar repetidamente diferentes aproximaciones de la grabación en el piano y decidir qué suena bien.

Muchos músicos pueden identificar los intervalos más fácilmente que los tonos aislados, por lo que a menudo el transcriptor puede escuchar la secuencia de intervalos, encontrar el tono inicial y hacer conjeturas muy bien fundamentadas sobre la secuencia de notas basadas en la firma de clave y los intervalos que tienen. Aun así, esta estrategia puede llevar mucho tiempo. La transcripción exacta de ritmos también requiere repetir bastantes veces la melodía. Hoy en día, la transcripción manual también se puede realizar en una computadora utilizando un software de notación musical, en lugar de hacerlo en papel con un bolígrafo o lápiz. Este software a menudo tiene la opción de reproducir la música que se ha ingresado. Escuchar esto puede ser invaluable para obtener ritmos y tonos precisos. Sin embargo, esta forma de transcripción asistida por computadora probablemente no califica como transcripción automática o semiautomática, ya que la parte de la transcripción todavía la realiza en su totalidad un músico capacitado, simplemente utilizando la generación de música por computadora para verificar su trabajo.

2.3.2. TRANSCRIPCIÓN AUTOMÁTICA DE MÚSICA

La transcripción automática de música, busca automatizar algorítmicamente el proceso de conversión de música de audio grabada a música escrita. Ha sido un área de investigación durante más de 40 años y es una parte central del procesamiento de señales musicales (Moorer, 1977).

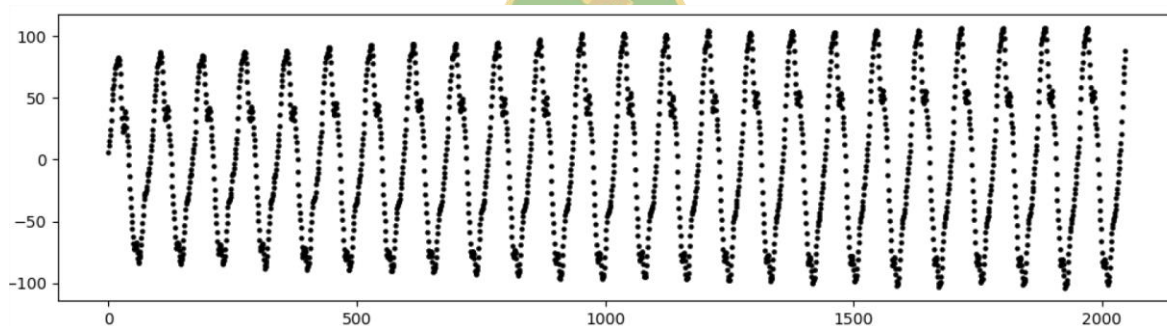


Figura 15: 2048 muestras (46 ms) de una nota de violín grabada a 44,1 kHz.

Fuente: (Quenneville, 2018)

La transcripción automática de música generalmente comienza con el llamado audio con calidad de CD (Goto, 2006). Consiste en música grabada con una frecuencia de muestreo de 44,1 kHz y muestras de 16 bits. La figura 15 muestra 46 ms de una nota de violín muestreada a 44,1 kHz. Las grabaciones pueden ser estereofónicas (estéreo, dos canales) o monoaural (mono, un solo canal) dependiendo de la fuente. Aunque la mayoría de las bases de datos de muestras contemporáneas utilizan una frecuencia de muestreo de 44,1 kHz, algunos sistemas de detección de tono más antiguos usaban frecuencias de muestreo significativamente más bajas o grabaciones de entrada de muestreo descendente artificialmente para mejorar el tiempo de ejecución (Moorer, 1977), pero con las computadoras modernas, este muestreo descendente es innecesario.

La salida de la transcripción automática de música suele ser una representación digital de notas con tono, ritmo y, a veces, volumen específico. Puede ser una notación simplificada, como un redoble de piano, o puede ser una partitura. Un formato de salida común es un archivo MIDI. El estándar MIDI (Musical Instrument Digital Inter-face) tiene la ventaja de una fácil conversión a notación de piano, partituras o interfaz con instrumentos digitales.

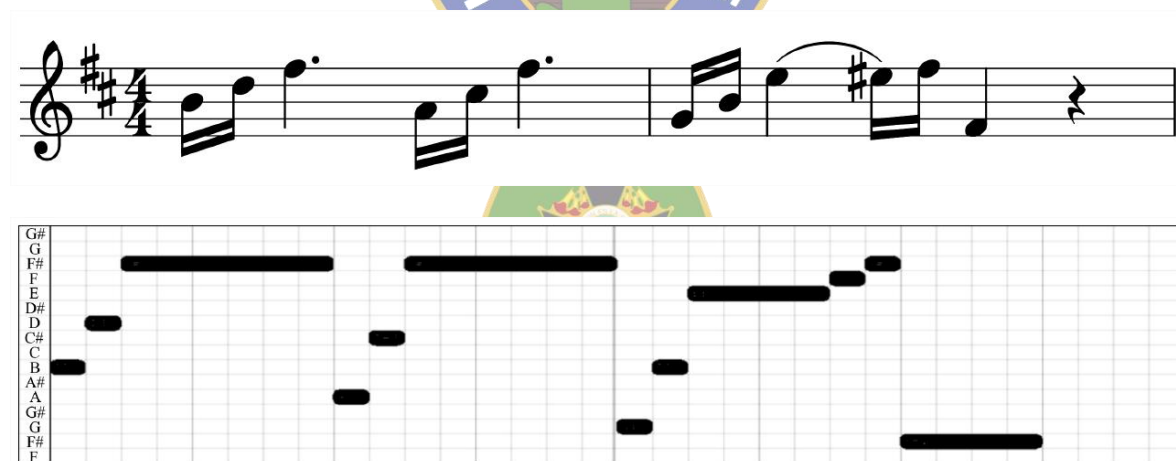


Figura 16: Los dos primeros compases de Telemann's Fantasia No. 3 in B minor representados como (a) una partitura musical y (b) un redoble de piano.

Fuente: (Quenneville, 2018)

La transcripción automática de música se puede dividir en varias sub tareas relacionadas. Las sub tareas más importantes son la estimación del tono y el seguimiento de notas. Los objetivos adicionales de la transcripción automática de música pueden incluir la detección de instrumentos y la identificación de armonías. El análisis de la escena auditiva y la descripción de la escena musical son tareas relacionadas con el análisis musical automatizado, pero generalmente se consideran distintas de la transcripción (Kunio y Goto, 2006).

2.4. ANÁLISIS DE FRECUENCIA

Los instrumentos musicales crean vibraciones periódicas cuando se tocan notas, por ejemplo: punteando una cuerda. Las oscilaciones de presión de aire resultantes se denominan tonos complejos armónicos. La energía de estos tonos se concentra en frecuencias armónicas, múltiplos enteros positivos $h \in \mathbb{Z}^+$ de una frecuencia fundamental f_0 :

$$f_h = hf_0 \quad (2.1)$$

En la ecuación 2.1, $h = 1$ se usaría para indexar la primera frecuencia armónica, también conocida como f_0 fundamental (frecuencia fundamental), de un tono complejo. Cuando se toca una nota, la distribución única de energía entre las frecuencias armónicas influye en una propiedad conocida como timbre, que determina cómo suena la nota. El timbre permite al oyente escuchar y distinguir entre varios instrumentos en una mezcla. El f_0 de un tono armónico complejo es lo que lo caracteriza por tener un cierto tono, la propiedad perceptiva por la cual los humanos pueden clasificar las notas en orden ascendente o descendente según la frecuencia. Aunque los diferentes instrumentos tienen un sonido único, el f_0 de notas idénticas es común en todos ellos.

Es importante destacar que, aunque los términos nota y tono a menudo se usan indistintamente, es mejor pensar que una nota tiene un tono. En la música occidental, hay 12 clases de tonos, indicados por las primeras siete letras del alfabeto y cinco variaciones sostenidas (#) o bemoles (b). Estas clases de tono están ordenadas, comenzando con C, y se

asignan a notas cíclicamente, repitiéndose después de un intervalo conocido como octava. El mapeo entre el tono y la nota es uno a uno, mientras que el mapeo entre la clase de tono y la nota es uno a varios (Cwitkowitz, 2019). La Figura 17 ilustra cómo las clases de tono se corresponden con la disposición estándar de las teclas del piano.

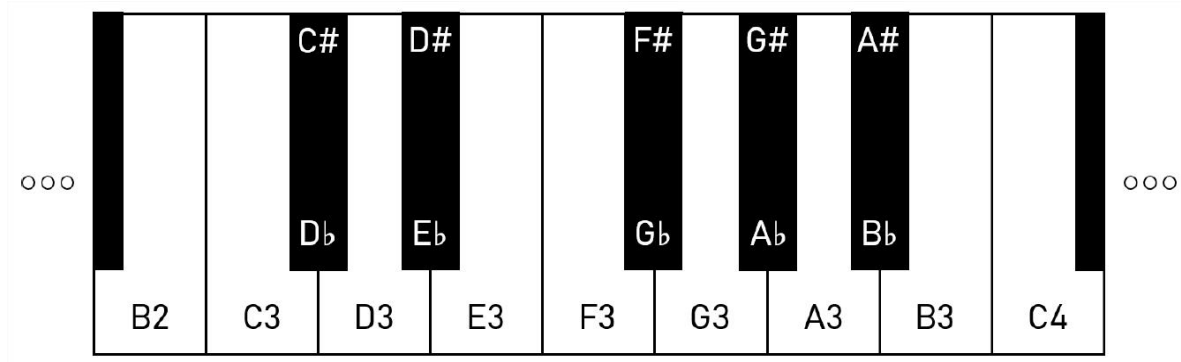


Figura 17: Clases de tono según lo dispuesto en un teclado.
Fuente. (Cwitkowitz, 2019).

En la figura 17 las teclas blancas se asignan a las primeras siete letras del alfabeto, mientras que las teclas negras se asignan a las variaciones nítidas y puntuales. Los números que siguen a cada clase de tono corresponden a la octava que contiene las notas. Un teclado de tamaño completo generalmente admite las notas A0 a C8.

Cada nota de la escala occidental está asociada con una clase de tono y una octava. El intervalo entre clases de tono adyacentes se denomina semitono. Esto significa que hay 12 intervalos de semitonos por octava. El término semitono también se puede utilizar para abstraer el concepto de nota. Como se indicó anteriormente, las clases de tono son cíclicas, es decir, un intervalo de un semitono a partir de B3 conduciría a C4. El f_0 en Hz^2 de cualquier semitono n está determinada por una relación geométrica que se basa en la f_0 del semitono anterior:

$$f_0(n) = 2^{\frac{1}{12}} f_0(n-1) \quad (2.2)$$

Para establecer la relación de (arriba), el f_0 de la nota A4 suele ser fijo. Observe que la diferencia en la frecuencia de los semitonos adyacentes sobre el conjunto completo es lineal en una escala logarítmica. Un resultado de (arriba) es el hecho de que el f_0 de cualquier nota es el doble del f_0 de la nota con la misma clase de tono en la octava anterior. Por ejemplo, si el f_0 de la nota A4 está fijado a 440 Hz, esto hace que el f_0 de la nota A5 880 Hz, y el f_0 de la nota A3 220 Hz. Además, establecer la f_0 de A4 en 440 Hz haría que la f_0 de A#4 466,16 Hz, la f_0 de Ab4 415,30 Hz, y así sucesivamente.

2.4.1. FRECUENCIA FUNDAMENTAL Y TONO

La frecuencia fundamental, a menudo denominada simplemente fundamental, se define como la frecuencia más baja de una forma de onda periódica. En música, lo fundamental es el tono musical de una nota que se percibe como el presente parcial más bajo.

Las señales sonoras son periódicas, por lo tanto, por definición, existe un $T > 0$ tal que:

$$\forall t, X(t) = X(t + T) \quad (2.3)$$

De lo que se deduce que existe un conjunto infinito de valores de $T > 0$ que verifican esta propiedad, de hecho definimos el período de una señal como el valor positivo más pequeño de T para el que se cumple la propiedad. La frecuencia fundamental f_0 se define formalmente como el recíproco del período. Esta definición es válida para cualquier señal periódica, independientemente de su forma. En el caso de la onda sonora, la percepción de la frecuencia fundamental se denomina tono. El tono se define como la altura tonal de un sonido, está estrechamente relacionado con la frecuencia fundamental, sin embargo, sigue siendo un concepto musical relativo a diferencia del F_0 de una señal que es un valor matemático absoluto. De hecho, la relación entre el tono y F_0 no es ni biyectiva ni invariante. (Asswad, 2020)

En teoría musical, el tono se define en un espacio discreto a diferencia del espacio de frecuencia continuo. Además, la percepción humana de la frecuencia es logarítmica, por lo

que la obtención del siguiente tono corresponde a la multiplicación de la frecuencia por un cierto valor r . (Asswad, 2020)

2.5. SEGMENTACIÓN TEMPORAL

La segmentación temporal es la tarea de encontrar los límites de tiempo de las notas musicales pertenecientes a un audio. El inicio de una nota musical se define como la hora en que comienza y su hora de finalización es la compensación.

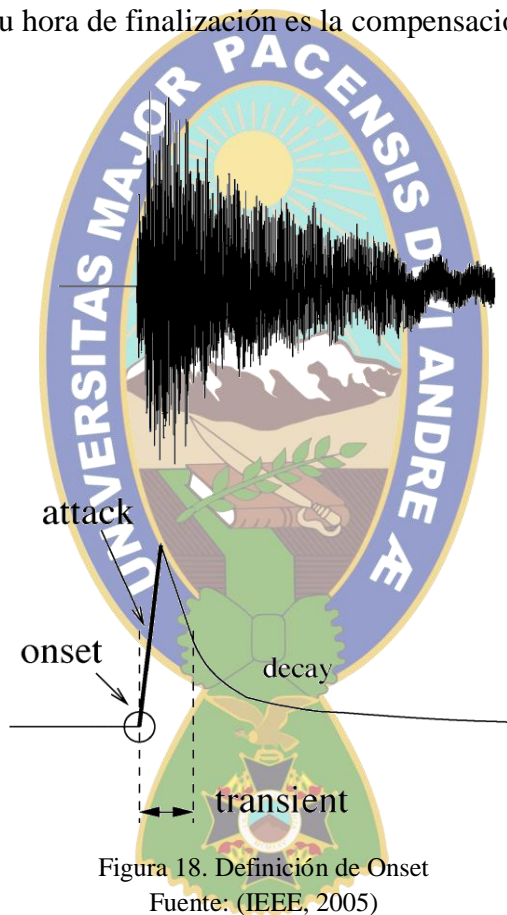


Figura 18. Definición de Onset
Fuente: (IEEE, 2005)

La forma de la señal varía según los instrumentos. El perfil de inicio en la figura 20 corresponde a un instrumento que produce ráfagas de energía repentinas, como un instrumento de acorde pellizcado (piano, guitarra, etc.) o un instrumento de percusión, a

diferencia de los instrumentos de cuerda inclinada y los instrumentos de viento que no exhiben tales ráfagas de energía.

2.5.1. FUNCIÓN DE DETECCIÓN DE INICIO

Las funciones de detección de inicio fueron diseñados para reconocer dos propiedades de la señal asociadas con un ataque agudo: la rapidez del cambio de señal y el aumento de energía. (Masri y Bateman, 1996).

Se eligió un método en el dominio de la frecuencia debido a su capacidad para revelar tanto los cambios en la energía general como la concentración de energía en la frecuencia. La ubicación de la frecuencia de la energía es importante porque el cambio repentino de la señal provocará discontinuidades de fase; en el espectro de frecuencias esto aparece como energía de alta frecuencia.

2.5.1.1. CONTENIDO DE ALTA FRECUENCIA (HFC)

La función propuesta por (Masri y Bateman, 1996) favorece las ráfagas de energía de banda ancha sobre los cambios en la modulación de amplitud y otorga un mayor peso a los contenedores espectrales de alta frecuencia.

$$E = \sum_{k=2}^{\frac{n}{2}+1} \{|X(k)|^2\} \quad (2.12)$$

Dónde: E es la función de energía para la trama actual, N es la longitud de la matriz FFT (por lo que $n/2 + 1$ corresponde a la frecuencia $F_s/2$, F_s es la frecuencia de muestreo), X (k) es el k-ésimo bin de la FFT.

La función enfatiza las ráfagas de energía de alta frecuencia, lo que la hace más adaptada a los inicios de percusión que las cuerdas con arco o los instrumentos de viento.

2.6. SEÑALES DE AUDIO

El procesamiento de la señal de audio está en el corazón de la grabación, mejora, almacenamiento y transmisión de contenido de audio. El procesamiento de la señal de audio se utiliza para convertir entre formatos analógicos y digitales, para cortar o aumentar los rangos de frecuencia seleccionados, para eliminar el ruido no deseado, para agregar efectos y para obtener muchos otros resultados deseados. (Koontz, 2016)

Para analizar el audio eficientemente con computadora, debe ser grabado y almacenado digitalmente. Los dispositivos de registro toman medidas discretas del desplazamiento de la presión del aire con respecto a un estado estable, con una periodicidad fija T_s conocida como período de muestreo. Durante este proceso, una señal $g(t)$ que es continua en el tiempo se convierte en una representación digital $g[k]$ que tiene un tiempo discreto y está indexada por el número de muestra k :

$$g[k] = g(kT_s) \quad (2.13)$$

En la ecuación 2.13 el período de muestreo T_s se puede representar de manera equivalente por la tasa de muestreo f_s , que es la inversa de T_s , es decir, el número de muestras tomadas por segundo. Los tonos complejos armónicos son los más interesantes al analizar señales musicales. Su presencia es típicamente lo que nos gustaría inferir al analizar señales muestreadas. Los tonos complejos armónicos son oscilaciones de la presión del aire con componentes de frecuencia que son múltiplos enteros positivos de f_0 :

$$g(t) = \sum_h A_h \sin(2\pi f_h t + \phi_h) \quad (2.14)$$

Cada componente de frecuencia es aproximadamente sinusoidal, lo que significa que se puede describir con una amplitud A_h , una frecuencia f_h y una fase ϕ_h . La figura 21 ilustra un ejemplo de la relación entre un tono puro en tiempo continuo y tiempo discreto. 22 es una conversión con pérdida y, como resultado, la información de frecuencia se borrará si la señal

continua contiene componentes de frecuencia mayor que $\frac{f_s}{2}$, también conocida como frecuencia de Nyquist. En la Figura 21, el tono puro oscila a una velocidad igual a la de Nyquist y, por lo tanto, es detectable. El muestreo insuficiente en la Figura 22 conduce a un efecto indeseable llamado aliasing. Aquí, el tono puro completa más de la mitad de una oscilación entre muestras. Como resultado, no hay suficiente información en la señal discreta para garantizar que haya frecuencias presentes que sean más altas que las de Nyquist.

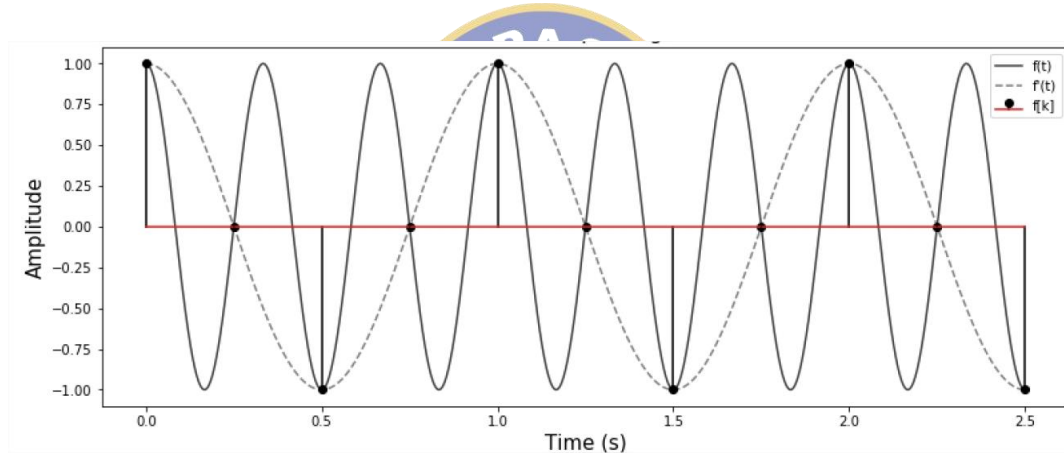


Figura 19: Sinusoide que oscila a 3 Hz con una fase de $\pi/2$ y amplitud unitaria. La señal se muestrea a una frecuencia de $f_s = 6$ Hz.
Fuente. (Cwtkowitz, 2019)

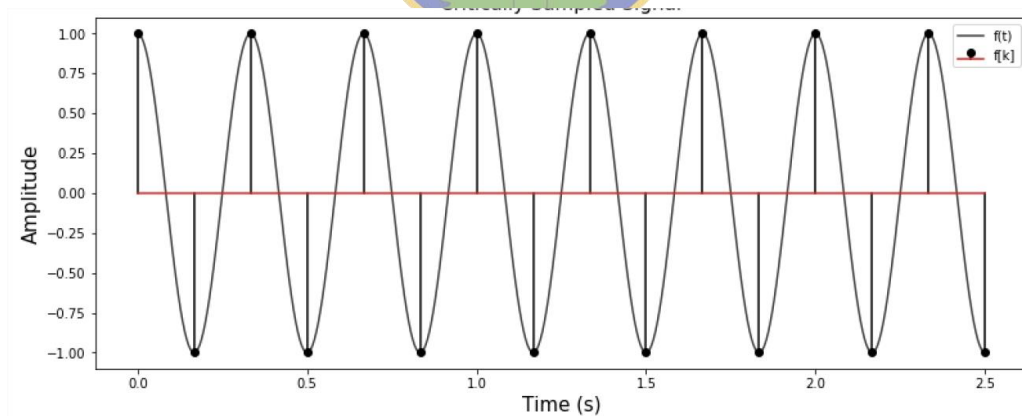


Figura 20: Sinusoide oscilante a 3 Hz con una fase de $\pi/2$ y amplitud unitaria. La señal se muestrea a una frecuencia de $f_s = 4$ Hz.

Figura 22. Fuente. (Cwtkowitz, 2019)

Los ejemplos de las Figuras 21 y 22 están muy alejados de la complejidad de las señales musicales naturales. Normalmente, los tonos armónicos complejos producidos por instrumentos acústicos son transitorios y dependientes del escenario.

Las formas de onda asociadas con dos grabaciones cortas de una guitarra acústica y un teclado electrónico tocando la nota G3 en las Figuras 23 y 24 revelan una expectativa más realista. En ambos ejemplos, la presencia de tonos complejos armónicos se evidencia por la aparición periódica de la forma de onda. La periodicidad más baja dentro de cada uno ocurre aproximadamente cada $T_1 \approx 5$ milisegundo. A partir de esto, se puede afirmar que $f_0 = \frac{1}{T_1} \approx 200$ Hz. Esto estaría muy cerca de la f_0 nominal para G3 bajo la sintonía A4 = 440 Hz, que sería 196 Hz.

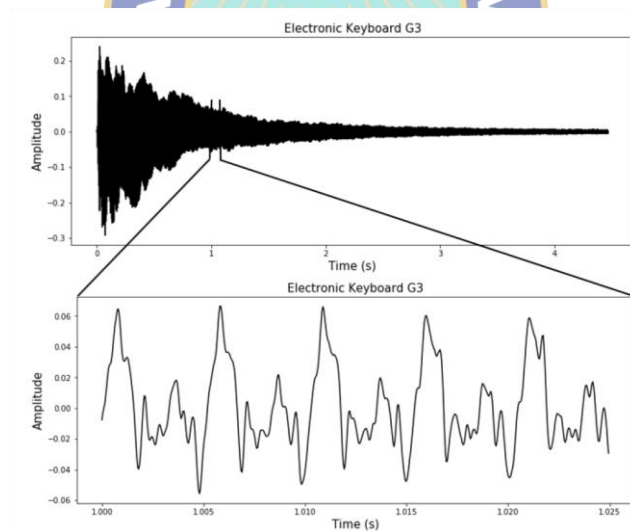


Figura 21: Forma de onda de una guitarra acústica que toca la nota G3, junto con un primer plano de la señal de 25 ms a partir de 1 segundo.

Fuente. (Cwitkowitz, 2019)

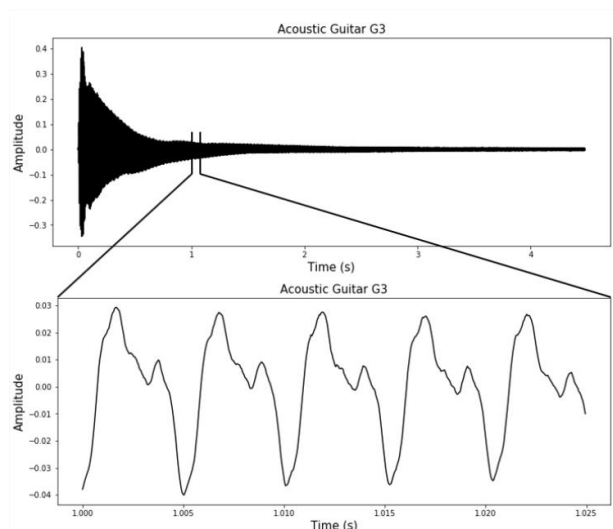


Figura 22: Forma de onda de un teclado electrónico que toca la nota G3, junto con un primer plano de la señal de 25 ms a partir de 1 segundo.

Fuente. (Cwitkowitz, 2019)

2.7. ANÁLISIS ARMÓNICO

Análisis armónico, es un procedimiento matemático para describir y analizar fenómenos de naturaleza periódica recurrente. Muchos problemas complejos se han reducido a términos manejables mediante la técnica de dividir curvas matemáticas complicadas en sumas de componentes comparativamente simples. Muchos fenómenos físicos, como las ondas sonoras, las corrientes eléctricas alternas, las mareas y los movimientos y vibraciones de la máquina, pueden ser de carácter periódico. (Hosch, 2006)

El Análisis Armónico es una rama de las matemáticas relacionada con la representación de señales y la generalización de nociones de transformadas de Fourier, es decir, una forma extendida del análisis de Fourier.

2.7.1. TRANSFORMADA DE FOURIER

La transformada de Fourier (FT) descompone una señal en las frecuencias que la componen. (Dubey, 2018)

Una FT es una transformación matemática que descompone una función en sus frecuencias constituyentes. El término transformada de Fourier se refiere tanto a la representación del dominio de frecuencia como al dominio de tiempo.

La Transformada de Fourier es una herramienta matemática utilizada para muchos tipos de procesamiento de señales, incluidos el audio y la música. La Transformada de Fourier es una función de frecuencia que proporciona sus respectivas amplitudes y fases. El dominio de la frecuencia es a menudo más práctico e intuitivo para las aplicaciones de procesamiento de señales que el análisis directo de la grabación del dominio del tiempo sin procesar. (Quenneville, 2018)

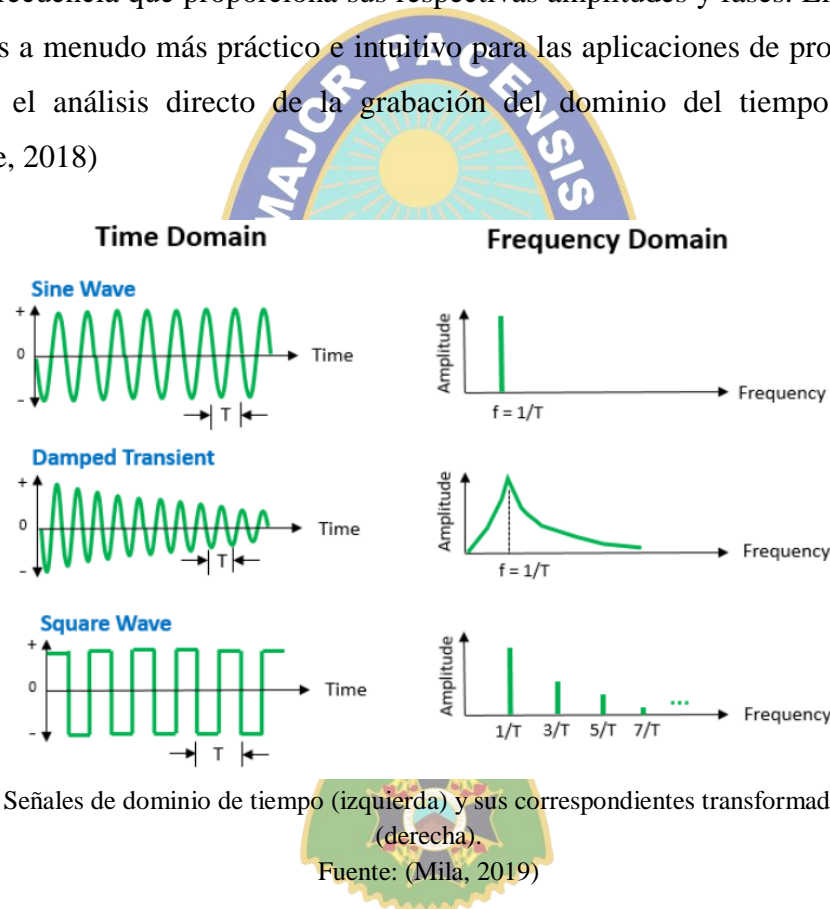


Figura 23: Señales de dominio de tiempo (izquierda) y sus correspondientes transformadas de Fourier (derecha).

Fuente: (Mila, 2019)

La transformación de Fourier comienza con el principio fundamental de que cualquier curva o función periódica (es decir, repetitiva) puede describirse mediante una suma infinita de sinusoides. Esto se conoce como la expansión de Fourier de la función periódica $X(t)$, y se describe matemáticamente como:

$$X(t) = a_0 + \sum_{m=1}^{\infty} a_m \cos \frac{2\pi m t}{T} + \sum_{n=1}^{\infty} b_n \sin \frac{2\pi n t}{T} \quad (2.15)$$

En donde, $T/2\pi$ es el periodo de $X(t)$, los coeficientes $a_0, a_1, a_2 \dots$ y $b_1, b_2 \dots$ son las amplitudes de un componente seno o coseno a la frecuencia $2\pi m/T$

La tarea general de la transformada de Fourier, entonces, es encontrar estos coeficientes a_m y b_n , que corresponden a la amplitud de la frecuencia n .

2.7.2. TRANSFORMADA DISCRETA DE FOURIER

La Transformada Discreta de Fourier (DFT) es una de las herramientas más importantes en el procesamiento de señales digitales. Primeramente la DFT puede calcular el espectro de frecuencias de una señal, esto es una examinación directa a la información codificada en la frecuencia, fase y amplitud de acuerdo a sus componentes sinusoidales. En segundo lugar, la DFT puede encontrar la respuesta de la frecuencia de un sistema a partir de la respuesta del impulso del sistema, y viceversa. Esto permite que los sistemas se analicen en el dominio de la frecuencia. (Smith, 2003)

La transformada discreta de Fourier es un tipo de transformada discreta utilizada en el análisis de Fourier. Transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo. Pero la DFT requiere que la función de entrada sea una secuencia discreta y de duración finita.

Uno de estos conjuntos de características a partir de los cuales se puede inferir directamente información musical es una medida del grado en que un rango de frecuencias está presente en la señal. La DFT transforma una señal discreta en una combinación lineal de funciones de base sinusoidal. Las funciones básicas se pueden especificar por magnitud A_k , frecuencia f_k y fase ϕ_k . Estos son representados como un conjunto de coeficientes complejos.

$$F(\mu) = \sum_{k=0}^{M-1} g[k] * e^{-j\frac{2\pi\mu k}{M}}, \mu = 0, \dots, M-1 \quad (2.16)$$

Es esencial descomponer la señal de entrada x_n en un conjunto de componentes de frecuencia utilizando las sinusoides complejas $e^{-j\frac{2\pi\mu k}{M}} = \cos\left(\frac{2\pi\mu k}{M}\right) - j * \sin\left(\frac{2\pi\mu k}{M}\right)$.

Esto genera los coeficientes de Fourier como ser: coeficientes complejos: $F(\mu) = a_\mu + jb_\mu$ para las frecuencias dentro del conjunto $f_\mu = \frac{2\pi\mu}{M}$, donde a_μ es la parte real y b_μ la parte imaginaria. Es importante tener en cuenta que el tamaño M de la DFT puede expandirse artificialmente rellendo con cero la señal de entrada $g[k]$ para un soporte de frecuencia más granular, pero por lo demás normalmente se establece en el tamaño de $g[k]$.

El valor absoluto de los coeficientes complejos $A_\mu = |F(\mu)| = \sqrt{a_\mu^2 + b_\mu^2}$ representa la magnitud de las bases, mientras que los ángulos $\phi_\mu = \arctan \frac{b_\mu}{a_\mu}$ representan la fase.

Para muchas tareas de estimación musical, la magnitud de la DFT es más informativa que la fase. La magnitud DFT puede elevarse a la segunda potencia para generar lo que se conoce como espectro de potencia. (Cwitkowitz, 2019)

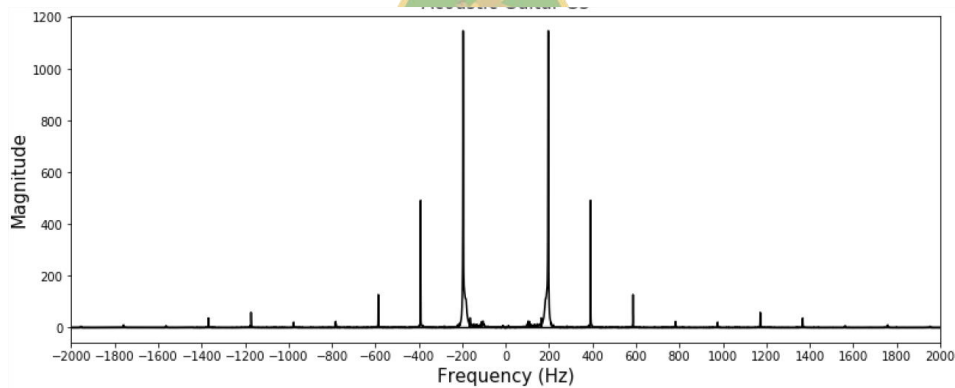


Figura 24: Espectro de magnitud de una guitarra acústica tocando la nota G3. Hay una energía fuerte en $h = 1$, que corresponde a 196 Hz, menos en $h = 2$, incluso menos en $h = 3$, y una energía pequeña pero notable entre los otros armónicos.

Fuente: (Cwitkowitz, 2019)

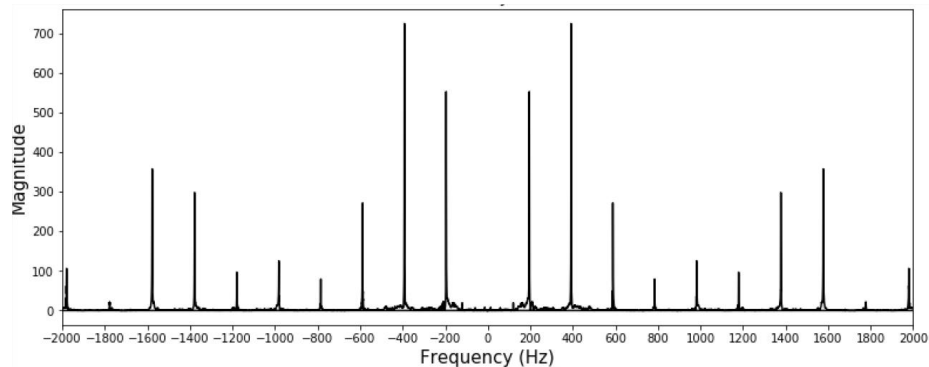


Figura 25: Espectro de magnitud de un teclado electrónico que toca la nota G3. Hay una energía significativa pero variable en casi todos los armónicos dentro de lo que se muestra en el DFT, que contiene los primeros diez armónicos.

Fuente: (Cwitkowitz, 2019)

Hay varios atributos importantes de estos DFT a tener en cuenta. Primero, en el dominio de la frecuencia, es mucho más obvio dónde la energía se distribuye a través de la frecuencia. En segundo lugar, está claro que los tonos complejos que se analizan son armónicos, es decir, todos los componentes de frecuencia son múltiplos enteros de $f_0 = 196$ Hz. En tercer lugar, el timbre de cada instrumento es indudablemente único. La energía armónica de la guitarra acústica no se distribuye de manera muy uniforme. En cambio, el primer armónico es excepcionalmente fuerte y hay algo de energía en el segundo y tercer armónico. Por el contrario, la energía armónica del teclado electrónico está mucho más dispersa y el segundo armónico tiene la mayor cantidad de energía. Curiosamente, estos timbres son exclusivos de las características específicas de cada instrumento físico, no simplemente de la clase de instrumento.

2.7.3. TRANSFORMADA RÁPIDA DE FOURIER

En teoría, la transformada discreta de Fourier descrita en la Sección 2.6.2 es suficiente para el análisis del espectro de señales de audio. Pero se presenta un antiguo problema computacional: la eficiencia. Aquí es donde entra en juego la Transformada Rápida de

Fourier (FFT). En su forma pura, la transformada discreta de Fourier $g[k]$ debe sumar valores de $g[k]_0$ a $g[k]_{M-1}$ para cada frecuencia μ , lo que significa una complejidad de tiempo $O(N^2)$ para N puntos de datos. Con la FFT, esto se puede reducir a $O(N \log_2 N)$ usando un enfoque de divide y vencerás.

El descubrimiento fundamental de la FFT es que la transformada discreta de Fourier de N puntos se puede describir como la suma de dos DFT, cada una de N/2 puntos. Específicamente, es la suma de la transformada de Fourier de los puntos pares E_k y un exponente complejo W^K multiplicado por la transformada de Fourier de los puntos impares O_k , donde W se define como $W = e^{-j\frac{2\pi}{M}}$ para facilitar la formula, porque este exponente es de uso tan frecuente. Escribiendo la ecuación se tiene.

$$\begin{aligned}
 F(\mu) &= \sum_{k=0}^{M-1} g[k] * e^{-j\frac{2\pi\mu k}{M}} \\
 F(\mu) &= \sum_{\mu=0}^{\frac{M}{2}-1} g_{2k} * e^{j\frac{2\pi k(2\mu)}{M}} + \sum_{k=0}^{\frac{M}{2}-1} g_{2k+1} * e^{j\frac{2\pi k(2\mu+1)}{M}} \\
 F(\mu) &= \sum_{k=0}^{\frac{M}{2}-1} g_{2k} * e^{j\frac{2\pi k\mu}{(\frac{M}{2})}} + W^\mu \sum_{k=0}^{\frac{M}{2}-1} g_{2k+1} * e^{j\frac{2\pi k\mu}{(\frac{M}{2})}} \\
 F(\mu) &= H_\mu^e + W^k H_\mu^o, \mu = 0, \dots, M-1 \quad (2.17)
 \end{aligned}$$

Todo lo que hay que hacer es utilizar esta función de forma recursiva. Sin embargo, existe una restricción importante sobre el uso recursivo de esta función. La ecuación (4.14) supone un número igual de puntos pares e impares, por lo que N debe ser un número par. Para poder dividir repetida y uniformemente un conjunto de datos en sus partes pares e impares, todas las divisiones deben permanecer pares en longitud, por lo tanto, N debe ser una potencia de

dos. En la práctica, esto presenta muy pocos problemas, ya que rellenar o desplazar los datos puede lograr este objetivo.

$$F(\mu) = \begin{cases} F(\mu), & N = 1 \\ E_k + W^k O_k, & 0 \leq \mu < \frac{N}{2} \\ E_{\mu - \frac{N}{2}} + W^k O_{\mu - \frac{N}{2}}, & \frac{N}{2} \leq \mu \leq N \end{cases} \quad (2.18)$$

El caso base es bastante simple. Cuando el tamaño de los datos $N = 1$, la función que se transforma es simplemente una línea horizontal en $F(\mu)$. Por tanto, la transformada de Fourier producirá $F(\mu) = a_0 = f(\mu)$. Esto se puede implementar de manera bastante sencilla usando dos llamadas recursivas y combinando los resultados de acuerdo con la Ecuación 2.17. Hay muchas implementaciones diferentes de la Transformada Rápida de Fourier, todas basadas en esta estrategia de divide y vencerás. Todas las mejores implementaciones logran una complejidad de tiempo $O(N \log^2 N)$ y no requieren espacio de memoria adicional.

2.7.4. TRANSFORMADA DE FOURIER DE TIEMPO REDUCIDO

La transformada de Fourier de tiempo reducido, como sugiere su nombre, se utiliza para ventanas muy pequeñas de una curva muestreada. Mientras que el análisis de Fourier de, por ejemplo, un terremoto realizará un análisis de toda la curva sísmica, dando un vector de transformación singular, el análisis de audio necesita producir no un solo espectro de todas las frecuencias presentes, sino el espectro cambiante a lo largo del tiempo, o espectrograma. La forma más común de utilizar la Transformada Rápida de Fourier para música es dividir la grabación en ventanas o marcos. Esto también sirve para hacer que las operaciones de la transformada de Fourier sean más manejables al tratar con ventanas de igual longitud y bastante pequeñas.

Todas las grabaciones de audio utilizadas en los experimentos fueron archivos WAV de 16 bits, muestreados a 44,1 kHz. Esto significa que en un segundo de audio hay 44,100 puntos de datos. Se han utilizado varios tamaños de ventana para los algoritmos de detección de tono

durante las últimas décadas, desde 512 muestras (Kunieda, Shimamura y Suzuki ,1996) hasta 4096 u 8192 muestras (Klapuri, 2003). El tamaño de la ventana ha variado con el tiempo a medida que las computadoras se han vuelto más eficientes y los vectores más grandes se han vuelto más manejables para el análisis. Sin embargo, un factor más importante al elegir un tamaño de ventana es el caso de uso (Klapuri, 2003).

Los algoritmos de detección de ritmo y tempo basados en el espectro también utilizan ventanas muy pequeñas porque se centran en el dominio del tiempo más que en el dominio de la frecuencia (Laroche, 2003). En tales sistemas, es común utilizar un espacio entre ventanas mucho más estrecho que el tamaño de la ventana. Goto y Muraoka (1999) utilizan audio reducido a 22,05 kHz o 11,025 kHz y ventanas de 1024 muestras, pero el desplazamiento de ventana es sólo de 256 o 128 muestras, lo que significa que hay una precisión de tiempo de 11,61 ms.

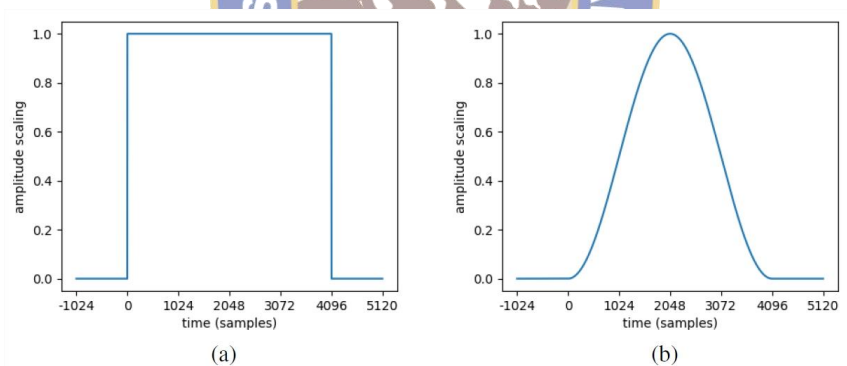


Figura 26: Una comparación de las funciones de ventanas para un marco de audio de 4096 muestras. La ventana rectangular.

Fuente: (Quenneville, 2019)

Sin embargo, para la detección del tono musical, la precisión de la frecuencia suele ser más importante que la precisión del tiempo. Considere un tempo común, 120 latidos por minuto (bpm). Esto significa que una negra tiene medio segundo o 500 ms de duración, una corchea tiene 250 ms y una semicorchea, 125 ms. Klapuri (2003) usa una ventana de 93 ms (4096 muestras) o 186 ms (8192 muestras), lo que significa que la resolución de semicorchea se

puede lograr completamente con la ventana más pequeña. Una técnica común para mejorar la precisión del tiempo es usar ventanas superpuestas, espaciadas más estrechamente que su ancho. Sin embargo, esto debe hacerse solo dentro de lo razonable para evitar un alto costo computacional.

Comparando con la DFT podríamos decir que, un inconveniente de la DFT es que no puede especificar cuándo está activo cada componente de frecuencia. Más bien, asume que las frecuencias están activas para la totalidad de la señal. La energía espectral de las señales musicales evoluciona rápidamente con el tiempo. La información de tiempo correspondiente a las frecuencias activas es crucial para una correcta transcripción musical. Se puede generar un conjunto mucho más poderoso de características para señales de audio empleando la Transformada de Fourier de tiempo reducido (STFT) (Wein, Jannach, Vatolkin, y Rudolph, 2016).

$$F(\mu, \lambda) = \sum_{k=0}^{M-1} \omega[k] g[k + \lambda R] * e^{-j \frac{2\pi \mu k}{M}}, \mu = 0, \dots, M-1 \quad (2.19)$$

El STFT rompe eficazmente la señal analizada en L marcos de ventana de tamaño M y genera un conjunto de coeficientes espectrales para cada marco, avanzado con tamaño de salto R e indexado con λ , utilizando el DFT. En 2.19 se emplea una función de ventana $\omega[k]$ para aislar un marco de análisis del resto de la señal. También es útil para reducir la actividad de la señal cerca del comienzo y el final del marco de análisis, lo que producirá espectros con picos de frecuencia más localizados.

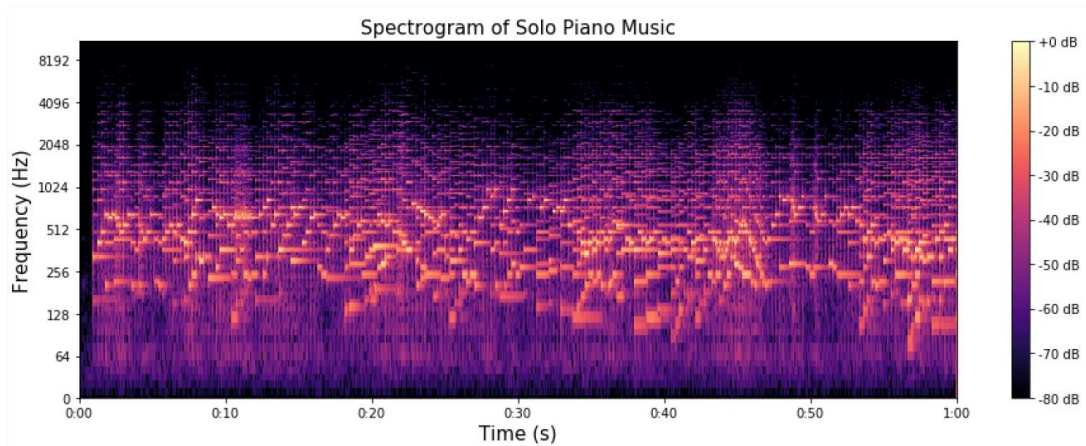


Figura 27: Espectrograma de la grabación de un solo de piano. El STFT se genera con tamaño $M = 2048$ y tamaño de salto $R = 512$.
Fuente: (Cwitkowitz, 2019)

Esto puede resultar en una dispersión de los espectros para modelar con precisión la señal, lo cual no es deseable para el análisis. Una mejor opción sería utilizar una función de ventana en forma de campana conocida como ventana Hanning:

$$\omega[k] = 0.5 - 0.5 \cos\left(k \frac{2\pi}{M-1}\right) \quad (2.20)$$

2.8. AMPLITUDES ARMÓNICAS

Klapuri (2006) propone un método robusto para la estimación de frecuencias fundamentales en señales musicales. El método busca la frecuencia fundamental que maximiza una fuerza de frecuencia sobre las frecuencias candidatas en un blanqueamiento espectral.

2.8.1. BLANQUEAMIENTO ESPECTRAL

Uno de los grandes desafíos en la estimación de F_0 es hacer que los sistemas sean robustos para diferentes fuentes de sonido. Una forma de lograr esto es intentar suprimir la información tímbrica antes de la estimación real de F_0 . Diferentes fuentes pueden tener diferente información tímbrica en el espectro de la señal. Para detectar y analizar las frecuencias de las diferentes fuentes Klapuri (2006) propone suprimir la información

tímbica antes de detectar las frecuencias dominantes en el espectro. Este proceso se realiza mediante una secuencia de transformaciones.

- Aplicar el filtro de paso de banda al espectro $X(F)$ para obtener frecuencias centrales c_b donde b es el índice de subbanda del espectro filtrado. Cada subbanda tiene una respuesta de potencia triangular $H_b(F)$ tal que se extiende desde c_{b-1} a c_{b+1} y es cero en todo lo demás.
- Calcular la desviación estándar σ_b dentro de las subbandas

$$\sigma_b = \left(\frac{1}{K} \sum_F H_b(F) |X(F)|^2 \right)^{\frac{1}{2}} \quad 2.21$$

Dónde K es el número de intervalos de frecuencia de la transformada de Fourier

- Calcular coeficientes de compresión $\gamma_b = \sigma_b^{v-1}$ donde v es el parámetro de blanqueamiento, el valor propuesto es $v = 0.33$.
- Interpolar $\gamma(F)$ para todos los contenedores de frecuencia F de γ_b
- Finalmente calcular el espectro blanqueado $Y(F)$ ponderando el espectro de entrada por los coeficientes de compresión obtenidos $Y(F) = \gamma(F)X(F)$

2.8.2. FUNCION DE PROMINENCIA

La fuerza de las frecuencias fundamentales candidatas se evalúan mediante una función de prominencia S que calcula la suma ponderada de las amplitudes de los parciales armónicos.

$$s(\tau) = \sum_{h=1}^H g(\tau, h) |Y(hf_\tau)| \quad 2.22$$

Donde $f_\tau = f_s/\tau$ es la frecuencia fundamental candidata correspondiente al periodo τ y $g(\tau, h)$ es el peso de h parcial del periodo τ .

2.9. PRECISIÓN Y EXHAUSTIVIDAD

Se aplica las métricas de precisión y exhaustividad propuestas por Music Information Retrieval, una ciencia interdisciplinaria que se encarga en la recuperación de información de la música.

La precisión es la fracción de ejemplos positivos verdaderos entre los ejemplos que el algoritmo clasificó como positivos. En otras palabras, el número de verdaderos positivos dividido por el número de falsos positivos más los verdaderos positivos.

$$P = \frac{T_p}{T_p + F_p} \quad 2.23$$

Exhaustividad o recall (R), también conocido como sensibilidad, es la fracción de ejemplos clasificados como positivos, entre el número total de ejemplos positivos. En otras palabras, el número de verdaderos positivos dividido por el número de verdaderos positivos más falsos negativos.

$$R = \frac{T_p}{T_p + F_n} \quad 2.24$$

El valor F es el promedio ponderado de precisión y recuperación. Por tanto, esta puntuación tiene en cuenta tanto los falsos positivos como los falsos negativos.

$$F = 2 \frac{P * R}{P + R} \quad 2.25$$

2.10. NOTACIÓN BIG O

La notación Big O es una herramienta muy funcional para determinar la complejidad de un algoritmo que estemos utilizando, permitiéndonos medir su rendimiento en cuanto a uso de

espacio en disco, recursos (memoria y ciclos del reloj del CPU) y tiempo de ejecución, entre otras, ayudándonos a identificar el peor escenario donde el algoritmo llegue a su más alto punto de exigencia. (Fuentes, 2018)

Las funciones de complejidad algorítmica más habituales en las cuales el único factor del que dependen es el tamaño de la muestra de entrada n , ordenadas de mayor a menor eficiencia son:

- $O(1)$: Complejidad constante. Cuando las instrucciones se ejecutan una vez.
- $O(\log n)$: Complejidad logarítmica. Esta suele aparecer en determinados algoritmos con iteración o recursión no estructural, ejemplo la búsqueda binaria.
- $O(n)$: Complejidad lineal. Es una complejidad buena y también muy usual. Aparece en la evaluación de bucles simples siempre que la complejidad de las instrucciones interiores sea constante.
- $O(n \log n)$: Complejidad cuasi-lineal. Se encuentra en algoritmos de tipo divide y vencerás como por ejemplo en el método de ordenación quicksort y se considera una buena complejidad. Si n se duplica, el tiempo de ejecución es ligeramente mayor del doble.
- $O(n^2)$: Complejidad cuadrática. Aparece en bucles o ciclos doblemente anidados. Si n se duplica, el tiempo de ejecución aumenta cuatro veces.
- $O(2^n)$: Complejidad exponencial. No suelen ser muy útiles en la práctica por el elevadísimo tiempo de ejecución. Se dan en subprogramas recursivos que contengan dos o más llamadas internas n .
- $O(n!)$; explosión combinatoria. Un algoritmo que siga esta complejidad es un algoritmo totalmente fallido. Una explosión combinatoria se dispara de tal manera

que cuando el conjunto crece un poco, lo normal es que se considere computacionalmente inviable.

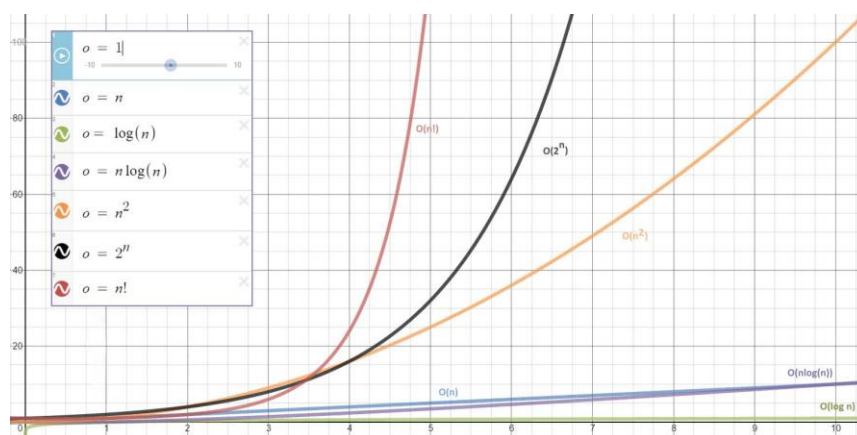


Figura 28. Comparación de órdenes de complejidad

Fuente: (Alarcón, 2016)

2.11. KIT DE HERRAMIENTAS MULTIPLAFORMA

Qt es un conjunto de bibliotecas C++ multiplataforma que implementan API de alto nivel para acceder a muchos aspectos de los sistemas móviles y de escritorio modernos. Estos incluyen servicios de ubicación y posicionamiento, conectividad multimedia, NFC y Bluetooth, un navegador web basado en Chromium, así como el desarrollo de IU tradicional.

PyQt5 es un conjunto completo de enlaces de Python para Qt v5. Se implementa como más de 35 módulos de extensión y permite que Python se utilice como un lenguaje de desarrollo de aplicaciones alternativo a C++ en todas las plataformas compatibles.

PySide2 es el módulo oficial de Python del proyecto Qt for Python, que proporciona acceso al framework completo Qt 5.12+.

CAPÍTULO 3. MARCO APLICATIVO

3.1. INTRODUCCIÓN.

La recuperación de información musical (MIR, por sus siglas en inglés, Music Information Retrieval) es la ciencia interdisciplinaria encargada de recuperar información de la música. MIR es un pequeño pero creciente campo de investigación con muchas aplicaciones en el mundo real. MIR es usado hoy en día en negocios y centros académicos para categorizar, manipular e incluso crear música como por ejemplo generación de partituras partiendo de un audio.

El presente trabajo tiene el objetivo de generar la partitura musical de una pieza musical partiendo de su audio en formato WAV. La transcripción musical es la práctica de anotar una pieza o un sonido que antes no estaba anotado y/o era impopular como música escrita.

En la generación automática de partitura se emplea un análisis profundo al espectro de frecuencia utilizando el análisis armónico. El proceso de conversión a partitura no es directo, se descomponen los aspectos musicales para recrear la partitura parte por parte.

La metodología planteada se basa en la programación modular, de la cual se definen los siguientes módulos.

- Módulo de extracción de datos.

Una vez obtenido los datos de un archivo .wav procedemos con las implementaciones.

- Módulo de preparación de datos: Funciones esenciales para un análisis eficaz del espectro de frecuencias.
- Análisis de Fourier: Implementación de la transformada de Fourier (SFFT)
- Computo de las frecuencias fundamentales.
- Función de detección de inicio: También llamado Onset.

Después del procesamiento y generación de frecuencias fundamentales se tiene como objetivo empezar a construir las notas musicales en dominio del tiempo.

- Módulo de generación de la partitura.
- Módulo de generación archivo MIDI.

El archivo MIDI posee todos los aspectos musicales los cuales se utilizan para generar una partitura musical en formato de imagen (.PNG).

La partitura se genera en base a las frecuencias obtenidas y los inicios de nota musical, se hallan la duración de cada nota para posteriormente crear el archivo MIDI que posee todos los aspectos musicales.

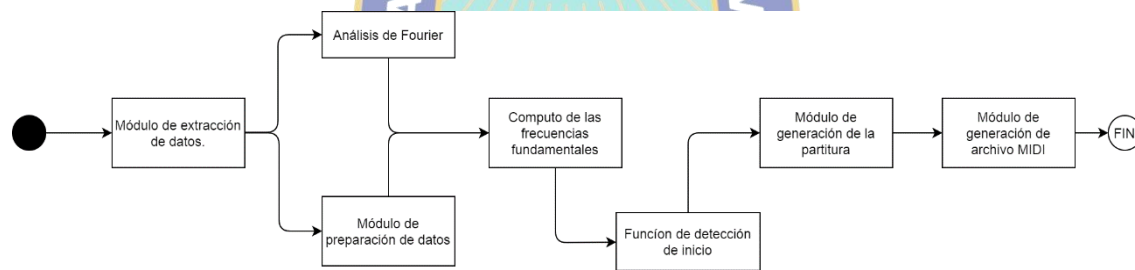


Figura 29. Estructura del proyecto dividido por módulos.

Fuente: (Elaboración propia)

3.2. DEFINICIÓN DEL PROBLEMA

La Transcripción automática de música es el proceso de convertir un registro de audio a notación simbólica, como una partitura o un archivo MIDI. Este proceso implica varias tareas que incluyen: detección de tonos distintos, detección del inicio de una nota musical o sonido, estimación de la duración y extracción de información rítmica mencionados en el capítulo anterior. Esta tarea se vuelve más compleja con un número mayor de instrumentos y un mayor nivel de polifonía.

3.3. ANÁLISIS DEL PROBLEMA

El problema radica en la complejidad que tiene la pieza musical al momento de empezar a procesar sus frecuencias, para ello se delimitó los casos a música monofónica de un solo instrumento.

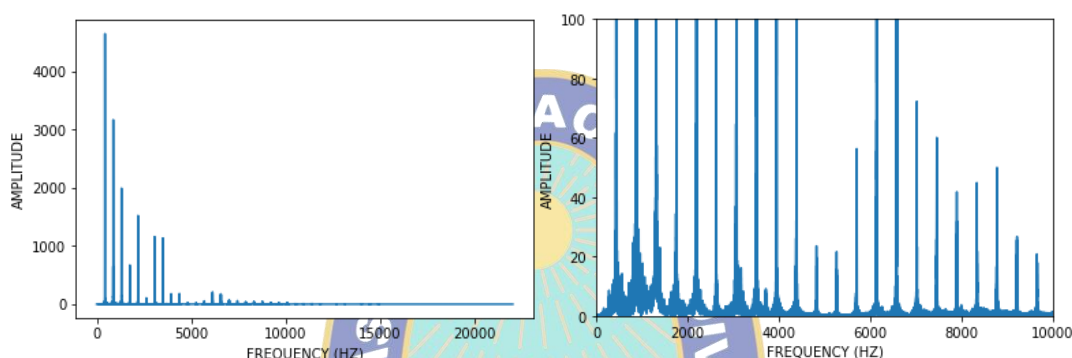


Figura 30. Espectro de frecuencias igualmente espaciadas de la nota A4 de un violín.

Fuente: (Elaboración propia)

Para comprender mejor esta situación dentro del dominio de la frecuencia, el desafío de la estimación del tono es dar sentido a un patrón de armónicos de la frecuencia base. El principio de armonicidad establece que los armónicos de una frecuencia son todos múltiplos de la frecuencia fundamental, por lo tanto, espaciados uniformemente a lo largo del espectro de frecuencias, causa un fenómeno interesante: el espectro de una sola nota es en sí mismo periódico, donde la frecuencia fundamental es el período, en pocas palabras la distancia entre cada pico es la misma. Esto se demuestra en la figura 31.

3.4. ANÁLISIS DE DATOS DE ENTRADA Y SALIDA

3.4.1. DATOS DE ENTRADA

Los datos que recibe el algoritmo constan de una dirección de carpeta (filepath) donde se encuentra el audio en formato WAV.

El algoritmo cuenta con una entrada en forma de señales musicales las cuales son suficientes para extraer la información necesaria, el archivo de entrada debe estar en formato .wav de 44100 samples.

3.4.2. DATOS DE SALIDA

La salida consiste en un archivo en formato PNG o PDF que contiene la partitura musical del archivo WAV. Uno de los objetivos planteados es generar un archivo MIDI el cual posea una correcta transformación de frecuencia a nota musical con su respectivo tiempo, el algoritmo exporta este archivo MIDI a partir de la partitura musical ya creada.

3.5. DISEÑO Y ANÁLISIS DEL ALGORITMO

3.5.1. EXTRACCIÓN DE DATOS

Para la extracción de datos se utiliza la librería wavfile de scipy.io la cual devuelve 2 datos.

- fs: Frecuencia de muestreo.
- data: Matriz de datos del archivo WAV.

La frecuencia de muestreo siempre será 44100 kHz.

La matriz de datos se determina de acuerdo al formato del archivo WAV.

Formato WAV	min	máximo	dtype de NumPy
Punto flotante de 32 bits	-1.0	+1.0	float32
PCM entero de 32 bits	-2147483648	+2147483647	int32
PCM entero de 24 bits	-2147483648	+2147483392	int32
PCM entero de 16 bits	-32768	+32767	int16

PCM entero de 8 bits	0	255	uint8
----------------------	---	-----	-------

Tabla 1. Formato de obtención de datos archivos WAV

3.5.2. PREPARACIÓN DE DATOS

Los datos de entrada obtenidos del archivo WAV pasan primeramente por un proceso de modificación para seleccionar los datos que nos interesan y así mejorar la precisión de los módulos de análisis armónico y procesamiento de la frecuencia fundamental. Todas las implementaciones se hicieron en el lenguaje de programación Python.

3.5.2.1. TAMAÑO DE VENTANA

Para aplicar la Transformada de Fourier a la música se necesita dividir la grabación en ventanas o marcos. Esto sirve para hacer que las operaciones de la transformada de Fourier sean más manejables al tratar con ventanas bastante pequeñas de igual longitud.

Para la detección del tono musical, la precisión de la frecuencia suele ser más importante que la precisión del tiempo. Considerando un tempo común, 120 latidos por minuto (bpm), esto significa que una negra tiene medio segundo o 500 ms de duración, una corchea tiene 250 ms y una semicorchea, 125 ms.

Utilizar tamaños de ventana de 93 ms (4096 muestras) o 186 ms (8192 muestras), llega a mejorar la resolución de notas como la semicorchea.

3.5.2.2. FUNCIÓN VENTANA

Una función de ventana (Window Function) se define como una función que produce valores dentro de un rango (la ventana) y valores de cero en todas partes fuera de la ventana. En este caso se utiliza la ventana de Hann. Esta ventana sirve como función de escalado para las amplitudes de entrada y se calcula fácilmente para cualquier longitud de cuadro. La figura 35 muestra una comparación de los factores de escala de amplitud de una ventana rectangular simple y de una ventana de Hann en un marco de audio de 4096 muestras.

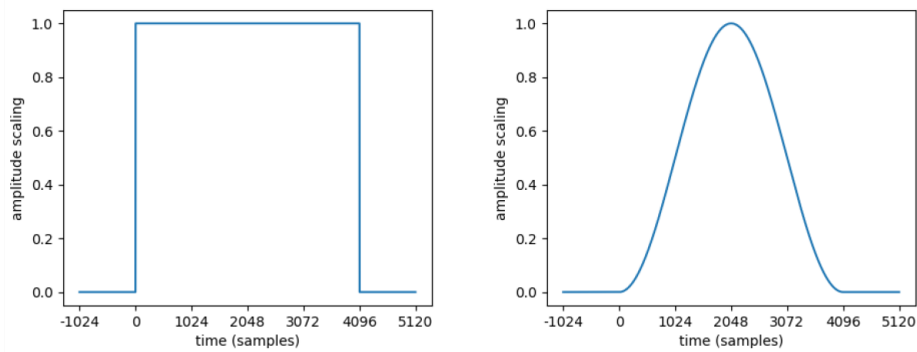


Figura 31. Comparación de las funciones de ventanas para un marco de audio de 4096 muestras.

Fuente: (Elaboración propia)

La ventana rectangular cumple los requisitos básicos de una función de ventana pero deja artefactos después del procesamiento. El uso de la ventana de Hann produce un espectro de Fourier más preciso.

3.5.2.3. RELLENO CERO

Relleno cero o Zero Padding no aumenta la resolución de la transformada de Fourier de una ventana, pero mejora la precisión de los picos espectrales. Esto es útil para convertir de índice en la transformada discreta de Fourier a frecuencia real (Hz) y nota musical. La Figura 36 muestra la diferencia entre un cuadro de audio de 1024 muestras procesado con una Transformada Rápida de Fourier con y sin relleno de ceros de 3072 muestras, creando un ancho de cuadro cuatro veces mayor que el original.

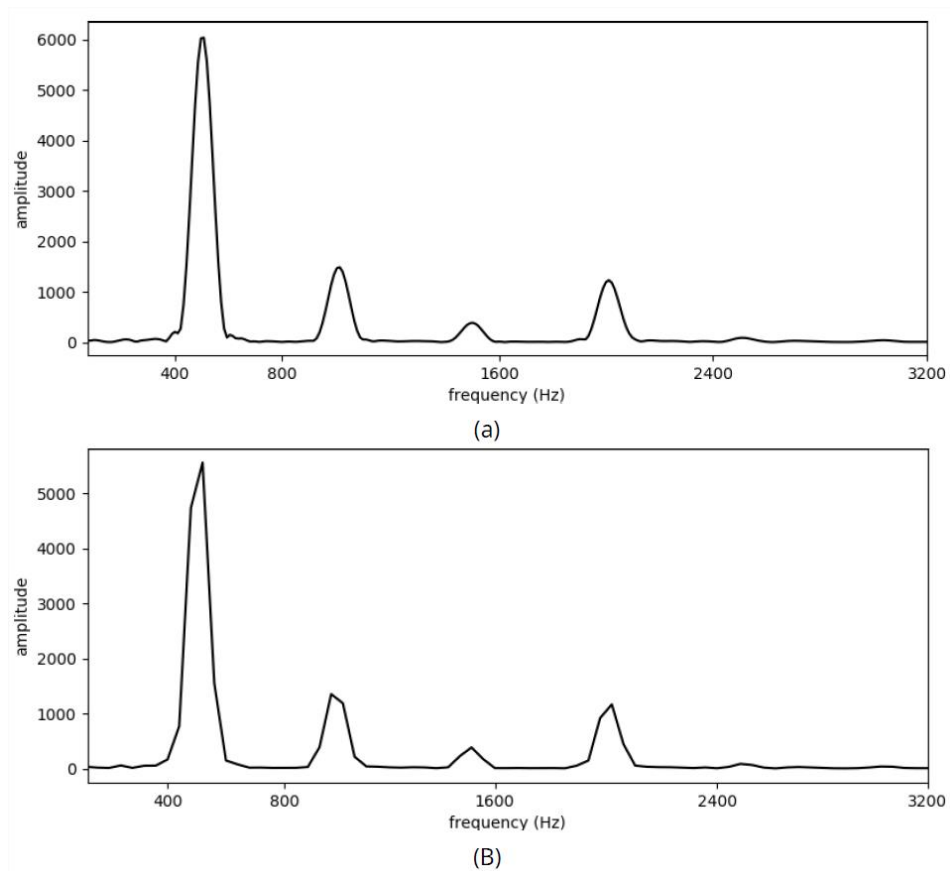


Figura 32. Una trama de 1024 muestras de audio procesada mediante la transformada rápida de Fourier.
Fuente: (Elaboración propia)

El espectro con relleno de ceros (a) tiene una curva más suave y picos ubicados con mayor precisión, a pesar de que en realidad no aumenta la resolución del espectro.

3.5.2.4. BLANQUEAMIENTO ESPECTRAL

Para detectar y analizar las frecuencias de las diferentes fuentes, se suprime la información tímbrica antes de detectar las frecuencias dominantes en el espectro.

Este proceso se realiza mediante el blanqueamiento espectral.

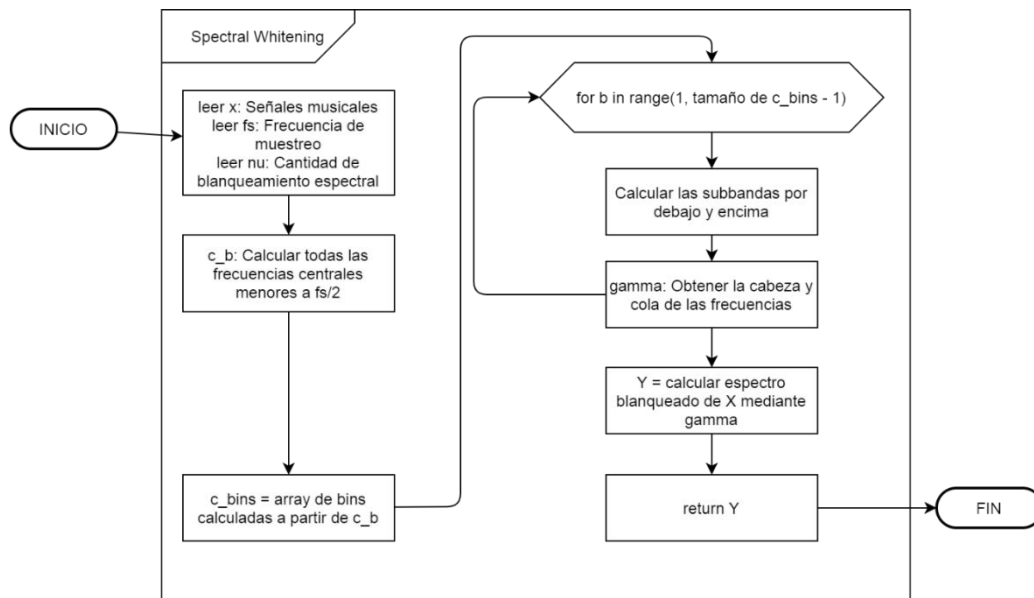


Figura 33. Algoritmo: blanqueamiento Espectral
Fuente: (Elaboración propia)

3.5.3. ANÁLISIS DE FOURIER

Después de la preparación de datos se implementa un módulo que calcule la Transformada de Fourier. En el capítulo anterior se describió el comportamiento de la transformada de Fourier, en este caso se utilizó la transformada rápida de Fourier (FFT).

La Transformada Rápida de Fourier es un algoritmo que determina la Transformada Discreta de Fourier de una entrada significativamente más rápido que calcularla directamente, también puede calcular su inversa. La FFT reduce el número de cálculos necesarios para un problema de tamaño N de $O(N^2)$ a $O(N \log N)$.

Para poder programar FFT primero se necesita definir la transformada discreta de Fourier (DFT). Entonces, como parámetro de entrada recibe señales obtenidas de los datos del archivo .wav para transformarlas en el dominio de frecuencia.

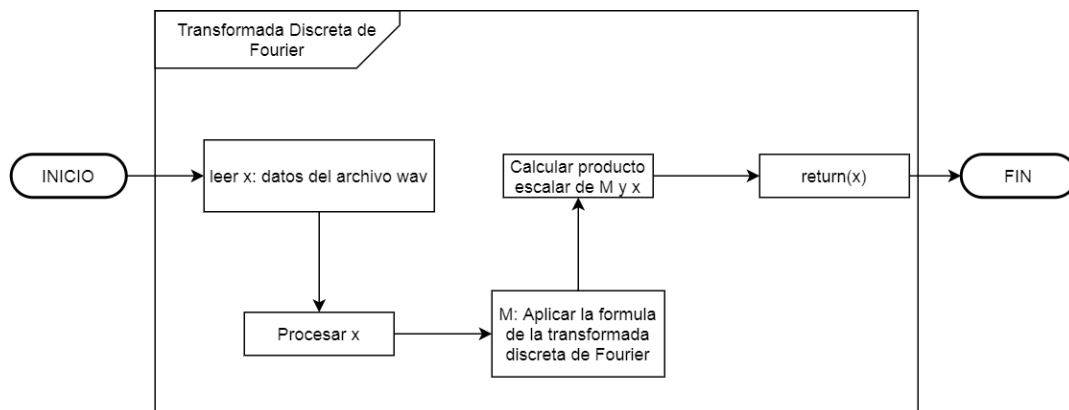


Figura 34. Algoritmo: Transformada Discreta de Fourier
Fuente: (Elaboración propia)

```

1. def dft(x):
2.     x = np.asarray(x, dtype=float)
3.     N = x.shape[0]
4.     n = np.arange(N)
5.     k = n.reshape((N, 1))
6.     M = np.exp(-2j * np.pi * k * n / N)
7.     return np.dot(M, x)
  
```

Listing 1. Transformada discreta de Fourier.

Una vez tenemos calculado la DFT podemos seguir con la FFT.

La FFT se puede calcular de varias maneras, la más clásica es por recursividad, en este caso lo haremos vectorialmente por mejorar el tiempo de ejecución.

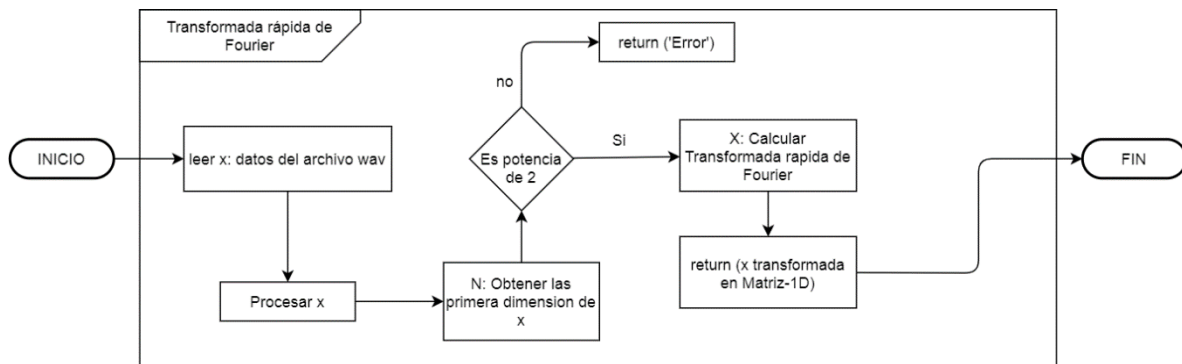


Figura 35. Algoritmo: Transformada Rápida de Fourier

Fuente: (Elaboración propia)

```

1. def fft_v(x):
2.     x = np.asarray(x, dtype=float)
3.     N = x.shape[0]
4.     if np.log2(N) % 1 > 0:
5.         raise ValueError("must be a power of 2")
6.
7.     N_min = min(N, 2)
8.     n = np.arange(N_min)
9.     k = n[:, None]
10.    M = np.exp(-2j * np.pi * n * k / N_min)
11.    X = np.dot(M, x.reshape((N_min, -1)))
12.
13.    while X.shape[0] < N:
14.        X_even = X[:, :int(X.shape[1] / 2)]
15.        X_odd = X[:, int(X.shape[1] / 2):]
16.        terms = np.exp(-1j * np.pi *
17.            np.arange(X.shape[0]) / X.shape[0])[:, None]
18.        X = np.vstack([X_even + terms * X_odd,
19.            X_even - terms * X_odd])
20.    return X.ravel()
  
```

Listing 2. Transformada rápida de Fourier.

En base a la Transformada Rápida de Fourier se implementa La Transformada de Fourier de corto tiempo para determinar el contenido en frecuencia sinusoidal y los cambios con respecto al tiempo de una señal musical.

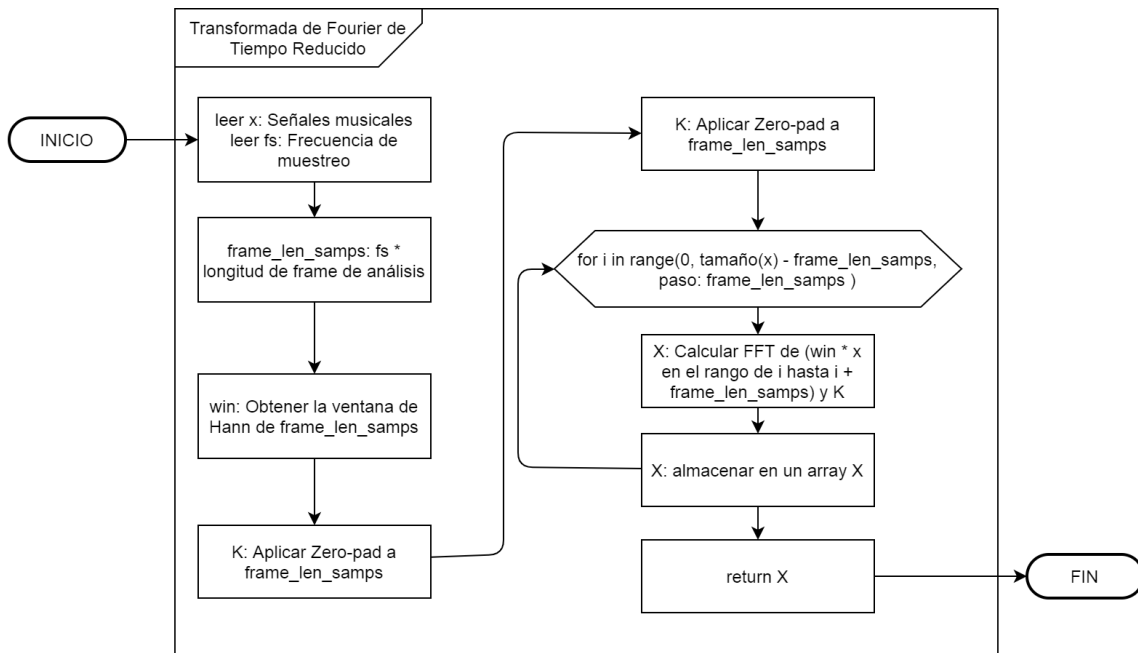


Figura 36. Algoritmo: Transformada de Fourier de Tiempo Reducido

Fuente: (Elaboración propia)

```

1.  def _stft(self, x, fs):
2.      frame_len_samps = int(fs * self._frame_len_sec)
3.      win = get_window(self._window_func, frame_len_samps)
4.
5.      # zero-pad al doble de la longitud del marco frame
6.      K = int(nextpow2(2*frame_len_samps))
7.      X = np.array([np.fft.fft(win*x[i:i+frame_len_samps],
8.      K)
9.      for i in range(0, len(x) -
10.      frame_len_samps, frame_len_samps)])
11.  return X
  
```

Listing 3. Transformada de Fourier de Tiempo Reducido.

3.5.4. COMPUTO DE FRECUENCIAS FUNDAMENTALES

Para el cómputo de frecuencias fundamentales se utiliza las funciones mencionadas en la preparación de datos y la Transformada de Fourier de corto plazo a las señales musicales del archivo WAV.

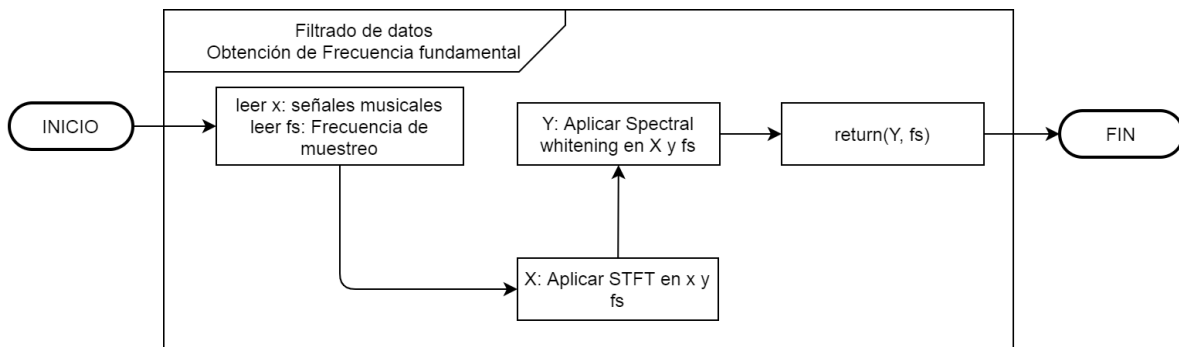


Figura 37. Algoritmo: Obtención de la frecuencia fundamental
Fuente: (Elaboración propia)

Una vez los datos son devueltos se empieza a procesarlos para obtener iterativamente las frecuencias fundamentales en el paso del tiempo. Se procesa la señal “Y”, la frecuencia de muestreo “fs” en un marco de ventana iterativamente.

```
1. def _iterative_est(self, Y, fs):
2.     f0_estimations = []
3.     T = Y.shape[0]
4.     for t in tqdm(range(T)):
5.         # espectro de magnitud residual del marco de análisis
6.         Y_t_R = np.abs(Y[t,:])
7.         # estimaciones de f0 para el marco actual
8.         f0_frame_estimations = []
9.         tau_hat, salience_hat, Y_t_D = self._search_smax(Y_t_R, fs, t
            au_prec=0.5)
10.        f0_frame_estimations.append(fs/tau_hat)
11.        f0_estimations.append(f0_frame_estimations)
12.    return f0_estimations
```

Listing 4. Computo iterativo de frecuencias fundamentales.

3.5.5. FUNCIÓN DE DETECCIÓN DE INICIO

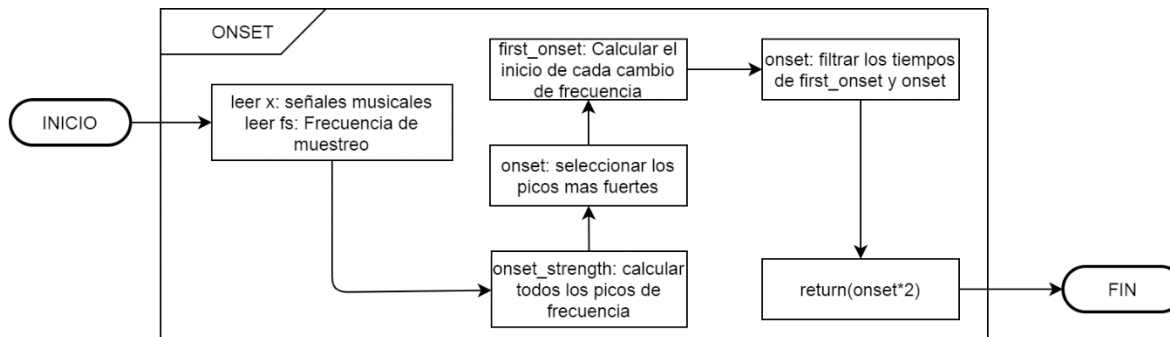


Figura 38. Algoritmo: Detección de Onset

Fuente: (Elaboración propia)

La función de detección de inicio crea marcas en dominio del tiempo para tener una referencia de cuando una nota musical es pulsada.



Figura 39. Pieza musical Perpetual Motion, 5 primeros segundos.

Fuente: (Suzuki, 1998)

Para la introducción de la pieza musical Perpetual motion se tiene los siguientes registros de nota MIDI generado por el cómputo de frecuencia fundamental (sección 3.5.4).

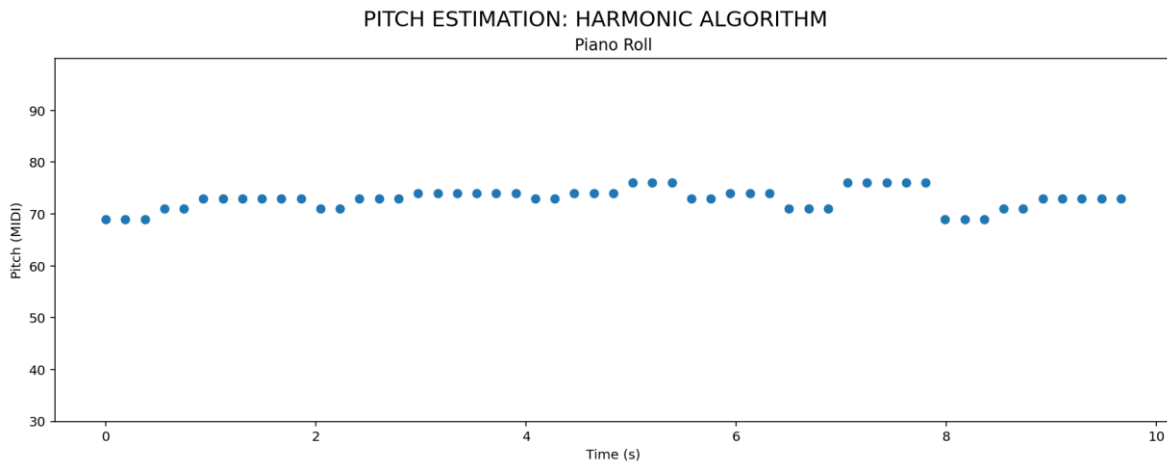


Figura 40. Piano roll de los 5 primeros segundos de la pieza musical Perpetual Motion

Fuente: (Elaboración propia)

De los cuales por cada cambio de frecuencia se entiende como un inicio de nota musical.

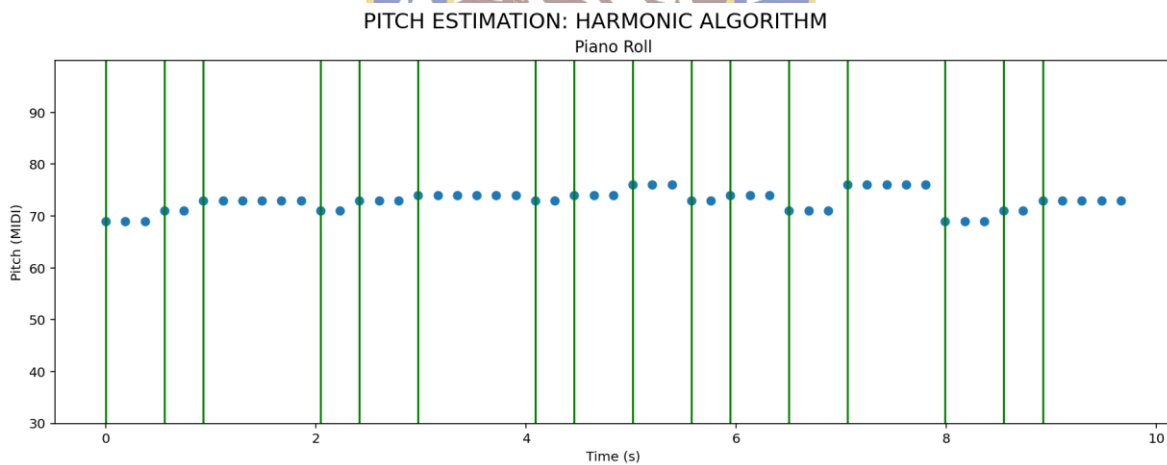


Figura 41. Selección de inicio de frecuencia. Primera estimación de Onset

Fuente: (Elaboración propia)

Paralelamente se analiza los picos de frecuencia en busca de más inicios de nota musical y se filtra con los ya encontrados (Figura 43).

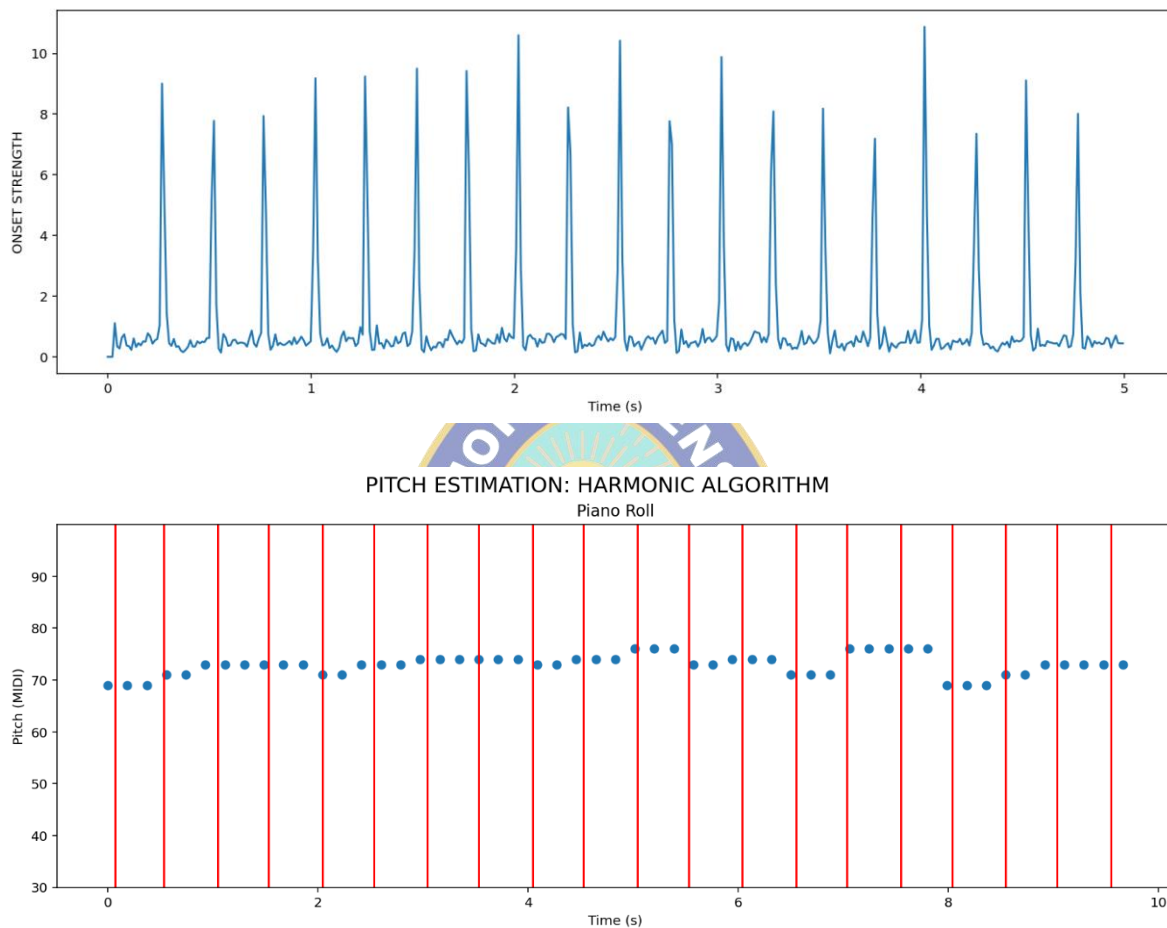


Figura 42. Selección de picos de frecuencia. Segunda estimación de Onset
Fuente: (Elaboración propia)

En la figura 43 (a) se puede observar los picos generados en dominio del tiempo, una vez obtenido los inicios de nota, estos son filtrados mediante 2 factores.

- Compensación de umbral: Decide hasta que rangos en función de Onset Strength se seleccionarán los inicios de nota musical.
- Heurística flexible: Decide si el posible pico generado es un candidato de inicio de nota basado en los picos obtenidos a su alrededor. Esto con el fin de evitar inicios de nota muy unidos.

3.5.6. GENERACIÓN DE LA PARTITURA MUSICAL

Cada nota musical tiene su equivalencia en frecuencia (en Hz), al igual que en MIDI (denotada por un número natural).

Pitch	Frequency [Hz]	MIDI
A4	440.000	69
A#4	466.164	70
B4	493.883	71
C5	523.251	72
C#5	554.365	73
D5	587.330	74
D#5	622.254	75
E5	659.255	76
F5	698.457	77
F#5	739.989	78
G5	783.991	79
G#5	830.609	80
A5	880.000	81

Figura 43. Equivalencia Nota Musical - Frecuencia - Nota Midi
Fuente: (Elaboración propia)

Las frecuencias fundamentales obtenidas. Primeramente, son transformados de Hz a notación MIDI mediante la siguiente formula.

$$m = 12 * \log_2 \left(\frac{f_m}{440 \text{ Hz}} \right) + 69$$

Para la conversión se utiliza un valor en común como la nota A4 cuyo valor en frecuencia es 440 Hz y valor en nota MIDI es 69.

Después de obtener la nota MIDI, y teniendo el inicio de cada nota musical, se empieza a armar el pentagrama musical.

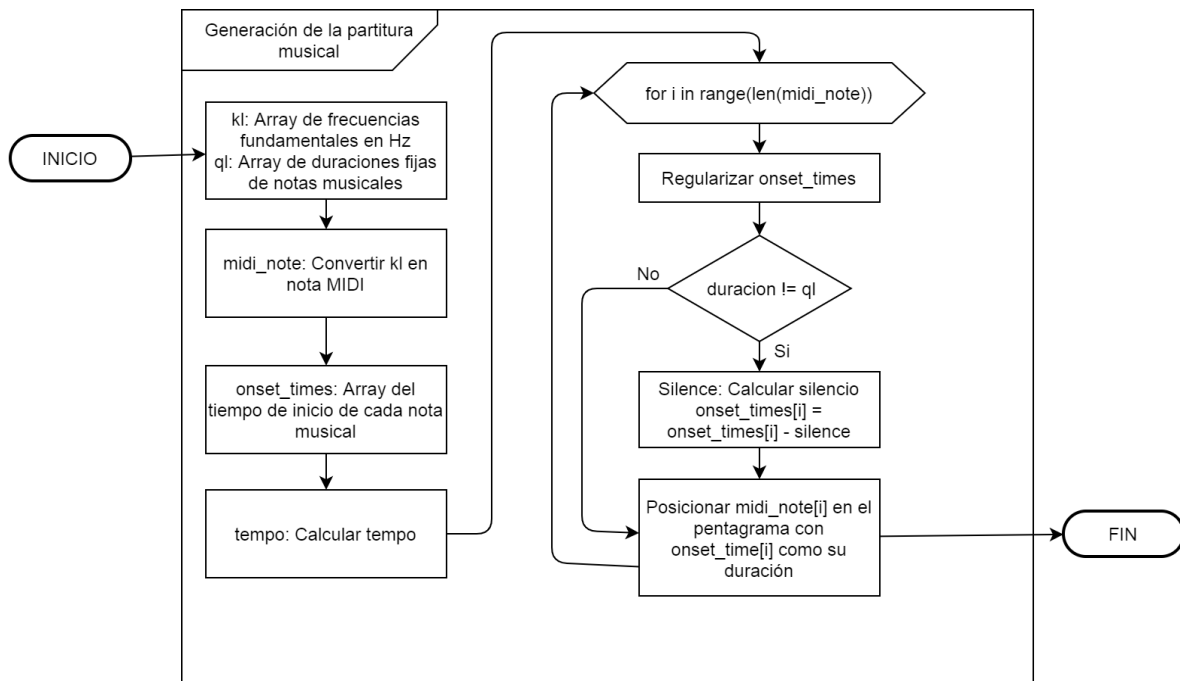


Figura 44. Diagrama de flujo: Generación de la partitura Musical

Fuente: (Elaboración propia)

3.5.7. IMPLEMENTACIÓN DE ARCHIVO MIDI

Un archivo MIDI contiene todos los aspectos musicales.

La librería music21 que ayuda a armar la partitura musical posee una directa conversión de partitura a archivo MIDI. Dado una partitura “s” se exporta de la siguiente manera.

```

1. s = music21.stream.Stream()
2. #MODULO DE CREACION DE PARTITURA
3. #...
4.
5. s.write(fp = output + 'outputfile.png', fmt = 'musicxml.png')
6. s.write("midi", output + "music21.mid")
  
```

Listing 5. Partitura exportada en formato png y midi.

3.6. IMPLEMENTACIÓN

Se creó un prototipo el cual recibe un filepath y ejecuta el algoritmo para convertirlo a partitura.

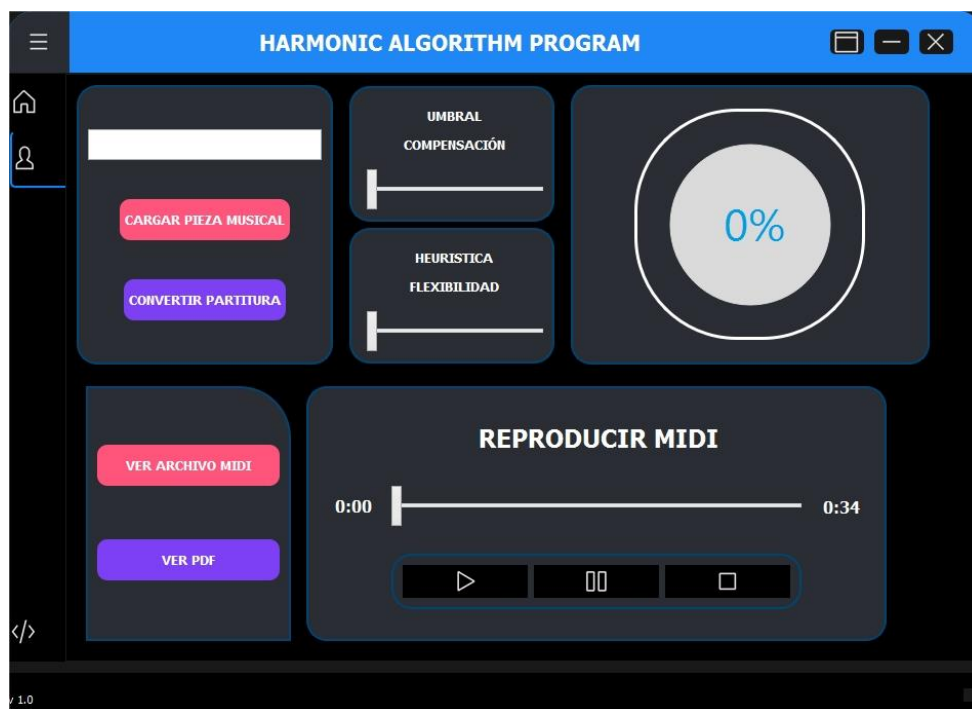


Figura 45. Prototipo pantalla de operaciones
Fuente: (Elaboración propia)

El primer recuadro es para cargar la pieza musical en formato wav.

Una vez elegida la pieza musical factores de Compensación de umbral y Heurística (explicados en 3.5.5) deben ser elegidos, caso contrario se tomara en cuenta los valores por defecto 0.5 y 3 respectivamente.

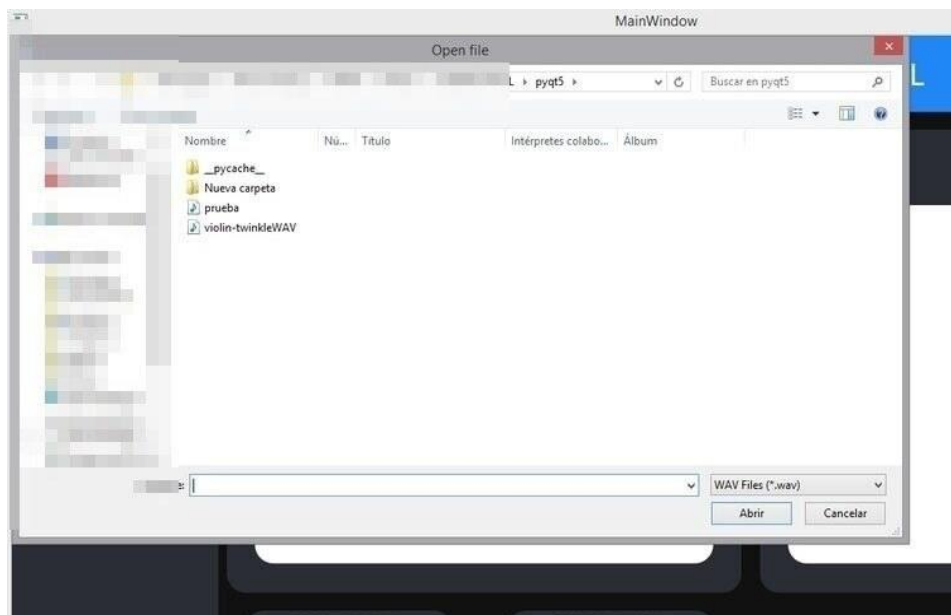


Figura 46. Prototipo carga de archivo wav
Fuente: (Elaboración propia)

Se carga la pieza musical en formato wav.

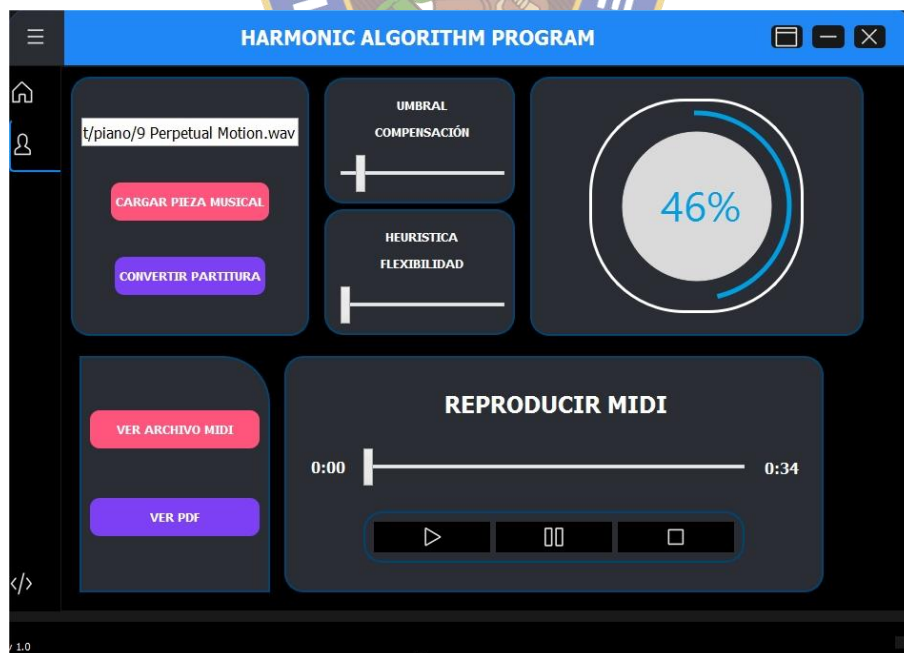


Figura 47. Prototipo, ejecución del botón convertir partitura
Fuente: (Elaboración propia)

El botón “convertir partitura” genera 2 resultados, el primero es la representación en formato MIDI de la partitura, el segundo es la partitura PDF con sus respectivos aspectos musicales, como ser la tonalidad en la que está la pieza musical con su respectiva armadura, el tempo y las notas musicales con su respectiva duración.

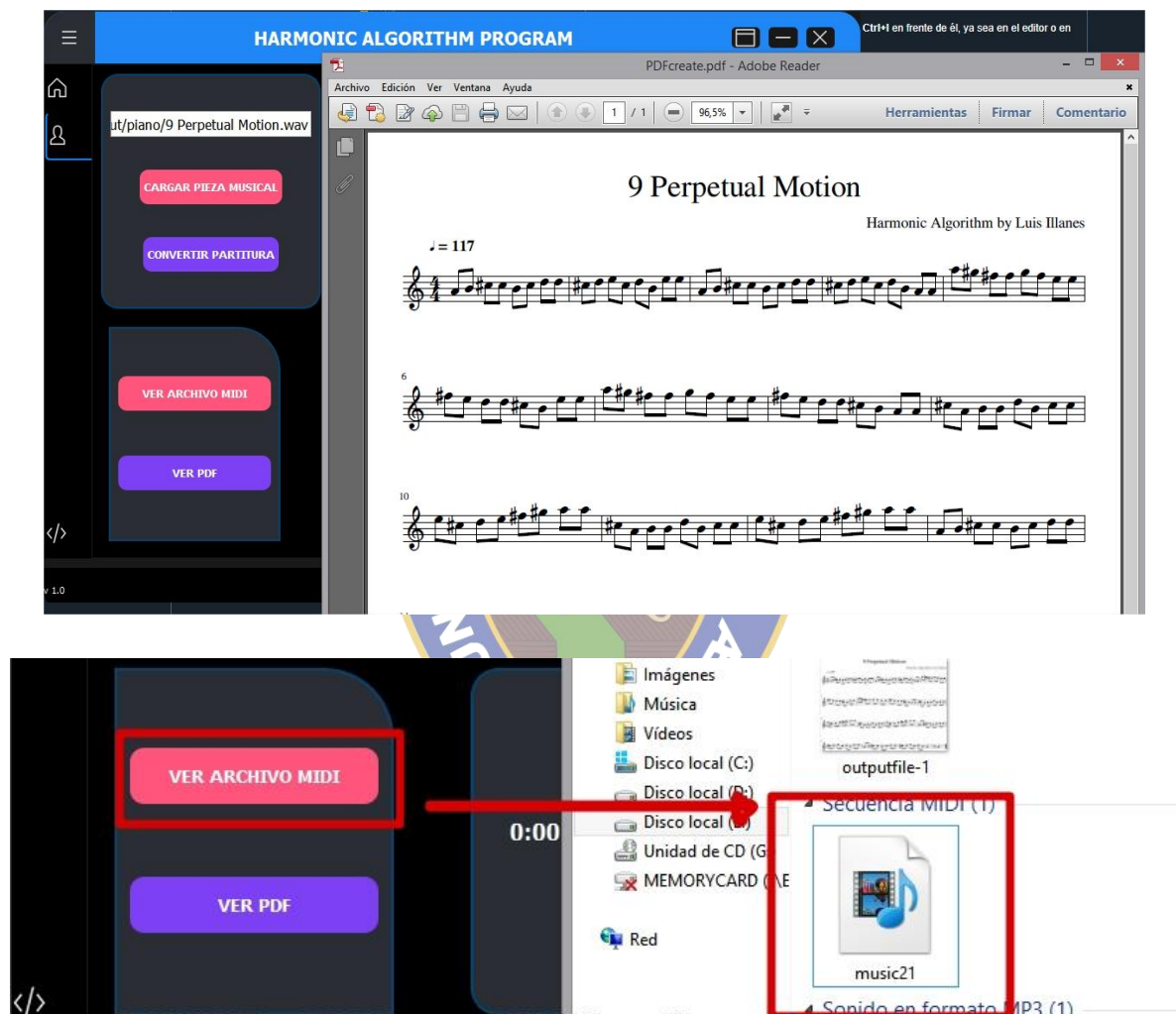


Figura 48. Prototipo, Resultados
Fuente: (Elaboración propia)

CAPÍTULO 4. RESULTADOS Y ANÁLISIS

4.1. RESULTADOS

Para los resultados se seleccionó las piezas musicales pertenecientes al libro Suzuki Violin School, Vol 1 puesto que es un libro muy conocido y de renombre. Las piezas musicales son:

- Twinkle Twinkle , Little Star Shinichi Suzuki
- Lightly Row Folk Song
- Go tell aunt Rhody Folk Song
- O Come, Little Children Folk Song
- Long, Long Ago T. H. Bayly
- Perpetual Motion Shinichi Suzuki
- Andantino Shinichi Suzuki
- Minuet No. 1 J. S. Bach
- Minuet No. 2 J. S. Bach
- Minuet No. 3 J. S. Bach
- The Happy Farmer R. Schumann R. Schumann



Las correspondientes piezas musicales fueron probadas en 3 distintos instrumentos musicales monofónicos con distinto timbre. Siendo así un total de 33 pruebas. Cada pieza musical fue grabada en formato WAV, 16 bits a 44,1 kHz. (Ver resultados en Anexo C)

INSTRUMENTO	TIPO DE INSTRUMENTO	N.º DE PRUEBAS
VIOLIN	Cuerda Frotada	11
CLARINETE	Viento-Madera	11
PIANO	Cuerda Percutida	11

Tabla 2. Instrumentos para el análisis

El piano puesto a pesar de que es un instrumento polifónico también puede ejecutar canciones monofónicas. En el caso del clarinete para efectos de prueba se utilizó bajo la afinación de si bemol.

Se aplicaron las métricas propuestas por Music Information Retrieval que consisten en el cálculo del valor-F para estimar los porcentajes de recuperación de información basada en:

- Onset: Detección de inicio de nota musical, sin tomar en cuenta la cantidad de notas estimadas y su duración.
- Evaluación de transcripción: Detección de la frecuencia de nota musical, duración y ubicación en el pentagrama.

4.1.1. DETECCIÓN DE INICIO DE NOTA MUSICAL

Se calcula la precisión, exhaustividad y valor-F de cada pieza musical basada en el inicio de nota musical u Onset, con el fin de determinar la media de precisión cubriendo el sesgo que existe en el producto final.

PIEZA MUSICAL	PRECISIÓN	EXHAUSTIVIDAD	VALOR-F
Twinkle Twinkle , Little Star	1	0.833	0.909
Lightly Row	0.882	0.789	0.833
Go tell aunt rhody	0.919	0.983	0.950
O Come, Little Children	0.773	0.773	0.773

Long, Long Ago	0.857	0.873	0.865
Perpetual Motion	0.966	0.890	0.926
Andantino	0.703	0.847	0.769
Minuet No. 1	0.907	0.599	0.721
Minuet No. 2	0.944	0.876	0.909
Minuet No. 3	0.973	0.938	0.955
The Happy FarmerR. Schumann	0.990	0.894	0.939

Tabla 3. Metricas: Recuperación de inicio de nota musical, Violin.

De la tabla se puede interpretar que para cada pieza musical la precisión varia en rangos grandes dando una media de 90% de precisión y esto se debe a que las piezas musicales fueron interpretadas en un instrumento de cuerda frotada que genera muchos armónicos el cual crea bastantes falsos negativos (inicio de nota musical donde no debería de existir) y se puede ver reflejado en los porcentajes restante de la exhaustividad que dieron una media de 85%. Un buen porcentaje de las notas referencia de cada pieza musical fue recuperado correctamente con una media de exactitud balanceada de 87%.

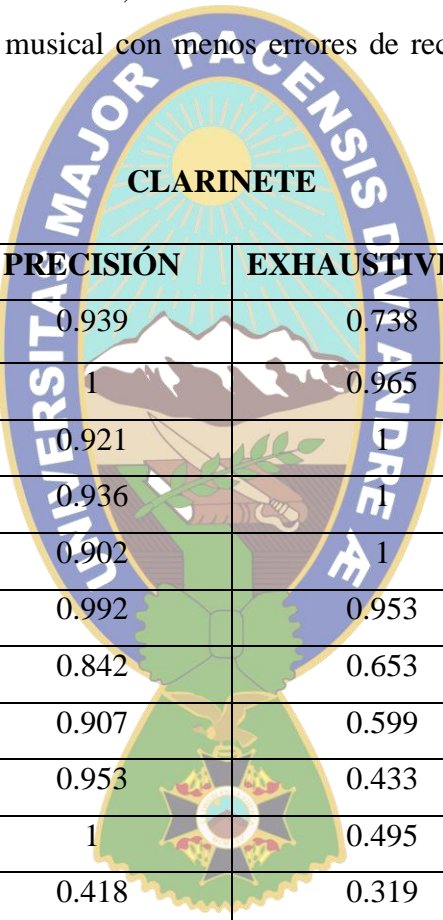
PIANO

PIEZA MUSICAL	PRECISIÓN	EXHAUSTIVIDAD	VALOR-F
Twinkle Twinkle, Little Star	0.976	0.976	0.976
Lightly Row	0.982	0.982	0.982
Go tell aunt rhody	0.983	0.983	0.983
O Come, Little Children	0.886	0.886	0.886
Long, Long Ago	0.964	0.964	0.964
Perpetual Motion	1	1	1
Andantino	0.875	1	0.933
Minuet No. 1	0.957	0.957	0.957
Minuet No. 2	0.934	0.934	0.934

Minuet No. 3	0.995	0.995	0.995
The Happy FarmerR. Schumann	0.995	1	0.998

Tabla 4. Métricas: Recuperación de inicio de nota musical, Piano.

El piano al ser un instrumento parte de la familia de cuerda percutida los picos de frecuencia de cada nota musical es claro haciendo que la detección de inicio de nota musical sea casi exacta. La precisión media es de 95%, con una exhaustividad media de 97% y exactitud de 96%, siendo el instrumento musical con menos errores de recuperación de inicio de nota musical.



PIEZA MUSICAL	PRECISIÓN	EXHAUSTIVIDAD	VALOR-F
Twinkle Twinkle, Little Star	0.939	0.738	0.827
Lightly Row	1	0.965	0.982
Go tell aunt rhody	0.921		0.959
O Come, Little Children	0.936		0.967
Long, Long Ago	0.902	1	0.948
Perpetual Motion	0.992	0.953	0.972
Andantino	0.842	0.653	0.736
Minuet No. 1	0.907	0.599	0.721
Minuet No. 2	0.953	0.433	0.596
Minuet No. 3	1	0.495	0.662
The Happy FarmerR. Schumann	0.418	0.319	0.362

Tabla 5. Métricas: Recuperación de inicio de nota musical, Clarinete.

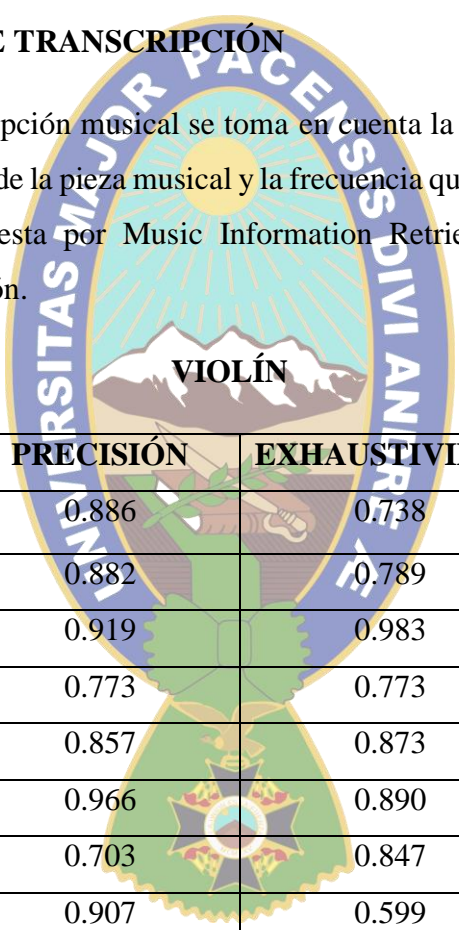
Al igual que el violín, los instrumentos de viento-madera como lo es el clarinete generan muchos armónicos. Se obtuvo una precisión media de 89% con una exhaustividad media de 74% y exactitud de 79%. Se puede observar que a medida que se va avanzando con piezas musicales que tienen un ritmo mayor existe mayor generación de falsos negativos.

INSTRUMENTO	PRECISIÓN
VIOLIN	90%
CLARINETE	89%
PIANO	95%

Tabla 6. Media de precisión obtenida del inicio de nota musical por instrumento

4.1.2. EVALUACIÓN DE TRANSCRIPCIÓN

En la evaluación de transcripción musical se toma en cuenta la duración de la nota musical con respecto al tiempo total de la pieza musical y la frecuencia que tiene. Se aplica las mismas métricas de valor-F propuesta por Music Information Retrieval para la estimación de recuperación de transcripción.



PIEZA MUSICAL	PRECISIÓN	EXHAUSTIVIDAD	VALOR-F
Twinkle Twinkle, Little Star	0.886	0.738	0.805
Lightly Row	0.882	0.789	0.789
Go tell aunt rhody	0.919	0.983	0.950
O Come, Little Children	0.773	0.773	0.773
Long, Long Ago	0.857	0.873	0.865
Perpetual Motion	0.966	0.890	0.926
Andantino	0.703	0.847	0.769
Minuet No. 1	0.907	0.599	0.721
Minuet No. 2	0.940	0.871	0.904
Minuet No. 3	0.973	0.938	0.955
The Happy FarmerR. Schumann	0.728	0.657	0.691

Tabla 7. Métricas: Recuperación de transcripción musical, Violín.

Una media de precisión de 86% fue recuperado correctamente. Una media de exhaustividad de 81% dando menos falsos negativos con una exactitud de 83%.

PIANO

PIEZA MUSICAL	PRECISIÓN	EXHAUSTIVIDAD	VALOR-F
Twinkle Twinkle, Little Star	0.976	0.976	0.976
Lightly Row	0.982	0.982	0.982
Go tell aunt rhody	0.983	0.983	0.983
O Come, Little Children	0.886	0.886	0.886
Long, Long Ago	0.964	0.964	0.964
Perpetual Motion	1	1	1
Andantino	0.848	0.969	0.905
Minuet No. 1	0.957	0.957	0.957
Minuet No. 2	0.870	0.870	0.870
Minuet No. 3	0.995	0.995	0.995
The Happy FarmerR. Schumann	0.853	0.856	0.855

Tabla 8. Métricas: Recuperación de transcripción musical, Piano.

Una media de precisión de 93% fue recuperado correctamente. Una media de exhaustividad de 94% dando menos falsos negativos con una exactitud de 93%.

CLARINETE

PIEZA MUSICAL	PRECISIÓN	EXHAUSTIVIDAD	VALOR-F
Twinkle Twinkle, Little Star	0.939	0.738	0.827
Lightly Row	0.945	0.912	0.929
Go tell aunt rhody	0.857	0.931	0.893
O Come, Little Children	0.840	0.898	0.868
Long, Long Ago	0.836	0.927	0.879

Perpetual Motion	0.943	0.906	0.924
Andantino	0.829	0.643	0.724
Minuet No. 1	0.907	0.599	0.721
Minuet No. 2	0.953	0.433	0.596
Minuet No. 3	0.9	0.495	0.662
The Happy FarmerR. Schumann	0.418	0.319	0.362

Tabla 9. Métricas: Recuperación de transcripción musical, Clarinete.

Una media de precisión de 85% con una exhaustividad media de 70% dando menos falsos negativos, y una exactitud media de 76%.

INSTRUMENTO	PRECISIÓN
VIOLIN	86%
CLARINETE	85%
PIANO	93%

Tabla 10. Media de precisión obtenida de la transcripción musical por instrumento

4.2. PRUEBA DE LA HIPÓTESIS

Para la prueba de la hipótesis “El algoritmo basado en el análisis armónico aplicado en señales musicales obtiene un espectro de frecuencias el cual genera la partitura de la pieza musical con una precisión del 85%” se siguió los siguientes pasos.

1. Se calculó la precisión de recuperación de información en base al inicio de nota musical de cada pieza musical comparado con su respectiva referencia.
2. Se calculó la precisión de recuperación de información en base a la duración de la nota musical, su frecuencia y ubicación de cada pieza musical comparado con su respectiva referencia.

3. Se evaluaron los demás aspectos musicales obtenidos a partir de las notas ya generadas en partitura.

En base a los cálculos obtenidos se puede afirmar que:

- La detección de inicio de nota musical se llegó a obtener una media de porcentaje de precisión de 91% dejando de lado la cantidad de notas detectadas y su duración.
- La detección de duración, frecuencia y ubicación de la nota musical en el pentagrama se llegó a obtener una media de porcentaje de precisión de 88%. A diferencia del anterior punto este representa un porcentaje menor ya que evalúa el pentagrama con todos sus aspectos musicales.

Por lo tanto, bajo los 2 puntos expuestos se puede afirmar que el Algoritmo para la detección de la partitura de una pieza musical basado en el análisis armónico genera la partitura de una pieza musical con una precisión mayor al 85%.

4.3. ANÁLISIS A LA COMPLEJIDAD ALGORITMICA

En el capítulo 3 se vió que el algoritmo se aborda mediante modulos los cuales en conjunto arman el programa principal.

Para el calculo de la complejidad algorítmica se utilizó la notación Big O con notación asintótica aplicada a los modulos mostrados en el 3.5.X. (Ver código completo en Anexo E).

Se midió teóricamente el tiempo de ejecución y la complejidad algorítmica de cada modulo para así hallar uno general del programa principal.

MODULO	TIEMPO DE EJECUCIÓN
Transformada de Fourier de corto plazo	$T(n) = W * n \log(n) + 3$
Blanqueamiento Espectral	$T(n) = 2n^2 + 15n + 9$
Computo de frecuencia fundamental	$T(n) = (2 + 15n)(20 + 12F + 8H)$
Detección de inicio de nota	$T(n) = 11n + 3$
Generación de la partitura	$T(n) = 16n^2 + 14n + 38$

Tabla 11. Tiempos de ejecución de cada módulo

MODULO	COMPLEJIDAD ALGORITMICA
Transformada de Fourier de corto plazo	$O(W * n \log(n))$
Blanqueamiento Espectral	$O(n^2)$
Computo de frecuencia fundamental	$O((F + H) * n)$
Detección de inicio de nota	$O(n)$
Generación de la partitura	$O(n^2)$

Tabla 12. Complejidad Algoritmica de cada módulo

Como se puede observar se evidencia los tiempos de ejecución teórico y la complejidad algorítmica de cada módulo, el programa principal es una ejecución secuencial de cada uno por lo que se obtiene una complejidad algorítmica de $O(n^2)$ siguiendo la notación asintótica.

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

Finalmente se concluye para cada objetivo específico.

- Reducir la complejidad de la implementación algorítmica para obtener la partitura de una pieza musical en tiempos óptimos.

En el capítulo 4 se presenta un análisis a la complejidad del algoritmo empezando por la implementación de Fourier, la generación de picos de frecuencia e hipótesis de frecuencias fundamentales, y finalmente en el procesamiento de figuras musicales a partir de notas MIDI.

- Realizar un módulo de pre procesamiento al espectro de frecuencias para la detección de frecuencias fundamentales.

Se realizó un módulo el cual procesa los datos de entrada ventana por ventana para generar correctamente los picos de frecuencia en el paso del tiempo, esto con el fin de analizar correctamente cada frame y así lograr que notas como la semicorchea sean tomados en cuenta.

- Diseñar un módulo basado en las frecuencias obtenidas que detecte el tempo de una pieza musical.

Se realizó un módulo aplicado a los datos de entrada el cual devuelve el tempo en BPM.

- Generalizar el rango de frecuencias de la pieza musical mediante índices de Fourier

Se estableció los rangos de frecuencia en índices de Fourier mínimo y máximo lo más óptimo mediante la supresión del timbre musical para que el algoritmo pueda ser aplicado a instrumentos cuyo rango de notas sean equiparables al violín.

- Realizar un módulo de post procesamiento para detectar la tonalidad en la cual se está interpretando la pieza musical.

Se realizó un módulo que detecta la tonalidad mediante las notas musicales ya detectadas generalizando sus alteraciones.

Así se llega a cumplir el objetivo principal de diseñar un algoritmo basado en el análisis armónico aplicado a señales musicales para obtener un espectro de frecuencias del cual se podrá generar la partitura de la pieza musical, ya que detecta correctamente los aspectos musicales como ser notas musicales, compas con armadura, tempo y accidentes. Bajo las justificaciones establecidas. El presente algoritmo sirve como basé para aquellas personas que inician una transcripción musical, puesta que está abierto a fallas en la detección de notas musicales.

En base a las pruebas realizadas al libro de Suzuki Violin School, Vol 1 la hipótesis no se puede rechazar por lo tanto se acepta que el algoritmo basado en el análisis armónico aplicado en señales musicales obtiene un espectro de frecuencias el cual genera la partitura de la pieza musical con una precisión del 85%.

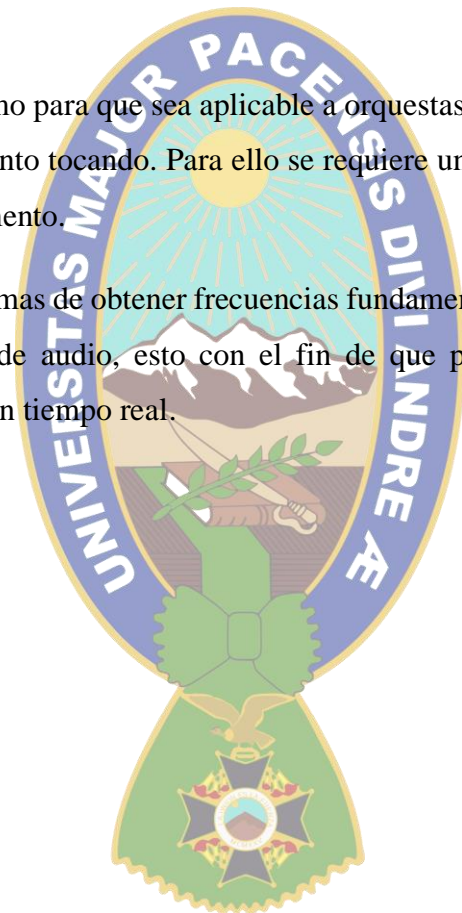
Asi mismo se concluye por cada instrumento musical.

- Violin: Siendo un instrumento que genera bastantes armónicos rebaja la precisión de transcripción musical, aun asi la detección de notas musicales e inicio de nota no se ven afectados
- Piano: El piano al ser un instrumento que genera notas semi puras, el algoritmo no tiene problemas con detectar las notas musicales y su inicio en el pentagrama.
- Clarinete: El clarinete siendo un instrumento de viento, tiene un timbre que hace que sea fácil detectar las notas musicales al igual que el piano, pero también genera falsos inicios de notas lo que hace que rebaje la precisión.

5.2. RECOMENDACIONES

Una vez se tienen las conclusiones del proyecto conociendo sus límites y alcances, se plantean recomendaciones para continuar con la línea de investigación.

- Diseñar un procesamiento extra para detectar múltiples tonos secuencialmente en distintos compases musicales, con el fin de aplicar a piezas musicales polifónicas tocadas en piano.
- Extender el algoritmo para que sea aplicable a orquestas completas los cuales tengan más de un instrumento tocando. Para ello se requiere un módulo que separe la pieza musical por instrumento.
- Estudiar nuevas formas de obtener frecuencias fundamentales de piezas musicales en distintos formatos de audio, esto con el fin de que pueda ser aplicable a piezas musicales tocadas en tiempo real.



CAPÍTULO 6. BIBLIOGRAFÍA

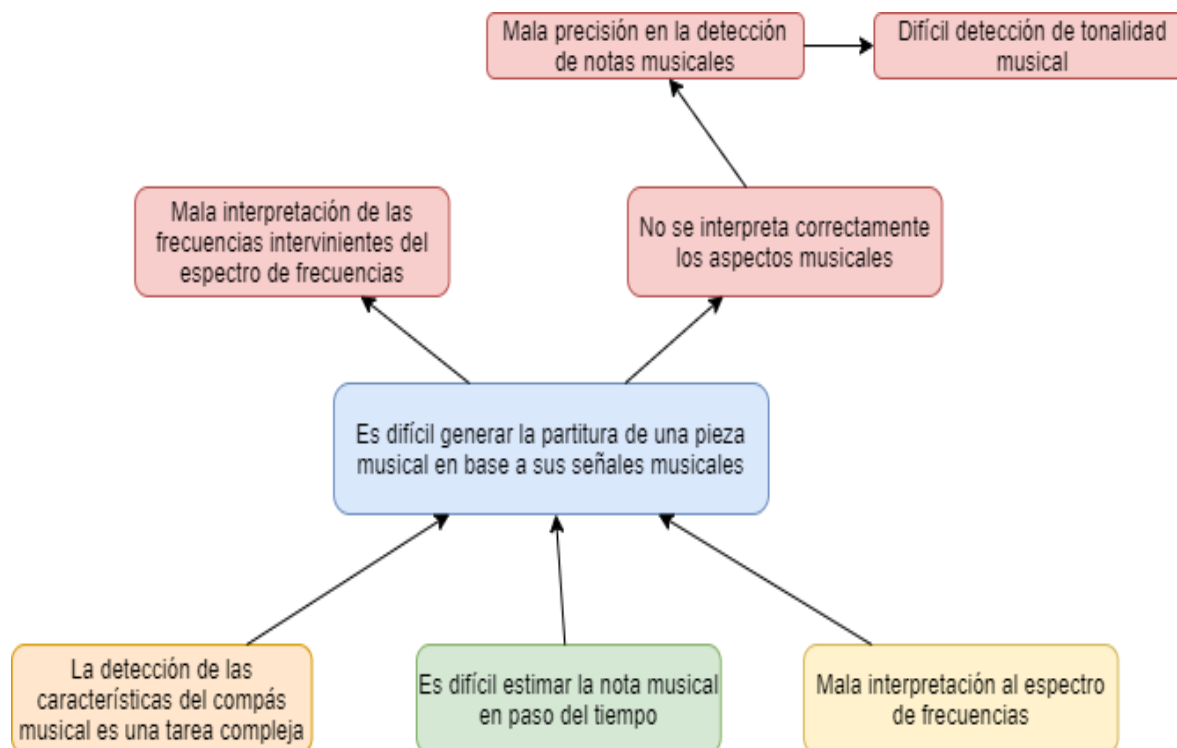
- Alarcon, J. (2016) Rendimiento de algoritmos y notación Big-O. s.l.
- Bainbridge, D. y Bell, Tim. (2003). A music notation construction engine for optical music recognition. s.l.
- Cwitkowitz, F. (2019). End-to-end Music Transcription Using Fine-Tuned Variable-Q Filterbanks. United States: New York.
- Crossley, P. (2002). Rhythm music. California, Los Angeles
- Dubey, A. (2018). Fourier Transformation and Its Mathematics. s.l.
- Degani, A. (2014). Harmonic analysis for music transcription and characterization. Italia: Brescia.
- Dobre, y Negrescu (2017). Automatic music transcription software based on constant Q transform. Romania: Ploiestí.
- Epperson, G. (2019). Music [En línea] <https://www.britannica.com/art/music> [Consultado 28 de julio de 2020]
- Fuentes, C. (2018). Notación Big O. s.l.
- Goto, M. (2006). Music scene description. In Signal Processing Methods for Music Transcription, pages 327–359. Springer.
- Goto, M. and Muraoka Y. (1999). Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions.
- Kashino, K. (2006). Auditory scene analysis in music signals. In Signal Processing Methods for Music Transcription, pages 299–325.
- Koontz, W. (2016). Introduction to Audio Signal Processing. s.l.

- Klapuri, A. (2001) Multipitch estimation and sound separation by the spectral smoothness principle. In IEEE International Conference on Acoustics, Speech, and Signal processing. s.l.
- Klapuri, A. (2003). Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. IEEE Transactions on Speech and Audio Processing.
- Klapuri, A. (2004). Signal Processing Methods for the Automatic Transcription of Music. Finlandia, Tampere.
- Klapuri, A. (2006). Introduction to music transcription. In Signal Processing Methods for Music Transcription.
- Kunieda, N., Shimamura T., and Suzuki J. (1996). Robust method of measurement of fundamental frequency by ACLOS: autocorrelation of log spectrum. In IEEE International Conference on Acoustics, Speech, and Signal Processing.
- Laroche, J. (2003). Efficient tempo and beat tracking in audio recordings. Journal of the Audio Engineering Society
- Makhmutov, Brown, Mazara y Johard (2016). MOMOS-MT: Mobile Monophonic Systemfor Music Transcription. s.l.
- Mila, T. (2019). Digital Signal Processing Fundamentals. s.l.
- Moorer, J. (1977). On the transcription of musical sound by computer.
- Naranjo, C. (2013) El pentagrama y sus componentes (símbolos). s.l.
- Nave, R. (1999). Fundamental and Harmonics. s.l.
- Orlandini, L. (2012). La interpretación musical. Chile: Santiago de Chile.

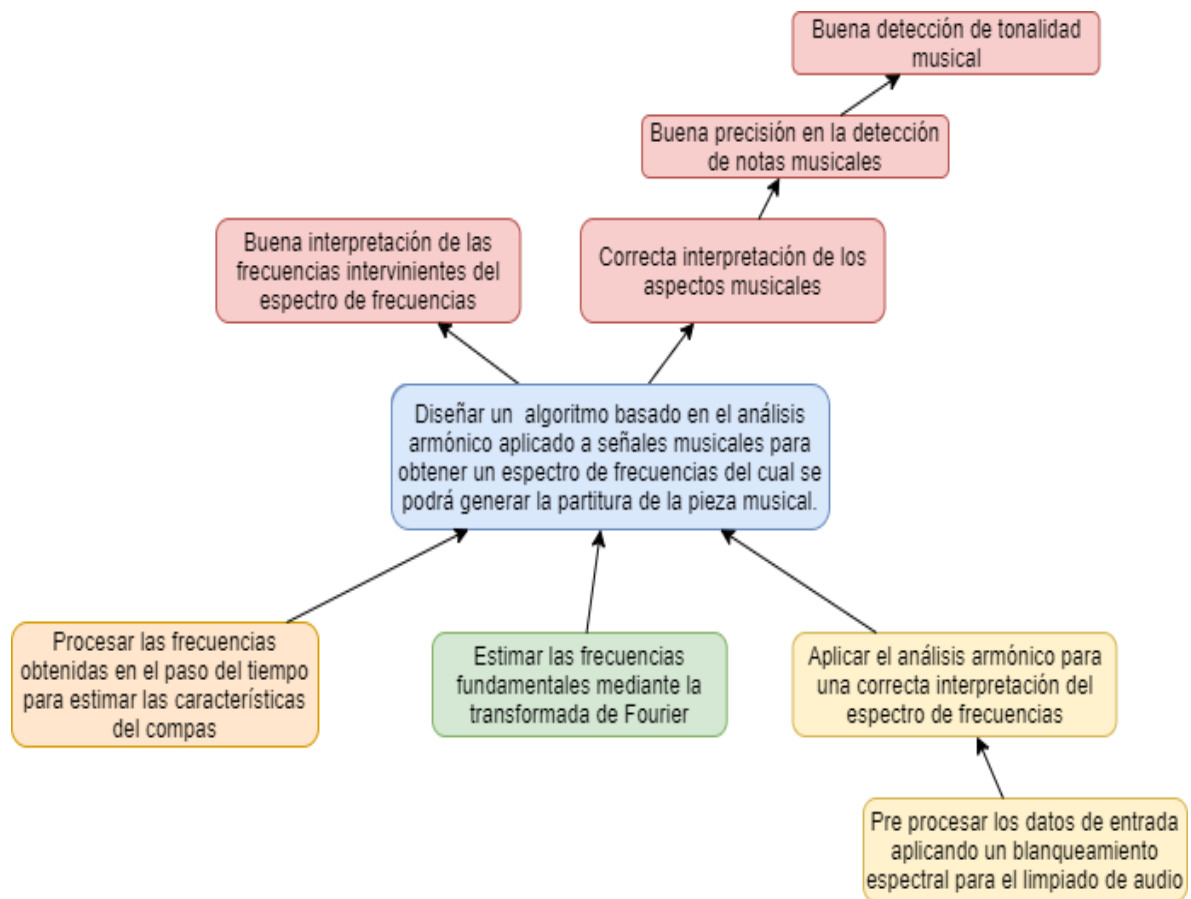
- Pertusa, A. e Iñesta, J. (2008). Multiple fundamental frequency estimation using Gaussian smoothness. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008), pages 105–108.
- Quenneville, D. (2018). Automatic music transcription. United States: Vermont.
- Rivera, A. (2020) Ritmo, pulso y tempo: definición y diferencias. s.l.
- Smith, L., Campbell O., Goldstein M., Sudara. Music information retrieval [en linea] <https://musicinformationretrieval.com/> [Consultado 28 de julio de 2020]
- Smith, S. (2002). Digital Signal Processing: A Practical Guide for Engineers and Scientists. s.l.
- Tomás, M. (2015). Transformada de Fourier - Representación de señales de sonido. Argentina: Bahía Blanca.
- Talkin, D. (1995) A robust algorithm for pitch tracking (RAPT). Speech Coding and Synthesis, pages 495 – 518.
- Weinberger, N. (2004). Music and the brain. Scientific American.
- Weihs, C., Jannach, D., Vatulkin, I., and Rudolph, G. (2016). Music data analysis: Foundations and applications. Chapman and Hall/CRC

ANEXOS

ANEXO A – ÁRBOL DE PROBLEMAS



ANEXO B – ÁRBOL DE OBJETIVOS



ANEXO C – RESULTADOS OBTENIDOS

Primeros 5 segundos de las primeras 5 piezas musicales.

Twinkle Twinkle, Little Star. (Shinichi Suzuki)

ORIGINAL



VIOLIN

1 violin-twinkle



CLARINETE

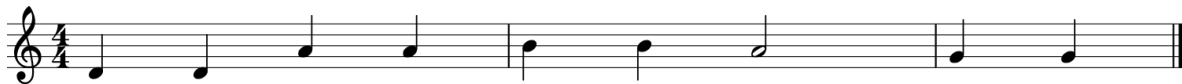
1 clarinete-twinkle



PIANO

1 piano-twinkle

♩ = 60



Lightly Row. (Folk Song)

ORIGINAL

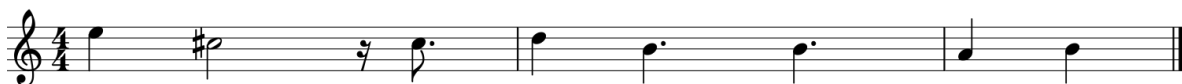
♩ = 120



VIOLIN

2 Lightly_Row

♩ = 135



CLARINETE

2 Lightly_Row

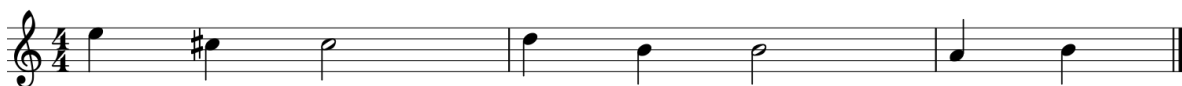
♩ = 151



PIANO

2 Lightly_Row

♩ = 60



Go tell aunt rhody (Folk Song)

ORIGINAL

♩ = 120



VIOLIN

4 Go_Tell_Aunt_Rhody_Violin

♩ = 117



CLARINETE

4 Go_Tell_Aunt_Rhody



PIANO

4 Go_Tell_Aunt_Rhody_Violin



O Come, Little Children (Folk Song)

ORIGINAL



VIOLIN

5 O_Come_Little_Children

♩ = 123



CLARINETE

5 O_Come_Little_Children

♩ = 117



PIANO

5 O_Come_Little_Children

♩ = 117



Long, Long Ago (T. H. Bayly)

ORIGINAL



VIOLIN

7 1 Long_Long_Ago



CLARINETE

7 1 Long_Long_Ago



PIANO

7 1 Long_Long_Ago



ANEXO E - IMPLEMENTACIÓN DEL ALGORITMO

E.1. PITCH: MAIN (pitch.py)

```
from abc import ABC, abstractmethod
from itertools import zip_longest
import numpy as np

from harmonic_algorithm.pitch_onset.kl import PitchGenerate

class Pitch(ABC):
    def __init__(self, signal, sampleRate, frameSize=2048, fftBins=4096):
        self.signal = signal
        self.sampleRate = sampleRate
        self.frameSize = frameSize
        self.fftSize = fftBins

    def __call__(self, signal=None):
        if signal is not None:
            return self._func(signal)
        pitch = []
        start, stop = 0, self.frameSize
        while stop <= self.signal.size:
            f0 = self._func(self.signal[start:stop])
            pitch.append(f0)
            start, stop = stop, stop + self.frameSize
        return np.array(pitch)

    @abstractmethod
    def _func(self, signal):
        return 0

class MonoPitch(Pitch):
    def __init__(self, signal, sampleRate, frameSize=2048, fftBins=4096, **kwargs):
        super().__init__(signal, sampleRate, frameSize, fftBins)
        self.estimate_f0 = None

        poly = kwargs.pop('max_polyphony', 1)
        self.estimate_f0 = PitchGenerate(max_poly=poly)
```

```

def __call__(self, signal=None):
    f0 = self.estimate_f0(self.signal, self.sampleRate)
    return np.array(list(zip_longest(*f0, fillvalue=np.nan)))
)

def _func(self, signal):
    return []

```

E.2. PITCH: OBTENER FRECUENCIAS FUNDAMENTALES (kl.py)

```

from __future__ import division
import numpy as np
from scipy.signal import get_window
from tqdm import tqdm

def nextpow2(x):
    return 2**np.ceil(np.log2(x))

class PitchGenerate(object):

    # lista ordenada de frecuencias utilizadas para encontrar el
    # tono más cercano

    # las frecuencias oscilan entre 65 Hz y 2100 Hz.
    frequencies = np.array([2**(n/12.0)*440 for n in range(-
33,28)])

    # lista ordenada de nombre de tono y octavas, que
    # corresponden al

    notes = [
        {'pname': 'C', 'oct': 2},
        {'pname': 'C#', 'oct': 2},
        {'pname': 'D', 'oct': 2},
        {'pname': 'D#', 'oct': 2},
        {'pname': 'E', 'oct': 2},
        {'pname': 'F', 'oct': 2},
        {'pname': 'F#', 'oct': 2},
        {'pname': 'G', 'oct': 2},
        {'pname': 'G#', 'oct': 2},
        {'pname': 'A', 'oct': 2},
        {'pname': 'A#', 'oct': 2},

```

```

{'pname': 'B', 'oct': 2},
{'pname': 'C', 'oct': 3},
{'pname': 'C#', 'oct': 3},
{'pname': 'D', 'oct': 3},
{'pname': 'D#', 'oct': 3},
{'pname': 'E', 'oct': 3},
{'pname': 'F', 'oct': 3},
{'pname': 'F#', 'oct': 3},
{'pname': 'G', 'oct': 3},
{'pname': 'G#', 'oct': 3},
{'pname': 'A', 'oct': 3},
{'pname': 'A#', 'oct': 3},
{'pname': 'B', 'oct': 3},
{'pname': 'C', 'oct': 4},
{'pname': 'C#', 'oct': 4},
{'pname': 'D', 'oct': 4},
{'pname': 'D#', 'oct': 4},
{'pname': 'E', 'oct': 4},
{'pname': 'F', 'oct': 4},
{'pname': 'F#', 'oct': 4},
{'pname': 'G', 'oct': 4},
{'pname': 'G#', 'oct': 4},
{'pname': 'A', 'oct': 4},
{'pname': 'A#', 'oct': 4},
{'pname': 'B', 'oct': 4},
{'pname': 'C', 'oct': 5},
{'pname': 'C#', 'oct': 5},
{'pname': 'D', 'oct': 5},
{'pname': 'D#', 'oct': 5},
{'pname': 'E', 'oct': 5},
{'pname': 'F', 'oct': 5},
{'pname': 'F#', 'oct': 5},
{'pname': 'G', 'oct': 5},
{'pname': 'G#', 'oct': 5},
{'pname': 'A', 'oct': 5},
{'pname': 'A#', 'oct': 5},
{'pname': 'B', 'oct': 5},
{'pname': 'C', 'oct': 6},
{'pname': 'C#', 'oct': 6},
{'pname': 'D', 'oct': 6},
{'pname': 'D#', 'oct': 6},
{'pname': 'E', 'oct': 6},
{'pname': 'F', 'oct': 6},
{'pname': 'F#', 'oct': 6},
{'pname': 'G', 'oct': 6},
{'pname': 'G#', 'oct': 6},
{'pname': 'A', 'oct': 6},

```

```

        {'pname': 'A#', 'oct': 6},
        {'pname': 'B', 'oct': 6},
        {'pname': 'C', 'oct': 7}
    ]

    def __init__(self, **kwargs):
        # frecuencia fundamental mínima para detectar
        if 'min_f0' in kwargs:
            self._min_f0 = kwargs['min_f0']
        else:
            self._min_f0 = 65

        # frecuencia fundamental máxima para detectar
        if 'max_f0' in kwargs:
            self._max_f0 = kwargs['max_f0']
        else:
            self._max_f0 = 2100

        # longitud del marco de análisis
        if 'frame_len_sec' in kwargs:
            self._frame_len_sec = kwargs['frame_len_sec']
            if self._frame_len_sec != 0.046 and self._frame_len_
sec != 0.093:
                raise ValueError('Analysis frame length must be
46ms or 93ms')
        else:
            self._frame_len_sec = 0.093

        if 'window_func' in kwargs:
            self._window_func = kwargs['window_func']
        else:
            self._window_func = 'hanning'

        # ancho de bin del espectro estimado del parcial
        # de los fundamentales detectados
        if 'partial_width' in kwargs:
            self._partial_width = kwargs['partial_width']
        else:
            self._partial_width = 10

        '''
Derived parameters
'''
        if self._frame_len_sec == 0.046:
            self._alpha = 27
            self._beta = 320
            self._d = 1.0

```

```

        else:
            self._alpha = 52
            self._beta = 320
            self._d = 0.89

    def __call__(self, x, fs):
        X = self._stft(x, fs)
        Y = self._spectral_whitening(X, fs)
        return self._iterative_est(Y, fs)

    def estimate_f0s(self, x, fs):
        X = self._stft(x, fs)

        # Blanquear espectralmente la señal para suprimir la
        información tímbrica
        Y = self._spectral_whitening(X, fs)

        # estimación iterativa de los períodos fundamentales en
        el archivo de audio
        f0_estimations = self._iterative_est(Y, fs)

        # obtener notas que correspondan a estas estimaciones de
        frecuencia
        notes = []
        for frame_ests in f0_estimations:
            notes.append([self._freq_to_note(f) for f in frame_e
sts])

        return f0_estimations, notes

    def _freq_to_note(self, freq):
        i_note = np.argmin(np.abs(PitchGenerate.frequencies-
freq))
        return PitchGenerate.notes[i_note]

    def _stft(self, x, fs):
        '''
        transformada de Fourier de corto tiempo en la señal que
        en si es
        Hann con ventana y relleno de ceros al doble de su
        longitud.
        Hopsize = longitud de la ventana
        '''

        frame_len_samps = int(fs
* self._frame_len_sec) #O(1)

```

```

        win = get_window(self._window_func, frame_len_samps)
#O(1) la ventana de Hann es una formula sin bucles

        # zero-pad al doble de la longitud del marco frame
        K = int(nextpow2(2*frame_len_samps))
#O(1)
        X = np.array([np.fft.fft(win*x[i:i+frame_len_samps], K)
                       for i in range(0, len(x)-
frame_len_samps, frame_len_samps)]) # O(n log(n) * W)
        #O(n^2 log(n) + 3)
        return X

    def _spectral_whitening(self, X, fs, nu=0.33):
        """
        Aplanado espectral ('blanqueamiento') de la señal de
        entrada dada en el dominio de frecuencia,
        para suprimir la información tímbrica.

        PARAMETROS
        -----
        X (T, K): señal de entrada en el dominio de la
        frecuencia con tramas T y FFT de longitud K
        fs: frecuencia de muestreo de la señal de entrada
        nu (flotar): cantidad de blanqueamiento espectral
        """

        T, K = X.shape #O(1)
        nyquist_freq = fs/2 #O(1)
        nyquist_bin = K>>1 #O(1)

        # Calculo de las frecuencias centrales c_b (Hz) de las
        subbandas en la escala de la banda crítica
        # c_b = 229 * (10 ^ [(b + 1) /21.4] -1)
        # calculo de una subbanda por debajo y por encima del
        rango para obtener la cabeza y la cola
        # de frecuencias de las ventanas triangulares
        c = [] # frecuencia central de bandas criticas
        b = 0 # indice de bandas criticas
        while True:
            centre_freq = 229*(10**((b+1)/21.4)-1)
            if centre_freq < nyquist_freq:
                c.append(centre_freq)
                b += 1
            else:
                break

```



```

c = np.asarray(c)
c_bins = np.asarray(np.floor(c*K/fs), np.int)

# coeficientes de compresión de subbanda -> gamma (K/2,)
gamma = np.zeros([T, nyquist_bin])

for b in range(1, len(c_bins)-1):
    H = np.zeros(nyquist_bin)

    left = c_bins[b-1]
    centre = c_bins[b]
    right = c_bins[b+1]

    # Construyendo la respuesta de potencia triangular
    para cada subbanda.
    H[left:centre+1] = np.linspace(0, 1, centre - left
+ 1)
    H[centre:right+1] = np.linspace(1, 0, right - centre
+ 1)

    # multiplicado por 2, ya que la energía es simétrica
    con respecto a la tasa de nyquist
    gamma[:, centre] = np.sqrt((2/K)*np.sum(H*(np.abs(X[:,
:nyquist_bin])**2), axis=1))**(nu-1)

    # interpolar entre el bin central anterior y el bin
    central actual
    #para cada marco STFT
    for t in range(T):
        gamma[t, left:centre] = np.linspace(gamma[t, left]
, gamma[t, centre], centre - left)

    # calcular el espectro blanqueado. Solo es necesario
    almacenar la mitad del espectro para el análisis.
    # dado que la energía del contenedor es simétrica con
    respecto a la frecuencia de nyquist
    Y = gamma * X[:, :nyquist_bin]

    return Y

def _iterative_est(self, Y, fs):
    f0_estimations = []

    T = Y.shape[0]
    # ventana STFT
    for t in tqdm(range(T)):

```

```

# espectro de magnitud residual del frame de
análisis
Y_t_R = np.abs(Y[t,:])

# estimaciones de frecuencia fundamental para el
frame actual
f0_frame_estimations = []

# realizar un seguimiento de las prominencias de las
estimaciones del período en este frame
S = -1
saliency_hats = []

tau_hat, saliency_hat, Y_t_D = self._search_smax(Y_t
_R, fs, tau_prec=0.5) #n^3 *logn
saliency_hats.append(saliency_hat)

f0_frame_estimations.append(fs/tau_hat)
f0_estimations.append(f0_frame_estimations)

cur_S = self._calc_S(saliency_hats)
if cur_S <= S:
    break
else:
    Y_t_R -= self._d*Y_t_D
    Y_t_R[Y_t_R < 0] = 0

    S = cur_S

return f0_estimations

def _calc_S(self, saliency_hats, gamma=0.7):
    '''
    Calculamos una suma normalizada de prominencias para
    determinar
    si es necesario buscar más frecuencias fundamentales en
    el espectro.
    '''

    j = len(saliency_hats)
    S = sum(saliency_hats)/(j**gamma)

    return S

```

```

def _search_smax(self, Y_t_R, fs, tau_prec=1.0):
    Q = 0          # index de un nuevo block
    q_best = 0     # index del mejor block

    tau_low = [round(fs/self._max_f0)] # samples/cycle
    tau_up = [round(fs/self._min_f0)]  # samples/cycle
    smax = [0]

    while tau_up[q_best] - tau_low[q_best] > tau_prec:
        Q += 1
        tau_low.append((tau_low[q_best] + tau_up[q_best])/2)
        tau_up.append(tau_up[q_best])
        tau_up[q_best] = tau_low[Q]

        for q in [q_best, Q]:
            salience, _ = self._calc_salience(Y_t_R, fs, tau
_low[q], tau_up[q])
            if q == q_best:
                smax[q_best] = salience
            else:
                smax.append(salience)

        q_best = np.argmax(smax)

    # período fundamental estimado del frame
    tau_hat = (tau_low[q_best] + tau_up[q_best])/2

    # calculamos el espectro del período fundamental
detectado y los armónicos
    salience_hat, harmonics = self._calc_salience(Y_t_R, fs,
tau_low[q_best], tau_up[q_best])
    K = len(Y_t_R)<<1
    Y_t_D = self._calc_harmonic_spec(fs, K, harmonics)

    return tau_hat, salience_hat, Y_t_D

def _calc_salience(self, Y_t_R, fs, tau_low, tau_up):
    salience = 0

    tau = (tau_low + tau_up)/2
    delta_tau = tau_up - tau_low

    # calculamos el número de armónicos bajo la frecuencia
nyquist
    # la siguiente declaración es equivalente a
floor((fs/2) / fo)
    num_harmonics = int(np.floor(tau/2))

```

```

        # calcular todos los harmonic weights
        harmonics = np.arange(num_harmonics)+1
        g = (fs/tau_low + self._alpha) / (harmonics*fs/tau_up
+ self._beta)

        nyquist_bin = len(Y_t_R)
        K = nyquist_bin<<1
        lb_vicinity = K/(tau + delta_tau/2)
        ub_vicinity = K/(tau - delta_tau/2)

        harmonics = []
        for m in range(1,num_harmonics+1):
            harmonic_lb = round(m*lb_vicinity)
            harmonic_ub = min(round(m*ub_vicinity), nyquist_bin)
            harmonic_bin = np.argmax(Y_t_R[harmonic_lb-
1:harmonic_ub]) + harmonic_lb-1
            harmonic_amp = Y_t_R[harmonic_bin]
            w_harmonic_amp = g[m-1] * harmonic_amp

            # guardar las propiedades de este período
fundamental y los armónicos
            harmonics.append({'bin': harmonic_bin, 'amp':
w_harmonic_amp})

            salience += w_harmonic_amp

        return salience, harmonics

def _calc_harmonic_spec(self, fs, K, harmonics):
    nyquist_bin = K>>1
    # inicializar el espectro de armónicos detectados
    Y_t_D = np.zeros(nyquist_bin)

    # calculamos el espectro parcial para cada armónico

    frame_len_samps = int(fs * self._frame_len_sec)
    win = get_window(self._window_func, frame_len_samps)
    window_spec = np.abs(np.fft.fft(win, K))
    partial_spectrum = np.hstack((window_spec[self._partial_
width::-1],
                                window_spec[1:self._partial_
width+1]))
    # normalizamos el espectro
    partial_spectrum /= np.max(partial_spectrum)

    for h in harmonics:

```

```

        h_lb = max(0, h['bin']-self._partial_width)
        h_ub = min(nyquist_bin-
1, h['bin']+self._partial_width)

        # traducimos el espectro de la función de ventana a
        la posición del armónico
        Y_t_D[h_lb:h_ub+1] = h['amp']*partial_spectrum[h_lb-
h['bin']+self._partial_width:h_ub-
h['bin']+self._partial_width+1]

        return Y_t_D

    def collapse_notes(self, notes):
        '''
        Contraemos notas consecutivas (notas que abarcan más de
        un marco de análisis).
        '''

        notes_c = []
        prev_frame = []
        for frame_n in notes:
            if len(frame_n) > 1:
                n_set = set([n['pname']+str(n['oct']) for n in f
rame_n])
                frame_n = [{'pname': n[:-1], 'oct': int(n[-
1])}] for n in n_set

                if len(frame_n) != len(prev_frame):
                    notes_c.append(frame_n)
                elif not np.all([n1['pname'] == n2['pname'] and n1['
oct'] == n2['oct']
                                for n1,n2 in zip(prev_frame, frame_n
) ]]):
                    notes_c.append(frame_n)

                prev_frame = frame_n

        return notes_c

    def estimate_multipitch(signal, sampleRate, polyphony_max=6):
        freq_est = PitchGenerate(max_poly=polyphony_max)
        f0_estimates, notes = freq_est.estimate_f0s(signal, sampleRa
te)
        #notes_c = freq_est.collapse_notes(notes)
        return f0_estimates

```

E.3. PITCH: DETECCIÓN DE ONSET (onset.py)

```
import numpy as np
import librosa
import matplotlib.pyplot as plt

# matplotlib options
from matplotlib import rcParams
rcParams['savefig.transparent'] = True
rcParams['text.usetex'] = False

class OnsetGenerate(object):
    def __init__(self, x, fs, h, uc):
        self.x = x
        self.fs = fs

        self.onsets = []
        self.onset_times_f = []
        self.onset_envelope = []
        self.tempo = 120

        self.heuristica = 3 if h == 0 else h

        self.umbral_compensation = 0.5 if uc == 0 else uc/100

    def __call__(self):
        self.onsets, self.onset_envelope, self.tempo = self.second_onset_detect(self.x, self.fs)
        self.onset_times_f = librosa.frames_to_time(self.onsets)

        peaks = librosa.util.peak_pick(self.onset_envelope, self.heuristica, self.heuristica, self.heuristica, 50, self.umbral_compensation, 10)
        #peaks = librosa.util.peak_pick(self.onset_envelope, 3, 3, 3, 5, 0.5, 10)
        times = librosa.frames_to_time(np.arange(len(self.onset_envelope)), sr=self.fs, hop_length=512)
        '''
        peaks = []
        for i in range(len(self.onset_envelope)):
            if self.onset_envelope[i] >= 2:
                peaks.append(i)
        '''
```

```

        #self.graphic(self.onsets, self.onset_envelope)
        self.graphic2(self.onset_envelope, self.fs, peaks, times
    )

    return np.multiply(times[peaks],2), self.tempo
    #return self.onset_times_f, self.tempo

    def graphic(self, onsets, onset_envelope):
        fig2, ax2 = plt.subplots(figsize=(25,10))
        ax2.plot(onset_envelope, label='onset strength')
        ax2.vlines(onsets, 0, onset_envelope.max(), color = 'r',
alpha=0.25)
        ax2.set_xticks([], ax2.set_yticks([])

        fig2.show()

    def graphic2(self, onset_envelope, sr, peaks, times):
        '''
        peaks = librosa.util.peak_pick(onset_envelope, 3, 3, 3,
5, 0.5, 10)
        times =
        librosa.frames_to_time(np.arange(len(onset_envelope)),
                                sr=sr, hop_length=512)
        '''
        fig2, ax2 = plt.subplots(figsize=(25,10))
        ax2.set_xlabel('Time (s)')
        ax2.set_ylabel('ONSET STRENGTH')

        ax2.plot(times, onset_envelope, label='onset strength')
        ax2.vlines(times[peaks], 0, onset_envelope.max(), color
= 'r', alpha=0.25)
        #ax2.set_xticks([], ax2.set_yticks([])

        fig2.show()

    def first_onset_detect(self, onset_times, pitch, time):

        onsetReplace = []
        onsetValue = 0

        onsetFinal = []

        for i in range(len(pitch)):

```



```

        if(onsetValue != pitch[i]):
            onsetValue = pitch[i]
            onsetReplace.append(time[i])
    #print(onsetReplace)
    onsetFinal.append(onsetReplace[0])
    verify = False
    '''
    for i in range(len(onset_times)):
        onset_sample = onset_times[i]
        for j in range(len(onsetReplace)):
            if(abs(onset_sample-onsetReplace[j]) < 0.12):
                onsetFinal.append(onsetReplace[j])
                verify = True
                break
            else:
                verify = False
        if(not verify):
            onsetFinal.append(onset_sample)
    #print('onset azul: ', onsetFinal)
    '''
    return onsetFinal

def second_onset_detect(self, x, fs):
    onset_envelope = librosa.onset.onset_strength(x, sr=fs)

    #onsets = librosa.onset.onset_detect(onset_envelope =
onset_envelope, hop_length=512,
    #
    wait=35,
pre_avg=100, post_avg=30, pre_max=30, post_max=30) #35 #100 30
30 70

    onsets = librosa.onset.onset_detect(onset_envelope = onset_envelope, hop_length=18, wait = 10)

    tempo, beats = librosa.beat.beat_track(onset_envelope = onset_envelope)

    return onsets, onset_envelope, tempo

```

E.4. TRANSCRIPCIÓN: CREACIÓN DE LA PARTITURA MUSICAL (score.py)

```
import numpy as np
import music21
import sys

class MiditoScore(object):
    def __init__(self, midi_note, midi_time, onset_times, tempo,
audio_duration, name):
        self.midi_note = midi_note
        self.midi_time = midi_time
        self.onset_times = onset_times
        self.tempo = tempo
        self.audio_duration = audio_duration
        self.name = name

        self.base_note = 125 # 4/4 con 120 bpm default
        self.midiN = []
        self.duration = []

        #Creacion de una nueva partitura
        self.s = music21.stream.Stream()

    def __call__(self):
        self.base_note = self.tempoToMilliSeconds(self.tempo)
        self.midiN, self.duration = self.onsetFilter(self.midi_note, self.midi_time, self.onset_times, self.audio_duration)
        self.generateScore(self.midiN, self.duration, self.base_note)
        return self.midiN

    def tempoToMilliSeconds(self, tempo):
        onefour_note = tempo/60
        base_note = onefour_note/4
        return base_note # nota base: doble corchea

    def onsetFilter(self, midi_note, midi_time, onset_times, last_duration):
        midiN = []
        duration = []
        midi_note = midi_note[0]

        if(abs(0-onset_times[0]) < 0.5):
            onset_times = onset_times[1:]
            midiN.append(midi_note[0])
```

```

else:
    midiN.append(midi_note[0])

duration_ant = 0

for i in range(len(onset_times)):
    time = onset_times[i]

    for j in range(len(midi_note)):
        aprox_time = midi_time[j]
        if(time <= aprox_time and (j+1)<len(midi_note)):
            '''
                if(midi_note[j] != midi_note[j+1] and
midi_note[j] == midi_note[j-2]):
                    midiN.append(midi_note[j+1])
            else:
                midiN.append(midi_note[j])
            '''
            midiN.append(midi_note[j])

            duration.append(time-duration_ant)
            duration_ant = time
            break
        else:
            if(j+1 >= len(midi_note)):
                midiN.append(midi_note[j])
                duration.append(time-duration_ant)
                duration_ant = time

    duration.append(duration[-1])

#print('notas finales: ', midiN)
#print('notas finales duracion: ', duration)

return midiN, duration

def QuarterLengthAprox(self, time):
    #quarterlength
    qls = [1/4, 1/2, 1, 1*2, 1*4]
    ql = [1*4, 1*2, 1, 1/2, 1/4]

    for data in ql:

```

```

        if(abs(data-time) < 0.25):
            return data
        elif(abs((data+data/2) - time) < 0.25):
            return data+(data/2)

    return 0

def generateScore(self, midi_note, onset_times, base_note):
    onset_times = np.around(onset_times,2)
    qls = [1/4, 1/2, 1, 1*2, 1*4]

    #Creacion de una nueva partitura
    self.s = music21.stream.Stream()

    #Asignación del tempo
    mm1 = music21.tempo.MetronomeMark(int(base_note*(4*60)))
    self.s.append(mm1)

    #compas 4/4
    ts = music21.meter.TimeSignature('4/4')
    self.s.append(ts)

    s_array = []
    pitch = 0

    Violin = music21.instrument.Violin()
    Violin.midiChannel=0
    self.s.append(Violin)

    print(len(midi_note),len(onset_times))
    #print('onset_time ant: ', onset_times)
    for i in range(len(midi_note)-1):
        silence = 0

        time= self.QuarterLengthAprox(onset_times[i])

        if(time==0):
            j = 0
            while(time == 0 and j < len(qls)):
                silence = qls[j]
                time = self.QuarterLengthAprox(onset_times[i
]-silence)

                j+=1

```

```

        s_array.append(music21.note.Note(midi_note[i], q
quarterLength = time))
        s_array.append(music21.note.Rest(quarterLength =
silence))

    else:
        s_array.append(music21.note.Note(midi_note[i], q
quarterLength = time))

s_array.append(music21.note.Note(midi_note[-1]))

#print(s_array)
self.s.append(s_array)

for n in self.s.notes:
    if n.pitch.accidental.name == "natural":
        n.pitch.accidental.name = "none"

key=self.s.analyze('key')
self.s.finalBarline = 'final'

self.s.insert(0, music21.metadata.Metadata())
self.s.metadata.title = self.name
self.s.metadata.composer = ''

#s.append(music21.key.KeySignature(3))
output = sys.path[-1] + "/samples/output/"

output = output.replace("\\", "/")
self.s.write(fp = output
+ self.name+'.png', fmt = 'musicxml.png')
self.s.write(fp = output
+ 'PDFcreate.pdf', fmt = 'musicxml.pdf')
#self.s.write(fp = output + 'outputfile.mp3', fmt =
'musicxml.png')

self.s.write("midi", output + "music21.mid")

```

E.5. PRINCIPAL

```
import sys
from os import path
project_root = path.abspath(path.join(path.dirname(path.abspath(
__file__)), '..'))
sys.path.append(project_root)

from harmonic_algorithm.io import AudioLoader
from harmonic_algorithm.pitch_onset import MonoPitch, OnsetGenerator
from harmonic_algorithm.transcription.score import MiditoScore
from harmonic_algorithm.util.units import Hz_to_MIDI2

from harmonic_algorithm.eval import transcription_eval, midi_to_score

import numpy as np
import matplotlib.pyplot as plt

# matplotlib options
from matplotlib import rcParams
rcParams['savefig.transparent'] = True
rcParams['text.usetex'] = False

def demo(audio, name, h, uc):

    print(sys.path[-1])
    #Obtener señales musicales
    fs = audio.sampleRate
    x = audio.signal
    audio_duration = audio.duration

    #Tamaño de ventana
    isset = 1
    frameSize = 2048*isset

    #Aplicar algoritmo harmonic
    kl = MonoPitch(x, fs, frameSize=frameSize)

    pitch = np.around(Hz_to_MIDI2(kl()), 0)
    time = np.arange(pitch.shape[1]) * (frameSize / fs)*4/isset

    #Obtener onsets y graficarlo
```

```

onset_times = OnsetGenerate(x, fs, h, uc)
onset_times_f, tempo = onset_times()

#Generar Partitura
score = MiditoScore(pitch, time, onset_times_f, tempo, audio
_duration, name)
score()

#Graficar piano roll
graphic_piano_roll(time, pitch, onset_times_f)

'''
#MIR EVAL
midi_to_score.main_converter(name)

output = sys.path[-1] + "/mir_eval/clarinete/"
output = output.replace("\\", "/")
reference_file = output + name + ".txt"
estimated_file = output + name + "_estimated" + ".txt"
output_file = output + name + "_results" + ".txt"
transcription_eval.__eval(reference_file, estimated_file,
output_file)
'''

def graphic_piano_roll(time, pitch, onset_times_f):
    fig, ax = plt.subplots(figsize=(25,10)) #25 10
    fig.suptitle("PITCH ESTIMATION: HARMONIC
ALGORITHM", fontsize=16)
    #ax.set_title("Piano roll of Beethoven's \"Für Elise\"")
    ax.set_title("Piano Roll")
    #pitch.shape[0]
    for m in range(pitch.shape[0]):
        ax.scatter(time, pitch[m])
        #print('Midi: ', pitch[m])

    ax.set_xlabel('Time (s)')
    ax.set_ylabel('Pitch (MIDI)')
    #ax.set_ylim(20, 109)
    ax.set_ylim(30, 100)
    ax.set_yticks(np.arange(30,100,10))

    #plt.show()
    ax.vlines(onset_times_f, 30, 100, color='r')

def initInterface(filepath, audio, h, uc):
    audio_file = filepath
    name = audio_file.split("/")

```



```

name = name[-1].split(".")
name = name[0]
demo(audio, name, h, uc)

if __name__ == '__main__':
    songs = [
        "1 violin-twinkle",
        "2 Lightly_Row",
        "4 Go_Tell_Aunt_Rhody",
        "5 O_Come_Little_Children",
        "7 1_Long_Long_Ago",
        "9 Perpetual Motion",
        "11 1 Nocturne-Andantino_voor_piano_en_viool_2013-
Viool",
        "13 Minuet_1",
        "14 Minuet_2",
        "15 Minuet_3",
        "16_The_Happy_Farmer2"]
    h = 3
    uc = 57

    audio_file = "../harmonic/samples/input/violin/" +
songs[1] + ".wav"
    name = audio_file.split("/")
    name = name[-1].split(".")
    name = name[0]

    audio = AudioLoader(audio_file)
    audio.cut(stop=5)
    print("Using first ", 10, " seconds of ", audio_file)

    demo(audio, name, h, uc)

```

ANEXO F – REPOSITORIO DE CÓDIGO

Enlace del repositorio de código en GitHub:

<https://github.com/TheProdigyK/HarmonicAlgorithm-thesis>



DOCUMENTACIÓN