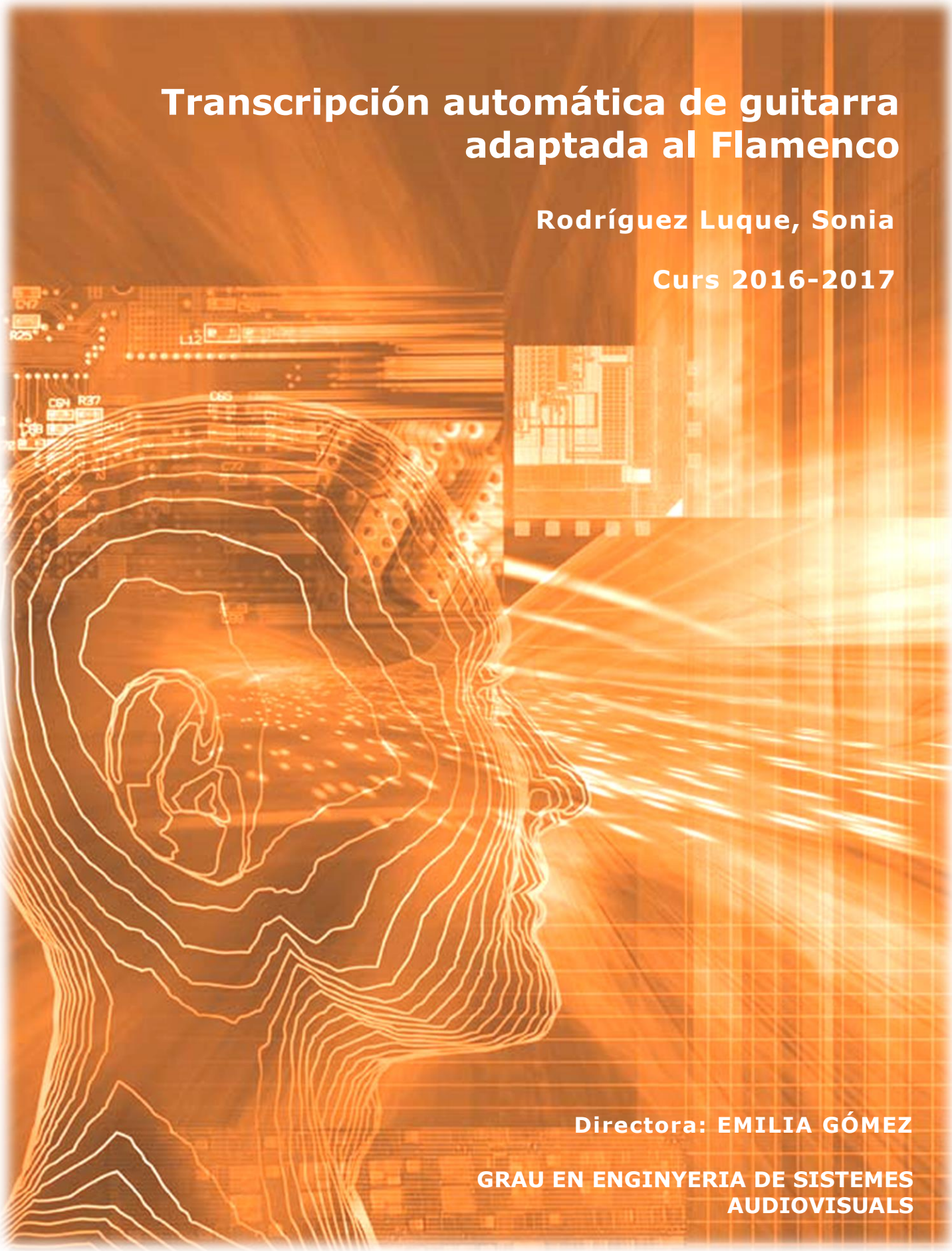
Transcripción automática de guitarra adaptada al Flamenco

Rodríguez Luque, Sonia

Curs 2016-2017



Directora: EMILIA GÓMEZ

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Transcripción automática de guitarra adaptada al Flamenco

Sonia Rodríguez Luque

TRABAJO DE FIN DE GRADO

INGENIERÍA DE SISTEMAS AUDIOVISUALES

ESCOLA SUPERIOR POLITÈCNICA UPF

2016 - 2017

DIRECTORA DEL TRABAJO

Emilia Gómez

Agradecimientos

En primer lugar, agradecer especialmente a Emilia Gómez, por toda la implicación y dedicación con la supervisión de este proyecto. Por introducirme en el equipo del MTG, del que he aprendido y disfrutado durante estos meses. También por darme la oportunidad de trabajar en el contexto de CoFla, un proyecto del que valoro enormemente su importancia y evolución, por todo lo que puede aportar a la riqueza musical y cultural de la que ya dota el Flamenco.

A Helena, por procurar que nunca faltara café, puertas del tiempo, sopas y cosas *random*. Y por enseñarme muchas cosas, aunque me "cuesta" aceptarlo.

A mi familia por aguantarme y animarme, aunque el viento no soplara a favor, a llegar hasta aquí. Y aunque no de sangre, a los míos, los que se han convertido en mi familia de carrera durante estos años. Finalmente, a Anna, por acompañarme en todo, siempre.

Resumen

El papel de la guitarra en el Flamenco, a diferencia de otros géneros, tiene una forma de transmisión del arte oral; tanto las canciones como la terminología usada, han pasado de generación en generación sin haber sido escritas.

En el área de *Music Information Retrieval* (MIR), se han desarrollado algoritmos que permiten obtener, automáticamente, representaciones simbólicas a partir de análisis de grabaciones sonoras. En este proyecto abordamos el problema de la transcripción automática de guitarra flamenca. El principal objetivo es el desarrollo de un algoritmo que procese una señal audio, que contenga una o varias falsetas de guitarra flamenca, para obtener su representación simbólica. Para ello, deberá primero localizar los segmentos sonoros considerados como falseta que, posteriormente, serán transcritos automáticamente a un archivo MIDI.

La meta de este proyecto es hacer, de este algoritmo, una herramienta útil tanto para el aprendizaje, como para el estudio de la guitarra flamenca. Intentamos así, proporcionar un soporte informático como primer paso para la única alternativa existente en la actualidad del flamenco: la transcripción manual, muy costosa y que requiere conocimientos, del flamenco y musicales, avanzados.

Abstract

Unlike other genres, the role of the guitar in Flamenco music is transmitted orally; both songs and terminology have passed down across generations without a writing system.

In the field of Music Information Retrieval (MIR), some algorithms have been developed to automatically obtain symbolic representations by analysing audio recordings. In this project, we deal with the problem of automatic transcription of flamenco guitar. The main goal is to develop an algorithm to process an audio signal which contains one or several guitar *falsetas* and extract their symbolic representation. To do so, we first need to locate the segments considered as a *falseta*, which are then transcribed into a MIDI file.

The goal of this project is to develop a tool which is useful both for learning and studying flamenco guitar. We aim at providing a computer-aided system as a first step to the only current alternative in flamenco: manual transcription, which is very difficult and requires advanced music and flamenco knowledge.

Índice

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN

Hablamos de flamenco como género musical en el que conviven, además de tradiciones y normas propias, tres facetas fundamentales: cante, toque y baile. Por definición, es una manifestación cultural de carácter popular andaluz, vinculado por su origen, al pueblo gitano [1]. No obstante, encontramos un origen ambiguo y numerosas vertientes que dotan de una gran controversia, imposibles de comprobar históricamente.

En referencia a este género, percibimos una gran fusión de distintas culturas coincidentes en Andalucía a lo largo de su creación y evolución. Según Lorca, su origen cultural se atribuye a los moriscos, pero por aquel entonces, el gran mestizaje cultural presente en Andalucía favoreció su aparición: oriundos, musulmanes, gitanos, castellanos y judíos. Se habla también del hecho de la existencia del flamenco antes de que llegaran los gitanos, puesto que, existiendo en toda Europa, este género únicamente se desarrolló en Andalucía [2].

Además de su origen geográfico, este movimiento sociocultural ha logrado extenderse universalmente, consiguiendo así, numerosos reconocimientos a nivel nacional e internacional. El más importante ocurrió en 2010, cuando la Unesco lo declaró Patrimonio Cultural Inmaterial de la Humanidad [3].

Durante el desarrollo del flamenco, debido a las situaciones geográficas y socioculturales, encontramos que el género se va descomponiendo en otros subgéneros llamados *palos.* No obstante, la mayoría mantienen componentes singulares que son esencia del flamenco: modulaciones y melismas provenientes de los cantos islámicos, el sistema musical judío, modos jónicos y frigios del canto bizantino entre otros. En su evolución: arte mestizo que aglutina influencias armónicas y rítmicas de músicas de todo el mundo sobre una base musical exclusivamente peninsular.

1.1 Breve historia del papel de la guitarra en el flamenco

De igual forma, el flamenco se trataba de una forma de expresión vocal y coreográfica dónde el papel protagonista de la guitarra era prácticamente nulo. Con el paso del tiempo, se fue integrando, al igual que otros instrumentos. En la época contemporánea ha ido adquiriendo un protagonismo mucho más notable, asumiendo papeles protagonistas, siendo más valorada y objeto de estudio.

La guitarra flamenca, en su evolución, ha ido incluyendo influencias de todo tipo a su lenguaje, principalmente definido por el ritmo, enriqueciendo su forma expresiva. Además del ritmo, en su mayoría compases ternarios, se suman los acordes rasgados, formando ciclos armónico-rítmicos con el que se acompaña el cante y el baile.

Las influencias fueron de índole internacional, pero también nacional, como es el caso de, entre muchas otras, la Jácara: género satírico del siglo XVII y XVIII dónde ya

aparecían, referentes a la guitarra, elementos como el uso del compás ternario en hemiolia[1], secuencias armónicas de tonalidad frigia y frases estructuradas en ciclos de 12 tiempos.

A mediados del siglo XIX, cuando el flamenco se asienta de forma clara, aparecen figuras pioneras en la guitarra que aportan y enriquecen el lenguaje: Entre otros, Paco el de Lucena, Javier Molina, Miguel Borrull o Ramón Montoya, creador del toque por taranta, rondeña y minera. Aparece también la transición del toque por medio al toque por arriba, y posteriormente definir otros tipos de toque y tonos.

Más tarde, y con la gran influencia hispanoamericana, en el siglo XX, se incluyen otros instrumentos que darán pie a la etapa de la ópera flamenca. Después, guitarristas y pioneros concertistas, como Agustín Castellón "Sabicas" o el Niño Ricardo, consolidaron la guitarra como algo más que acompañamiento del cante, internacionalizando el género, y marcando un punto de inflexión.

Después de un estancamiento creativo, en los 70, llegan las influencias del rock, el jazz y el pop, propiciando una renovación del lenguaje, llevada a cabo por los artistas de la etapa llamada "Nuevo Flamenco". Transcendieron de lo que se denominaba "arte puro" y artistas como Enrique Morente, Camarón de la Isla, Lole y Manuel, Triana, o Paco de Lucía, se encargaron de abrir y crear una nueva cultura flamenca que marcó un antes y un después en el género.

En referencia a la guitarra, Paco de Lucía, es el gran protagonista de esta época. Siendo responsable en su totalidad de la expansión universal de la guitarra flamenca y de la innovación relacionada con la fusión, añadiendo detalles armónicos propios de otros géneros, giros, modulaciones y tonalidades que se escapan de lo tradicional, pero manteniendo la base flamenca.

Innombrables avances, el uso de efectos, la inclusión de otros instrumentos propios de otras culturas, y que actualmente, son básicos en el flamenco, como el cajón. Consiguiendo así, la liberación total de la función de acompañante, dotando a la guitarra de mayor libertad y posibilidad de improvisación. Más tarde, Manolo Sanlúcar aportó una gran ampliación del campo tonal del flamenco. [4]

a) Características sonoras y técnicas

Con el fin de facilitar la posterior comprensión de algunos términos y metodologías usadas, en esta sección se explican las diferencias entre guitarra flamenca y clásicas, técnicas y tipos de interpretación.

La guitarra flamenca tiene un ataque más rápido: después de pulsar la cuerda, la tapa vibra rápidamente, ataca, y alcanza el máximo nivel de volumen más rápido que una

[1] Hemiolia o hemiola: Fenómeno rítmico en el que los acentos cambian de posición en una ratio 3:2.

clásica. Este hecho hace que tenga más "percusividad" y facilite el toque en tempos rápidos. De la misma manera, el volumen también tiende a caer más rápido y mantenerse menos en el tiempo: se corresponde con el concepto de "guitarra seca", que da más definición a las notas ejecutadas y no emborrona el sonido.

Otra característica fundamental es la claridad y brillantez del sonido, con menos armónicos, que facilita el buen resultado de algunas técnicas del toque y el equilibrio en el acompañamiento. Esto se debe a un balance de frecuencias, donde los graves producen menos energía y se realzan los medios y los agudos. [5] También se debe al tipo de madera, que normalmente es ciprés macizo para los arcos y el fondo, y abeto alemán para la tapa. Además, el cuerpo suele ser más estrecho que el de la clásica, dotándolo de un sonido más nasal.

En cuanto a las técnicas más comunes encontramos el trémolo y los arpegios, cuya diferencia está en que, en el arpegio, cada dedo índice, medio y anular tocan diferentes cuerdas con sus respectivas notas. Al contrario del trémolo: el dedo pulgar es el que toca diferentes cuerdas, y los demás dedos tocan la misma nota y cuerda, uno detrás del otro en tiempo, normalmente equivalente a una negra. Estas dos técnicas junto con el picado, que consiste en alternar dos dedos (normalmente medio y índice) ejerciendo la fuerza en el nudillo, corresponden al toque "por arriba". En cambio, en el toque "por medio" o "por lo flamenco" encontramos los rasgueos y el *alzapúa* (se usa íntegramente el pulgar) especialmente usadas cuando se ejerce el acompañamiento al cante o baile.

1.2 Proceso comunicativo: Diálogos y Falsetas

El flamenco se compone de una articulación perfecta de sus elementos, para formar, una unidad sonora y sensorial. Por lo tanto, toque, cante y baile se complementan y se comunican en un proceso que enriquece el lenguaje, uniéndose en un vínculo que de no-subordinación, si no de comunicación.

El denominador común de todas ellas es el ritmo, factor fundamental e innato que envuelve todo el conjunto y que se exterioriza en modos propios de cada sección. Todos los elementos tienen que mantener unos códigos de comunicación, como el compás, encargado de unificar el ritmo para los tres componentes fundamentales, transcendiendo de la terminología estrictamente musical, matizado por los acentos. [6]

Nos centraremos en el dialogo cante – toque, dado que es la parte potencialmente útil para desarrollar el algoritmo, sobre todo la extracción de lo que denominamos falseta. Una falseta se define como una frase melódica realizada por la guitarra, que se intercala entre las sucesiones de acordes correspondientes al acompañamiento del cante. [7]

El proceso comunicativo se inicia con una introducción instrumental que define el palo, su respectivo compás y tonalidad. En base a esta información, se produce el temple de la voz ejecutado por el cantaor, que se acomoda en la base armónica y rítmica. Después

de esta primera "conversación", puede que la parte instrumental pueda dar un apunte instrumental en forma de falseta corta. Seguidamente se inicia el primero de los tercios[2], ejecutados cada uno de ellos con una intención y recursos distintos.

Según Lola Fernández, en su libro "Teoría Musical del Flamenco" [37], se definen algunas secciones dentro del acompañamiento instrumental:

- Introducción: secuencia breve que se puede ejecutar mediante rasgueados de acordes pertenecientes a la tonalidad de la obra, o mediante adornos[3] tales como floreos o apoyaturas de carácter similar al rasgueado, es decir no melódico.
- Llamadas: como clave en la construcción del cante, haciendo sonar la cadencia típica y el compás del cante para identificar rápidamente del tipo de cante que se va a realizar. Sirve también como nexo entre la voz y el instrumento, o bien para dar paso a una falseta
- Falseta: propiamente, interludios instrumentales que se alternan con las estrofas vocales.
    - Cierre: elementos de duración muy corta, normalmente un compás, que ayudan a concluir falsetas o llamadas.

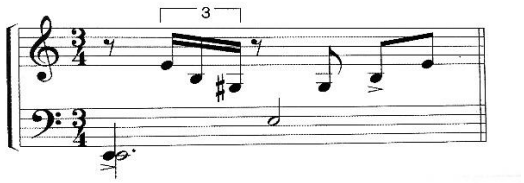


Fig 1.1 Ejemplo de cierre de Soleá para piano en Mi Flamenco.

    - Remate: elemento propio de la guitarra, que consiste en cerrar o "rematar" una falseta o una llamada, acelerando los dos últimos compases, creando un ambiente de precipitación al cierre.

El papel de la guitarra durante los tercios es complementario a la voz, pudiendo realizar distintos tipos de toque: solo acordes, en contrapunto (pudiendo realizar una melodía paralela a la de la voz), rellenando los espacios entre versos o palabras con algún matiz melódico, o realizando un ritmo con la mano derecha y tapando las cuerdas con la izquierda, sin melodía.

Entre estrofas, como hemos visto, aparecen las falsetas: parte potencialmente útil para la posterior extracción de estas, ya que podremos asumir, que las partes donde no haya cante, encontraremos posibles falsetas. En cuanto a la duración de estas, existe una controversia notable, desde su aparición hasta la actualidad. Empezaron como frases

[2] Tercio: Frases musicales de cada estrofa. Puede incluir uno o dos versos.
[3] Notas de adorno: notas que no pertenecen al acorde: floreo, apoyatura, retardo, nota de paso, nota escapada y anticipación

“breves”, aunque la relevancia que se le ha dado a la guitarra con el paso del tiempo ha podido modificar los conceptos de duración.

Se considera más importante que exista un equilibrio, y no tanto el tiempo exacto de cada elemento [8]. Teniendo en cuenta este hecho, planteamos la opción de incluir parámetros variables que permitan al usuario tener el control sobre la duración de cada parte.

1.3 El Flamenco bajo una perspectiva tecnológica

Dados los importantes avances tecnológicos a nivel musical en el siglo XX y XXI, la etnomusicología, también es objeto de investigación. Esta investigación tiene como objetivo entender las estructuras y significados atribuidos a las músicas del mundo, teniendo en cuenta la importante tradición oral que normalmente tiene, y sobre todo el entorno cultural donde se han desarrollado. Por lo tanto, comprender las conexiones entre aspectos culturales y usarlas a favor de los problemas planteados para las investigaciones [11]. Este concepto se define como Etnomusicología Computacional y aparece por primera vez en 1978. [12]

En concreto el Flamenco, como fenómeno cultural y musical, ofrece una identidad y originalidad dignas de análisis y estudio en profundidad mediante métodos científicos. Dentro del gran abanico de proyectos que dan pie a este estudio, encontramos: el análisis de las propiedades de preferencia de diferentes palos para los aficionados, comparación automática de ritmos o melodías y similitudes, o la obtención de una transcripción de disciplinas como el cante. [11]

El uso exponencial del internet, y actualmente, la cantidad de servicios que ofrecen posibilidad de reproducir, grabar, editar, tratar y procesar datos han influenciado y motivado a cubrir las necesidades que surgen a raíz de estos hechos. Estas necesidades exigen que se abra un nuevo campo de investigación, que trate de solucionar nuevas formas de análisis de archivos musicales, clasificación, indexación, transcripción, procesado de señal, etc. [9]

Se denomina *Music Information Retrieval* (MIR) al área interdisciplinar que se dedica a la recuperación de información musical mediante la aplicación de técnicas computacionales y modelos matemáticos, donde encontramos una estrecha relación entre música y matemáticas, tanto en el terreno analítico como en el compositivo. Es importante destacar que para llevar a cabo estas investigaciones también pueden estar involucradas técnicas de musicología, psicología, estudios musicales, procesado de señal o *machine learning*, entre otras.

a) Técnicas de MIR y análisis computacional: CoFla

Debido a su reciente desarrollo, las técnicas de MIR, en su mayoría, son de tipo genérico. Por lo tanto, surge una necesidad de desarrollar ciertas metodologías y nuevos sistemas que se pueden usar específicamente en algunos estilos musicales. Este hecho requiere tener conocimientos específicos tanto técnicos como artísticos del área musical donde se quieran implementar. Aquí confluyen los rasgos de la etnomusicología computacional y las conexiones con las técnicas convencionales de MIR.

En el caso concreto del Flamenco, el Proyecto CoFla, establece las bases para la investigación del género y su teoría computacional, dando solución a las necesidades de adaptación y optimización de los métodos convencionales, incluso los usados para otros géneros específicos [14]. El objetivo principal radica en estudiar analíticamente las estructuras musicales del flamenco basadas en el uso de herramientas computacionales y como estas pueden dar pie a análisis de particularidades específicas del género, representaciones, síntesis [13].

Se define como un área interdisciplinar, ya que, para su óptimo funcionamiento, se requiere la combinación de conocimientos relacionados con Historia, Literatura, Antropología del flamenco, Musicología, Ingeniería, etc. [13] Este proyecto abarca un amplio rango de actuación, desde creación de corpus de anotaciones y transcripciones hasta la creación de algoritmos para transcripciones de cante o clasificadores automáticos de subgéneros [14].

Se complementa con la intención divulgativa y fortalecedora del campo musical del flamenco y su avance, tanto a nivel cultural como a nivel tecnológico. Algunos de los temas tratados dentro de la Flamencología Computacional por CoFla son [13]:

- Transcripción automática de cante: con el fin de solucionar los malos resultados que se obtienen si se aplican métodos generales al cante flamenco. Estos problemas son debidos a la cantidad de afinaciones micro-tonales, modulaciones y melismas. Además, de la gran tradición de transmisión oral y poco trascrita manualmente, dando una herramienta útil para el análisis, aprendizaje y la docencia de este arte. [18]
- Extracción automática de melodía u otros aspectos como timbre o ritmo.
- Similitud melódica, que dota de gran dificultad, ya que da pie a otros problemas a desarrollar, como la limpieza de audios o la segmentación de *pitch*[4]. [17]
- Clasificación de subgéneros: Mediante la búsqueda de parámetros melódicos definitorios de los cantes, que se codifican y se integran en un análisis automático. [15]

[4] *Pitch*: Propiedad perceptual del sonido que permite su colocación relativa en una escala de frecuencias.

Entre otras como:

- Análisis métrico de estructuras
- Detección de patrones melódicos mediante el análisis de piezas polifónicas [16]
- Análisis semántico
- Musicología computacional
- Identificación de cantante

En relación con el objetivo de este proyecto, en 2016, se desarrolló un algoritmo para la detección de falsetas de guitarra flamenca [38], que sirve como posible punto de partida, complemento y fuente de información útil, junto con [18] para el desarrollo del algoritmo propio.

1.4 Objetivos

La meta de este proyecto es crear un algoritmo de transcripción automática como herramienta útil tanto para el aprendizaje, como para el estudio de la guitarra flamenca. Proporcionando así, un soporte informático como primer paso para la única alternativa existente en la actualidad del flamenco: la transcripción manual, muy costosa y que requiere conocimientos, del flamenco y musicales, avanzados.

Estas transcripciones manuales dotan de gran información complementaria a la representación de las notas: digitación, dinámica y otros conceptos propios de la guitarra flamenca. Esta información es prácticamente imposible de transcribir automáticamente, por lo que acotamos nuestro proyecto a ofrecer una técnica complementaria y alternativa como primera fase a una transcripción completa optimizando el tiempo.

Adicionalmente, y con el fin de evaluar el algoritmo, crearemos una colección de anotaciones manuales, tanto para la delimitación las falsetas, como las transcripciones. Esta evaluación se analizará para determinar la precisión de algunos métodos que se implementarán, y cómo se comportan para según qué técnicas propias del Flamenco.

Uno de los objetivos es crear un prototipo de interfaz, que contenga todas las funcionalidades que se desarrollaran. El fin es proporcionar reproducibilidad a usuarios de todo tipo, familiarizados o no, con entornos de programación. Este prototipo se usará para la posterior implementación en proyectos futuros.

2. ENTORNO DE DESARROLLO Y ESTRATEGIA DE EVALUACIÓN

2.1 Herramientas utilizadas

Para el desarrollo del algoritmo de detección y transcripción de falsetas usaremos el lenguaje Python. En cuanto a entorno y herramientas complementarias se han utilizado las siguientes:

Integrated Development Environment (IDE) PyCharm: Entorno de desarrollo multi-plataforma para programación, especialmente en el lenguaje Python, pero también potencialmente útil para plataformas de desarrollo web como Django, Flask, Pyramid u otros como JavaScript, SQL o HTML/CSS. Permite el desarrollo mediante intérpretes remotos, además de un editor inteligente de Python, una gran colección de herramientas y un terminal incorporado, depurador gráfico, entre otras ventajas. Esta desarrollado por la compañía JetBrains bajo la licencia de Apache License. [19]

Librería Essentia: Biblioteca C++ de código abierto y multi-plataforma, para el análisis de audio y la recuperación de información musical basada en audio. Contiene una extensa colección de algoritmos reutilizables que implementan la funcionalidad de entrada / salida de audio, bloques de procesamiento de señal digital estándar, caracterización estadística de datos y un gran conjunto de descriptores de música espectral, temporal, tonal y de alto nivel. Desarrollado en el Music Technology Group (MTG) de la UPF. Además de estar disponible para Python, incluye extensiones que permiten usarse en plataformas como PureData o Matlab, incluyendo la opción de ser usado para web mediante una compilación cruzada a JavaScript. [20]

MELODIA: El complemento MELODIA estima automáticamente el tono de la melodía principal de una canción. Más concretamente, implementa un algoritmo que estima automáticamente la frecuencia fundamental correspondiente al tono de la línea melódica predominante de una pieza de música polifónica (o monofónica). Está incluido en la librería Essentia, además de estar en formato *Vamp Plug-in*. La utilidad principal será, además de extracción de melodía dentro del algoritmo, usar la propuesta visual que ofrece como *Vamp Plug-in* en Sonic Visualiser; así poder complementar gráficamente el material de la colección de audios. Desarrollado por J. Salamon and E. Gómez. [21]

Sonic Visualiser: Aplicación de código abierto para visualizar, analizar y anotar archivos de audio. Facilita el estudio y la visualización de grabaciones, mediante funcionalidades propias y la posibilidad ampliarlas incluyendo complementos que permiten un gran espectro de utilidades, relacionadas tanto con la extracción como con la visualización. Desarrollado en el *Centre for Digital Music, Queen Mary, University of London*. [22]

CANTE es una herramienta para la transcripción automática de cante flamenco, desarrollada en el contexto del proyecto de investigación COFLA [13]. El software extrae una representación de las notas correspondientes al cante en grabaciones polifónicas, desarrollado por [18]. Este es el punto de partida del algoritmo que proponemos, siendo la base para adaptarlo, sobre todo, potencialmente útil en la parte de detección y extracción de falsetas. El algoritmo tiene como entrada un archivo *.wav* y como salida un *.csv*, útil para ser visualizado en herramientas como *Sonic Visualiser*, incluyendo tiempo, duración y valor de *pitch*. Se compone de las siguientes fases:

I. Extracción de tonalidad vocal
   a) Selección de canal donde la voz sea predominante.
   b) Extracción de melodía predominante mediante MELODIA
   c) Filtrado de contornos con el fin de eliminar los que correspondan a guitarra, utilizando características tímbricas.

II. Transcripción
   a) Segmentación según características de contorno, o bien, relacionadas con el volumen.
   b) Cuantización de tono mediante estimación de afinación, probabilidad de *pitch class* e histogramas locales del tono.
   c) Post-procesado para conseguir la representación a nivel nota

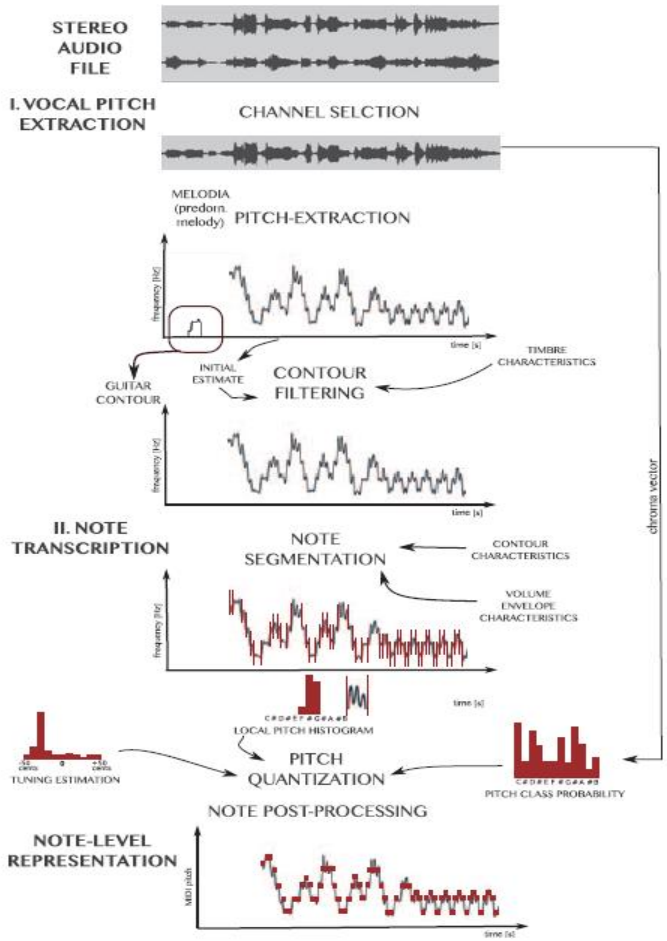


Fig 2.1 Diagrama gráfico del algoritmo PyCante [18]

2.2 Estrategia para la evaluación

Con el objetivo de evaluar, tanto la precisión total del algoritmo como algunos de los parámetros que usaremos en funciones internas, usaremos dos métodos independientes. Ambos, parte de la librería Python *mir_eval* [23], que nos permitirá evaluar de forma empírica las dos fases de algoritmo: extracción y transcripción. Dicha librería proporciona una forma transparente y estandarizada de evaluar sistemas de MIR.

Para evaluar la extracción de falsetas, utilizaremos *mir_eval.segment.detection()*: Considerando una falseta como un segmento, es decir, cada falseta tiene un tiempo de inicio y un tiempo final. Evaluaremos, por una parte, el número de falsetas encontradas, y por la otra, la precisión en los bordes al segmentar cada una de ellas.

Los parámetros de entrada que ajustaremos en esta función son: intervalos de referencia, intervalos estimados y tamaño de la ventana de tolerancia alrededor de la cual se considerará como correcto. En nuestro caso, ajustaremos el tamaño de la ventana a 4 segundos, teniendo en cuenta la ambigüedad y subjetividad de si consideramos las llamadas, cierres, remates y otros recursos como parte de la falseta o no.

Por ejemplo, la inclusión, o no, de los cierres o de las partes armónicas precedentes, ya que nuestro algoritmo elimina los contornos vocales, pero no sabemos si es relativamente preciso con las distintas fases dentro de todo el proceso de la guitarra solista.

Por lo tanto, como parámetro de referencia tendremos un vector (n,2) de dos columnas (inicio y final) con el intervalo correspondiente, y tantas filas como falsetas (n) se hayan encontrado previamente de forma manual. Los intervalos estimados, con el mismo formato explicado anteriormente, se crearán dentro del algoritmo, en la fase de segmentación de las falsetas.

En segundo lugar, evaluaremos la transcripción con el módulo *mir.eval.transcription.precision_recall_f1_overlap*(), encargado de comparar todas las notas de estimadas con las de referencia, midiendo las coincidencias entre ellas. Todas las notas se componen de un tiempo de inicio, uno de final y un valor de *pitch*.

Los criterios por defecto para que dos notas sean coincidentes, de acuerdo con MIREX, fijan que el tiempo de tolerancia del inicio de cada nota es de 50ms. En cuanto a frecuencia, la tolerancia es de un cuarto de tono. En nuestro caso, aumentaremos la tolerancia del tiempo de inicio a 0.1 segundos, dada la precisión relativa de la anotación manual.

En ambos casos, obtenemos métricas de relevancia, usadas para medir el rendimiento en sistemas de recuperación de información y reconocimiento de patrones, entre otros:

- Precisión: Ratio de número de instancias relevantes recuperadas entre el número de instancias recuperadas. Por lo tanto, cuanto más cerca esté de uno, más instancias recuperadas serán de relevancia.

$$Precisión = \frac{|\{instancias\ relevantes\} \cap \{instancias\ recuperadas\}|}{|\{instancias\ recuperadas\}|} \quad (1)$$

- Exhaustividad (*Recall*): Expresa la proporción de instancias relevantes recuperadas entre el total de instancias que son relevantes, independientemente si se recuperan o no. Si este valor se acerca a uno, se habrán encontrado todas las instancias relevantes de la base de datos.

$$Exhaustividad = \frac{|\{instancias\ relevantes\} \cap \{instancias\ recuperadas\}|}{|\{instancias\ relevantes\}|} \quad (2)$$

- Valor-F: Se define como una media armónica que determina un valor ponderado entre la precisión y la exhaustividad.

$$ValorF = 2 \cdot \frac{Precisión \cdot Exhaustividad}{Precisión + Exhaustividad} \quad (3)$$

Además, en el caso de la trascripción, obtenemos una medida más: *Average Overlap Ratio*, que determina la relación de duración del segmento el que se superponen las dos notas y el tiempo abarcado por las notas combinadas. Teniendo en cuenta el tiempo de inicio más temprano y el de final más lejano.

2.3 Conjunto de datos usado para el entrenamiento

Dado que mediremos la precisión independientemente para cada parte, usaremos dos colecciones de audios adecuadas para cada caso.

a) Colección de datos para la extracción

Para exportar al máximo esta fase, se debe realizar con un set formado por temas que respeten el proceso de diálogos clásicos entre cante y toque. Para ello hemos escogido 20 canciones de la discografía de Camarón de la Isla, en su primera etapa junto a Paco de Lucia (1969-1977). En el anexo 1 se especifica la anotación manual de los tiempos dónde hay falseta para cada uno de los archivos.

Como el tiempo mínimo de falseta será un parámetro de entrada que el usuario podrá escoger, limitaremos el tiempo mínimo a 15 segundos, con tal de evaluar la precisión en este caso concreto. Por lo tanto, todas las falsetas anotadas manualmente, respetan dicho tiempo mínimo.

b) Colección de datos para la transcripción

Para evaluar la fase de trascripción, crearemos un conjunto de falsetas con las anotaciones manuales correspondientes en formato MIDI. Resulta complicado encontrar audios de guitarra flamenca con suficiente calidad y que tengan una anotación manual en forma de archivo MIDI fiable. Estas anotaciones deben contener un tiempo de inicio, uno de final y un valor de *pitch*.

Dado que no se han encontrado estas anotaciones, hemos creado un conjunto de 10 falsetas con sus respectivos archivos MIDI. La mayoría de ellas no tienen acompañamiento, con tal de evaluar desde el caso más sencillo hasta alguno más complejo.

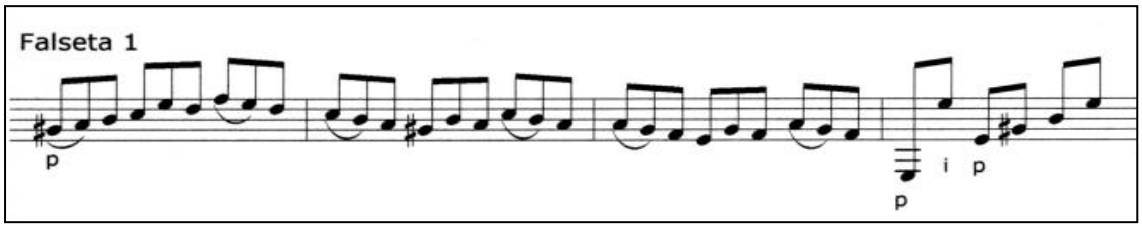
Analizaremos técnicas usadas en las falsetas, relacionadas con la ejecución o la velocidad de la línea melódica, determinando, como funciona la transcripción en casos concretos y en el caso general.

De esta manera, mediante la obtención de valores métricos evaluativos, tendremos herramientas futuras mejoras o posibles métodos complementarios que mejoren casos concretos. A continuación, se muestra el proceso de obtención de partitura y proceso de conversión a MIDI para cada falseta:

1. Obtención de audio y partitura para cada uno de los elementos en [34][35]. Edición de la partitura, si es necesario, en la herramienta MuseScore, y exportación a formato MIDI. El tempo en el cual se exporta se ajusta manualmente, en algunas ocasiones, con la ayuda de la herramienta TAP de Ableton Live.

2. Dado que el MIDI exportado es una partitura, por lo tanto, tiene un tempo y una precisión exacta, debemos alinearlo con el audio original. Aunque el tempo esté ajustado, a nivel interpretación, hay variaciones que hacen que la precisión no sea exacta. Para alinearlo, hemos usado Sonic Visualiser, de manera que se van ajustando los segmentos manualmente para una mayor precisión. A continuación, se muestra el proceso:

Exportamos la partitura correspondiente en formato MIDI, y lo visualizamos en Sonic Visualiser.



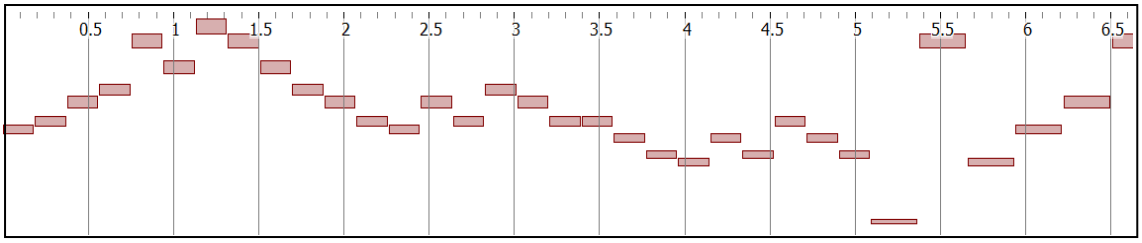


Fig. 2.2 Visualización del proceso de anotación manual I

Paralelamente, visualizamos el audio que contiene la falseta. Como ayuda complementaria para la tarea de alineación, computamos el flujo espectral[5] y los tiempos de inicio [25]. A continuación, alineamos manualmente el MIDI anterior:

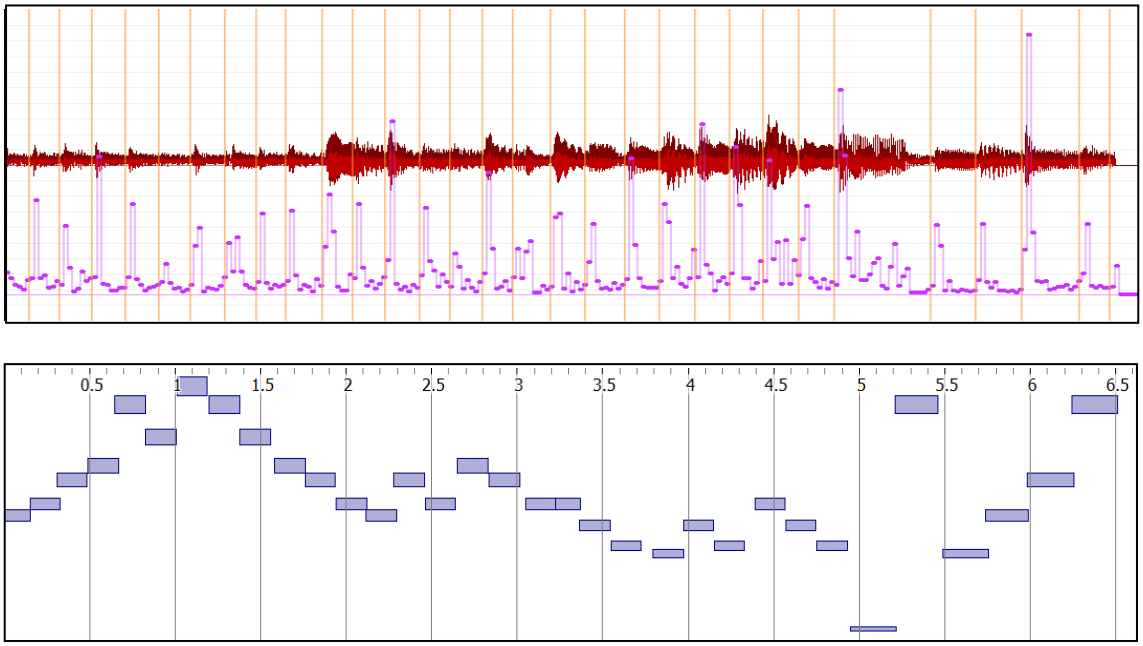


Fig 2.3 Visualización del proceso de anotación manual II

Dentro del conjunto de falsetas analizadas se encuentran:

- Seis falsetas básicas de soleá por arriba, utilizadas por varios guitarristas clásicos a lo largo de la historia. Aparecen a lo largo de la discografía flamenca antigua y se basan en las técnicas rudimentarias de la guitarra flamenca como el pulgar, los rasgueados y los arpegios. Todas ellas monofónicas y de duración entre 6 y 7 segundos [34].

- Una falseta por bulerías, interpretada por Manolo de Huelva. Concretamente, fue grabada junto a Canalejas de Puerto Real, dónde el primer tercio es "Santa Cruz es un barrio que hay en Sevilla", tras el cual podemos encontrar la falseta en cuestión [34]. La grabación es monofónica y tiene una duración de 9 segundos.

[5] Flujo espectral: Muestra cómo cambian las densidades espectrales entre *frames* consecutivos.

- Fragmento monofónico introductorio del tema “Nuevo Dia” de Lole y Manuel, perteneciente al álbum “Nuevo Dia: El origen de la leyenda” publicado en 1975. Interpretación del fragmento y composición por Manuel Molina. Duración de 30 segundos.

- Fragmento de 26 segundos de la pieza por Alegrías “Punta y Tacón” interpretada por Amir-John Haddad “El Amir”, perteneciente al álbum “9 Guitarras” grabado en 2012. A diferencia del resto, este elemento es polifónico y además de contener guitarra, contiene palmas. Podemos encontrar técnicas muy comunes como la *alzapúa* o el rasgueo.

3. DESARROLLO DEL ALGORITMO

A continuación, se detallarán los pasos realizados en el algoritmo para la extracción y transcripción de falsetas de guitarra en señales de audio polifónicas. Como hemos explicado anteriormente, en el flamenco clásico, se produce un proceso comunicativo de diálogo entre instrumentos. Sobre todo, entre la guitarra y la voz, de manera que cuando se está cantando, la guitarra tiene menos protagonismo, simplemente acompaña y marca el ritmo. Mayoritariamente, cuando el cante termina, es decir, cuando dejamos de tener presencia vocal en el audio, aparece la falseta de guitarra como protagonista.

Este detalle nos será de gran utilidad para desarrollar la primera parte del algoritmo, que consiste en seleccionar y extraer solo aquellas partes clasificadas como falseta del audio original. Posteriormente, una vez seleccionadas estas partes, y descartadas las partes vocales, podremos proceder a la transcripción.

El desarrollo del algoritmo es una adaptación de la técnica usada en la transcripción automática de cante, desarrollada en 2015 por N.Kroher y E.Gómez: *"Automatic Transcription of Flamenco Singing from Polyphonic Music Recordings"[18]*. Este algoritmo consigue descartar las partes donde la guitarra es protagonista, transcribiendo sólo el cante. Como hemos visto anteriormente, gracias al proceso de diálogo, podremos adaptar esta propuesta para modificarla, aplicar conceptos a la inversa y conseguir algunos de nuestros objetivos.

Los pasos de nuestro algoritmo, también ilustrados en la figura 3.1, son los siguientes:

1. Extracción:
    a. Separación de canales y selección de canal con menos presencia vocal.
    b. Extracción de la melodía (f0) predominante mediante MELODIA (requiere librería *Essentia*)
    c. Filtrado gaussiano a partir de bandas de Bark. Clasificación de *voiced* y *unvoiced frames* y eliminación de todos los *unvoiced* (los segmentos candidatos a falseta tendrán valor 0 en el vector f0)
    d. Delimitación de las falsetas y extracción: Buscamos segmentos de 0 consecutivos en f0 (de una longitud mínima que el usuario puede escoger) y los delimitamos.
    e. Relación de esos segmentos con el audio original: Falsetas extraídas.
        i. Output 1: [input_Lfalsetas.wav] con todas las falsetas (canal izquierdo)
        ii. Output 1: [input_Rfalsetas.wav] con todas las falsetas (canal derecho)
        iii. Output 2: [input_timefalsetas.txt] fichero de texto donde se indican los tiempos (en segundos) dónde empieza y acaba cada falseta.

2. Transcripción
    a. Extracción de melodía mediante *MultiPitchKlapuri* (Essentia)
    b. Detección de tiempos de inicio (*onsets)* de cada nota, mediante *OnsetDetection* (Essentia)
    c. Segmentación: Definición de *onsets* y *offsets.*
    d. Estimación del *pitch* y etiquetado
3. Post-procesado: re-escalado de *pitch* y control de *outliers*[6] temporales. Creación del archivo .csv y MIDI

A continuación, se especifican las entradas, parámetros variables y salidas del algoritmo:

- Entrada: Archivo de audio, estéreo o mono, a 44.100Hz y 16 bits PCM. Puede ser polifónico o monofónico, y podrá contener fragmentos que no sean únicamente de guitarra.
- Parámetros:
    - *'acc' (booleano)*: 'True' si se espera acompañamiento, 'False' para falsetas únicamente de guitarra, ya delimitadas.
    - *'recursive' (booleano):* 'True' para ejecutar el algoritmo para todos los archivos de un mismo directorio.
    - *'tmin'(entero):* Tiempo mínimo, en segundos, para considerar un segmento como falseta. Todas las falsetas con t<tmin serán eliminadas.
    - *'method' (entero)*: '1' para escoger el valor de *pitch* calculando la mediana de las frecuencias. '2' para escogerlo mediante el histograma local de frecuencias.
    - *'tone' (cadena de caracteres):* 'Ef' para ajustar los valores de *pitch* teniendo como referencia la escala frigia *mayorizada* de Mi. 'Af' para tener como referencia la de La frigia *mayorizada* y 'E' para usar la escala de Mi mayor. 'default' para no ajustar la escala.
    - *'transposed' (entero):* En caso de que el audio a transcribir esté transportado y usemos algún ajuste de escala, de 0 a 7, la cantidad de semitonos a transportar en la escala de referencia.
- Salida:
    - <filename>_timefalsetas.txt
    - <filename>_Lfalsetas.wav sólo si el archivo de entrada es estéreo.
    - <filename>_Rfalsetas.wav sólo si el archivo de entrada es estéreo.
    - <filename>_Mfalsetas.wav sólo si el archivo de entrada es mono.
    - <filename>_ notes.csv, con la información de la transcripción.
    - <filename>_MIDI.mid con el archivo tipo MIDI de la transcripción.

El siguiente diagrama contiene un resumen gráfico del algoritmo, el cual se explicará detalladamente en los siguientes apartados:

[6] *Outliers*: valores atípicos, fuera de rango, que deben ser eliminados

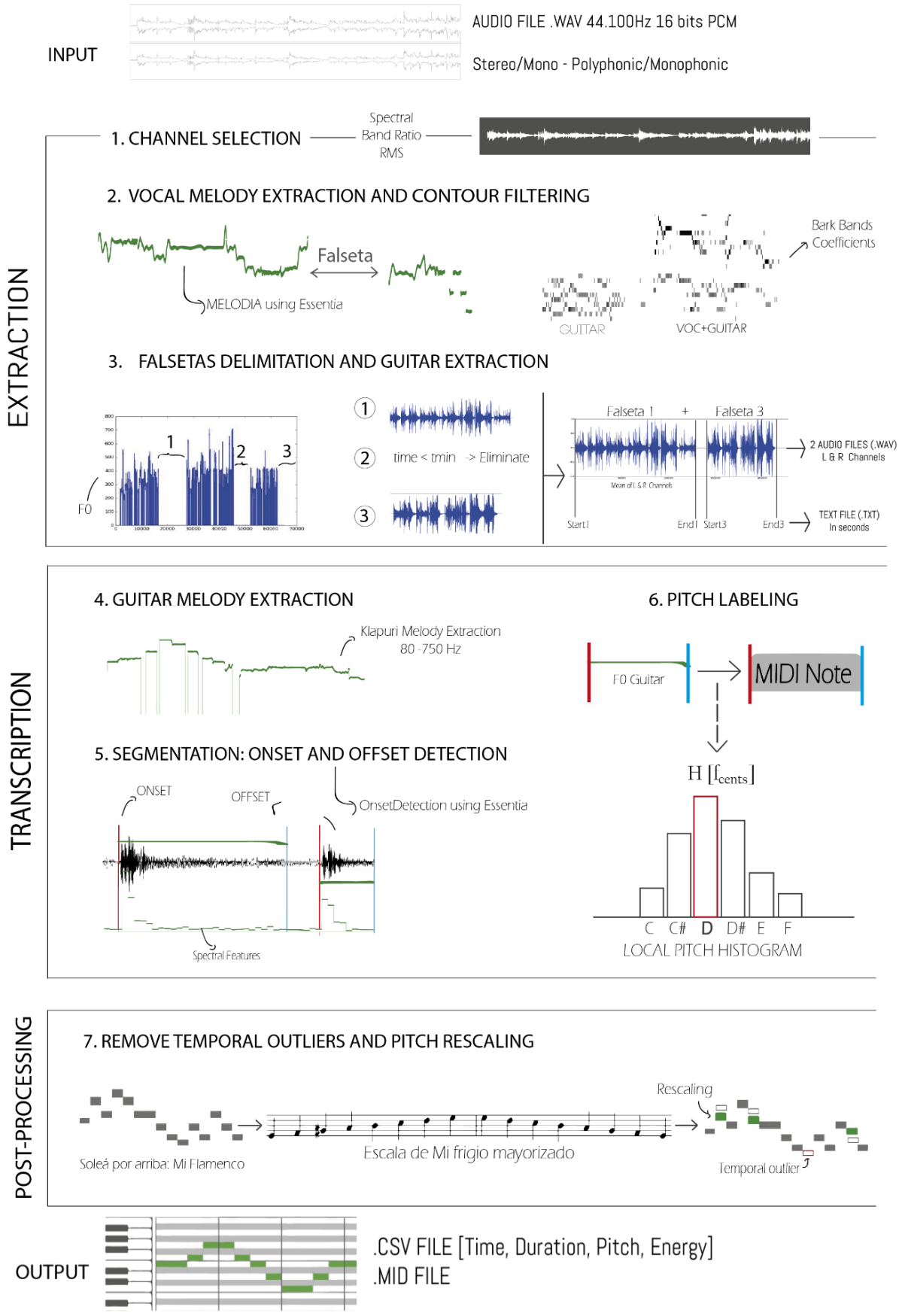


Fig 3.1 Diagrama completo del algoritmo propuesto: PyToque

3.1 Extracción de falsetas

Como hemos visto anteriormente, aprovechamos el proceso comunicativo que encontramos en multitud de canciones del flamenco clásico. Como hemos visto en capítulo de la evaluación, la colección utilizada en esta fase está compuesto por 23 canciones, la mayoría de las cuales, pertenecen a la discografía de Camarón de la Isla, en su fase inicial junto con Paco de Lucia (1969-1977).

Dicha etapa, representa perfectamente el proceso de diálogo entre cante-toque que mencionamos anteriormente, siendo así, una opción óptima para hacer las pruebas de extracción de guitarra. A continuación, se explicarán con detalle, cada una de las fases de esta primera parte del algoritmo.

a) Selección de canal

Normalmente, si el audio es estéreo, la voz predominante se encuentra en uno de los canales. Sin embargo, en muchas canciones, pueden actuar dos guitarras, una encargada de hacer la línea melódica principal y la otra que acompaña armónicamente. Cada una de ellas con más presencia en un canal que en el otro.

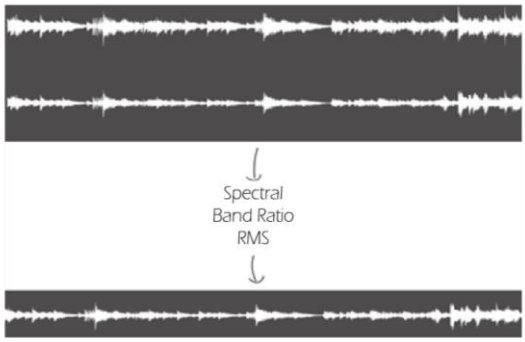


Fig. 3.2 Visualización de la selección de canal

Si hablamos de rango de frecuencias, como podemos observar en el siguiente espectrograma, en segmentos vocales hay un incremento de densidad entre 500 Hz y 6kHz. Por lo tanto, si queremos extraer los segmentos vocales, escogeremos el canal que tenga mayor presencia media de dichas frecuencias. De esta manera conseguiremos eliminar información que puede restar precisión al algoritmo.

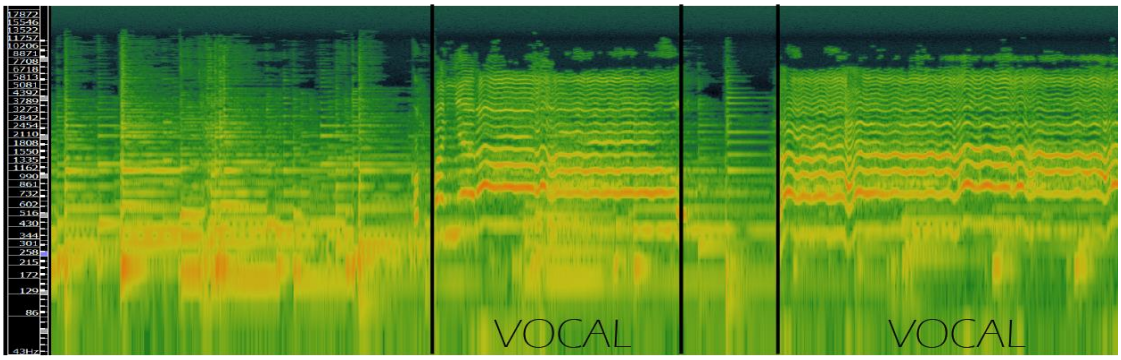


Fig 3.3 Visualización de densidad de frecuencias mediante espectrograma

La selección automática se basa en la distribución espectral de energía. En nuestro caso, escogeremos el que tenga menos presencia de las frecuencias mencionadas anteriormente, ya que nos interesa restar contenido vocal para poder tener más precisión en el momento de acotar la falseta. Eliminando así, posibilidades de obtener contornos vocales ("jaleos") muy cortos que se producen durante la falseta y pueden entorpecer o cortar una falseta en dos o más partes.

Si encontramos el caso de las dos guitarras, mayoritariamente, la voz que se encarga de la línea melódica (la que nos interesa transcribir) se encuentra en el mismo canal de la voz predominante. En este caso, escogeremos, igualmente, el otro canal, ya que la guitarra de acompañamiento se suele quedar en los rangos de frecuencia más bajos. En cambio, la melódica puede solapar frecuencias altas que pertenecen a lo que hemos considerado como cante. Conseguiremos, después de realizar la extracción de la melodía predominante y el filtrado, tener un segmento mucho más acotado y claro de la falseta como se indica a continuación:

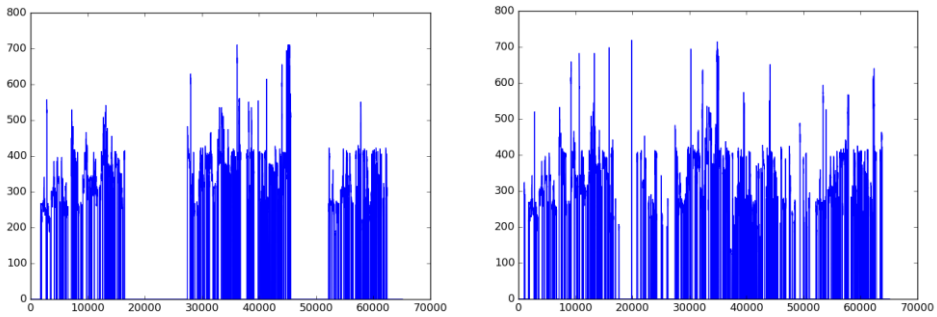


Fig 3.4 Visualización de la línea melódica (a) seleccionando el canal con menos energía y (b) seleccionando el canal con más energía

Realizaremos todos los pasos para encontrar el canal con más presencia vocal, pero escogeremos el otro:

1. Siendo la frecuencia de muestreo es de 44.1 kHz, aplicamos una STFT de 4096 multiplicada por una *Hanning window*, m=2 como factor de *zero padding* [7]y un tamaño del salto (*hop size*) de 1024 muestras.

2. Fijamos las frecuencias límite para voz y para guitarra:
   freqGuitLow (f_{11}) = 80.0 Hz; freqVocLow (f_{21}) = 500.0 Hz
   freqGuitHigh (f_{12})=400.0 Hz; freqVocHigh (f_{22}) = 6000.0 Hz

3. Mediante la siguiente operación, conseguimos los índices k correspondientes a cada frecuencia central f. Usando los valores definidos en el punto 1 para los parámetros m, N, fs:

$$k(f) = round\left(\frac{f \cdot m \cdot N}{fs}\right) \quad (4)$$

[7] Zero Padding: Técnica que consiste en añadir ceros al final de la señal en dominio temporal, que se traduce a una interpolación de muestras en dominio frecuencial.

4. Encontraremos la presencia vocal, realizando una ratio de banda espectral, dividiendo la suma de magnitudes en la banda superior (cante) e inferior (guitarra).

$$S[n] = 20 \cdot log10 \left(\frac{\sum_{k(f21)<k<k(f22)} |\dot{X}[k,n]|}{\sum_{k(f11)<k<k(f12)} |\dot{X}[k,n]|} \right) \quad (5)$$

5. Eliminamos parte del volumen excesivo dividiendo el espectro de magnitud del *frame* por su valor máximo en cada valor *n*. Una vez tenemos la relación de banda espectral en cada muestra *n* y su media total para cada canal independiente. Escogeremos el que tenga una media inferior, acorde a las conclusiones anteriores.

b) Extracción de la melodía predominante

Una vez tenemos la selección de canal realizada, el siguiente paso es extraer la melodía predominante. En nuestro caso, buscaremos la línea predominante de la voz, al igual que sucede en [18]. Esto nos ayudará, posteriormente, a delimitar las zonas donde hay falseta dado que, hasta el momento de la recuperación del audio original, nuestra intención será marcar las falsetas como secuencias de 0.

Este algoritmo estima posibles valores de *pitch* en cada muestra basándose en una suma de armónicos. Seguidamente, agrupa en contornos continuos de *pitch* y tiempo usando principios de *auditory streaming:* estimando el *pitch track* correspondiente a la melodía predominante perceptual. Además, puede determinar si la melodía principal es presente en algún *frame* o no, permitiendo clasificar cada *frame* en *melody frames* o *non-melody frames.* Asumimos "contorno" como una secuencia de *melody frames* consecutivos.

Para extraer esta información usaremos el método [21] incorporado en la librería Essentia, aunque otros métodos de *pitch track* también pueden ser usados. Incluso, usar el resultado del plugin MELODIA exportado con Sonic Visualiser, como archivo de entrada, omitiendo este paso.

El resultado, entonces, es un vector f0[n] con un solo valor de *pitch* para cada muestra *n* de todos los *melody frames.* Seguiremos los parámetros sugeridos en [18]:

- *Analysis window = 4096 samples*
- *Hop Size = 128 samples*
- *Sample Rate = 44.1 kHz*

El rango frecuencial escogido, correspondiente al cante flamenco, es 120Hz – 720 Hz. Ajustaremos el *voicing threshold* a 0.2, que corresponde a la relación relativa del umbral, para considerar si un contorno pertenece, o no, a la melodía principal.

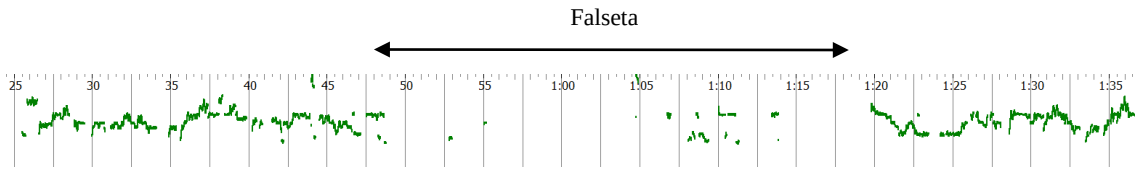


Fig 3.5 Visualización de la línea melódica mediante MELODIA vía Sonic Visualiser

c) Filtrado de contornos: clasificación *voiced* y *unvoiced*

Siguiendo el método propuesto en [18], en esta fase del algoritmo, nuestro objetivo será clasificar cada *frame* como *voiced*, si tiene contenido vocal, o *unvoiced* si no lo tiene. Por lo tanto, en este último tipo de *frame*, encontraremos los que pueden ser candidatos para pertenecer a una falseta.

Fig 3.6 Ilustración del diagrama de PyToque: Detección y delimitación de falsetas.

Una vez clasificados estos *frames*, podremos buscar segmentos consecutivos de *melody frames*, es decir, contornos. Posteriormente, se eliminarán los clasificados como *unvoiced* para poder proceder, seguidamente, a la extracción de estos contornos en la siguiente fase del algoritmo.

El principio en el que basaremos esta clasificación es que la guitarra y la voz tienen características espectrales diferentes. Basado en experimentos realizados previamente por expertos, se usará una distribución Gaussiana de la energía en las 12 primeras bandas de *Bark*, para discriminar entre los dos tipos de *frame* que queremos clasificar.
También asumimos que la mayoría de *frames*, serán vocales, tal y como especifican Kroher et al. (2016):

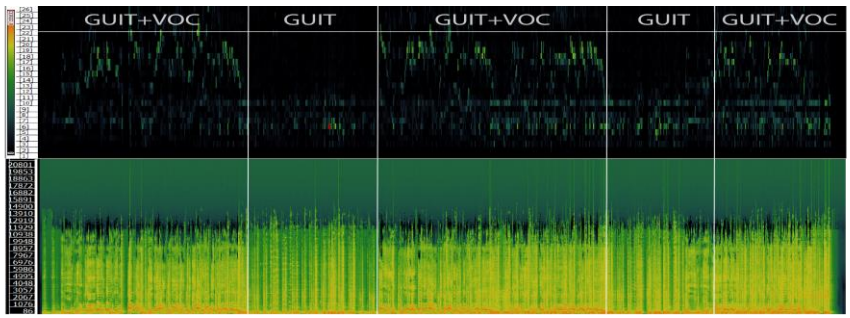


Fig 3.7 Coeficientes de Bark y espectrograma vía Sonic Visualiser

La escala de *Bark* es una medida subjetiva de audio para la distinción del timbre, cualidad esencial que caracteriza un sonido, al igual que el tono, la duración e intensidad. Es una escala psico-acústica, y se corresponde directamente, en su rango de 1 a 24, con las bandas críticas del oído. Las frecuencias correspondientes a estas bandas son: 0 Hz, 50 Hz, 100 Hz, 150 Hz, 200 Hz, 300 Hz, 400 Hz, 500 Hz, 630 Hz, 770 Hz, 920 Hz, 1080 Hz y 1270 Hz.

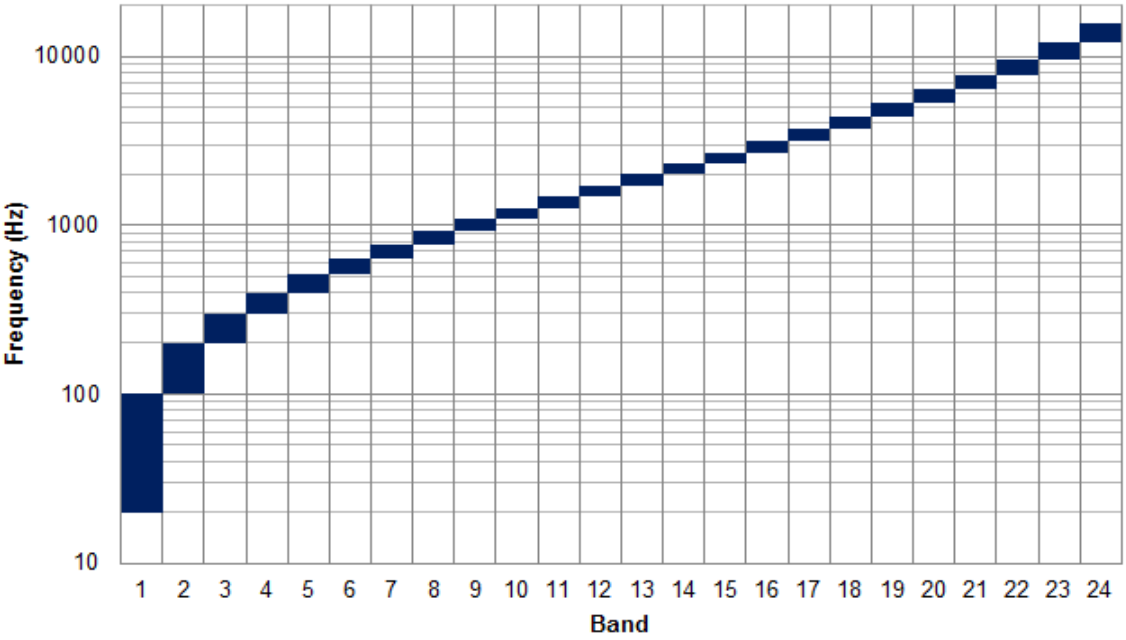


Fig 3.8 Escala de Bark[8]

El primer paso es computar la energía en cada una de las 12 primeras bandas de *Bark* para cada *frame* n, de manera que, obtenemos un vector x[n] multidimensional de 12 variables con cada una de ellas. Dichas energías se computan con una ventana de tamaño 1024, un salto de 128 muestras y una frecuencia de muestreo de 44.1kHz, aplicando un sumatorio de los valores de la magnitud del espectro para cada muestra dentro de la banda correspondiente:

$$B[n,m] = \sum_{k(f1,m)<k<k(f2,m)} |X[k,n]|^2 \qquad (6)$$

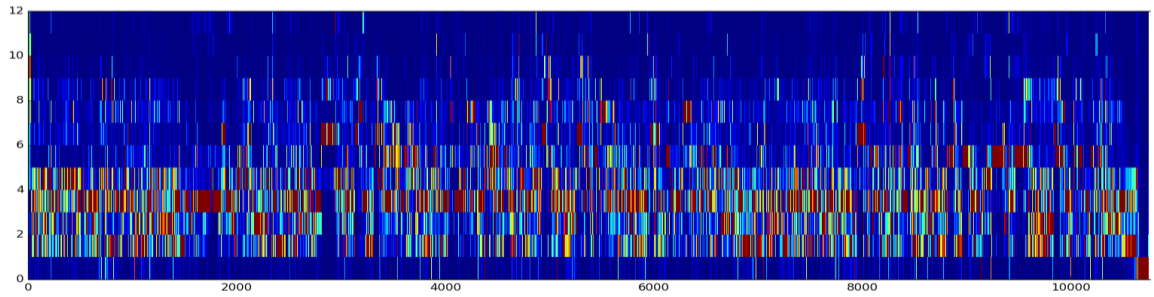


Fig 3.9 Representación de las primeras 12 bandas de Bark para "A los santos del cielo (seguiriya)"

[8] https://commons.wikimedia.org/wiki/File:Bark_scale.png

Una vez tenemos dichas energías hospedadas en x[n], se realiza una primera estimación y se etiquetan como *voiced*, aquellos *frames* donde la melodía predominante (f0) coincida con la melodía vocal. Por lo tanto, los *melody frames* serán clasificados como voiced, y los *non melody frames* serán clasificados como *unvoiced*.

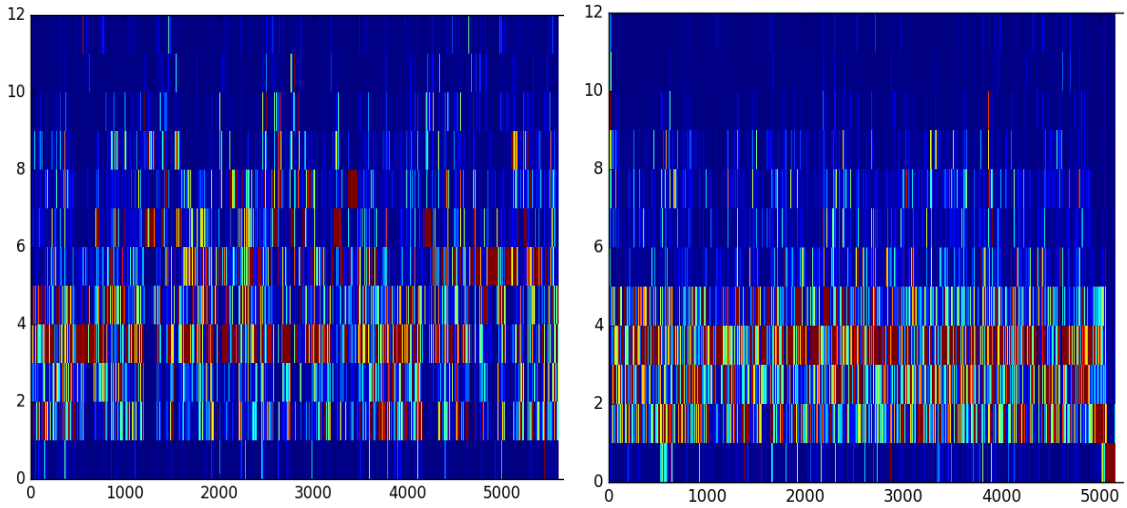


Fig 3.10 Representación de las primeras 12 bandas de Bark de los frames (a) clasificados como "voiced" y (b) clasificados como "unvoiced" de "A los santos del cielo (seguiriya)".

Después de esta primera estimación, se extraen la media y la covarianza para cada *set*, y se aplica a cada uno por separado, una distribución Gaussiana multi-variable única. Obteniendo así, una probabilidad para cada valor del vector x[n], que clasificaremos asignando un valor binario, dependiendo de la probabilidad que sea más mayor.

Con tal de eliminar fluctuaciones rápidas, se aplica un filtro binario de media móvil para la suavización de la detección vocal. Después de transformar estos valores a secuencias binarias, buscaremos segmentos que, posteriormente, evaluaremos de la siguiente forma:

$$\sum_{d1}^{d2} v[n] \begin{cases} = 0 \; eliminate \\ > 0 \; retain \end{cases} \qquad (7)$$

Si todos los valores dentro de un segmento *d* suman 0, se eliminan. Estos segmentos son los que nos interesa tener delimitados correctamente, ya que serán correspondientes a una falseta. En cambio, si los valores suman por encima de 0, se mantienen, perteneciendo al grupo de contornos vocales.

El resultado es el vector f0 con los segmentos de falseta anulados. Este formato nos será potencialmente útil, porque en el siguiente paso, sólo se deben extraer los segmentos que tengan valor 0 y relacionarlos con el audio original para obtener la falseta.

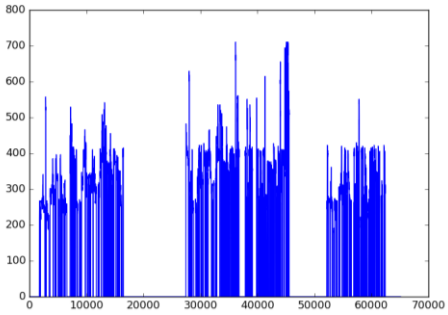


Fig 3.11 Visualización de la línea melódica (f0) después de la fase de filtrado de contorno.

La razón por la cual no eliminamos la guitarra y si la voz directamente es porque durante el proceso de extracción de melodía, perdemos muestras que nos ayudan a delimitar con precisión. Después de algunos experimentos, llegamos a la conclusión de que el resultado más óptimo se obtiene, primero, detectando los contornos vocales y después asumiendo que el resto corresponderá a falsetas.

d) Extracción, delimitación de la guitarra y escritura de las falsetas

El objetivo principal de esta fase es extraer los segmentos clasificados como *unvoiced* de la fase anterior, relacionarlos con el audio original y obtener las falsetas.

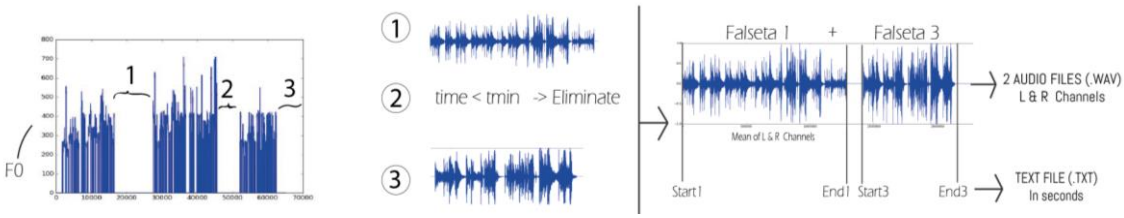


Fig 3.12 Fragmento del diagrama de PyToque: Extracción, delimitación y escritura de falsetas.

Primeramente, se vuelven a buscar segmentos en el vector f0 que sale de la función anterior, esta vez, con valor 0, equivalentes a nuestros candidatos a falsetas. Para clasificar un segmento como falseta debe cumplir dos condiciones:

- Todas las muestras del segmento encontrado tienen que sumar 0.
- El número de muestras tiene que ser superior a un valor asociado a un tiempo mínimo. Este tiempo, expresado en segundos, se encuentra como valor de entrada del algoritmo, por lo tanto, el usuario lo puede modificar. La utilidad de esta funcionalidad es poder filtrar las falsetas que queremos extraer mediante el tiempo, pudiendo así, omitir ciertas falsetas demasiado cortas o que pueden no ser consideradas como tal.

Una vez tenemos la delimitación de los segmentos clasificados como falsetas, relacionamos el valor de comienzo y de final con el audio original, recuperando la información inicial. Las salidas de esta función son:

- Dos archivos *.wav*, uno por cada canal, con todas las falsetas que se han localizado. En caso de que el archivo de audio no sea estéreo, la salida será un único archivo *.wav*.

- Un fichero *.txt* con el tiempo de comienzo y final de cada falseta dentro del audio original, expresado en segundos. Esto nos ayudará, con mayor facilidad, a evaluar los resultados de este algoritmo.

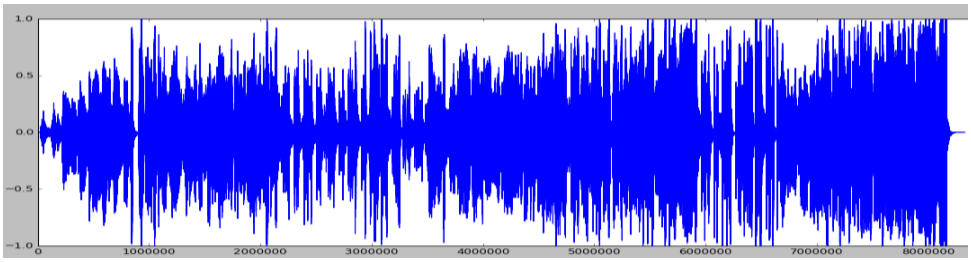


Fig. 3.13 Forma de onda del audio original (mono)

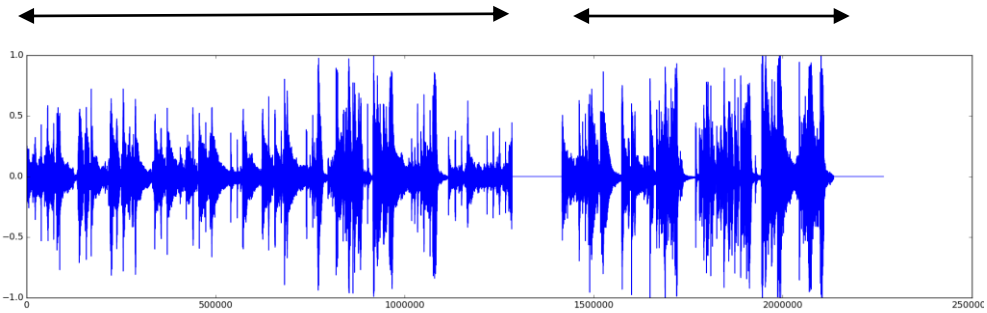


Fig. 3.14 Forma de onda de la falseta 1 + falseta 2

3.2 Transcripción de falsetas

Una vez tenemos la extracción de las falsetas, procedemos a la transcripción de las mismas. Utilizaremos las salidas de la fase anterior para realizarla, usando métodos que adaptaremos en función de si queremos una transcripción polifónica o monofónica.

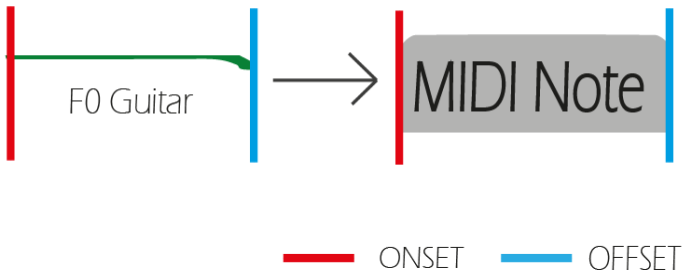


Fig. 3.15 Principal objetivo de la fase de transcripción.

Para crear el algoritmo y evaluarlo, asumiremos una transcripción monofónica en el caso de que no esperemos acompañamiento. Es decir, cuando indiquemos en el parámetro de entrada que no hay que eliminar contornos de voz.

En caso contrario, asumiremos todo el proceso anterior, y en la transcripción podremos tener, en caso de que exista, más de una nota a la vez. De esta manera podremos evaluar los dos métodos, además de cual es más preciso y óptimo para cada caso.

La idea principal de esta fase es analizar la línea melódica mediante *pitch tracking*, definir la segmentación de cada nota y etiquetar cada una de ellas con el valor de *pitch* correspondiente:

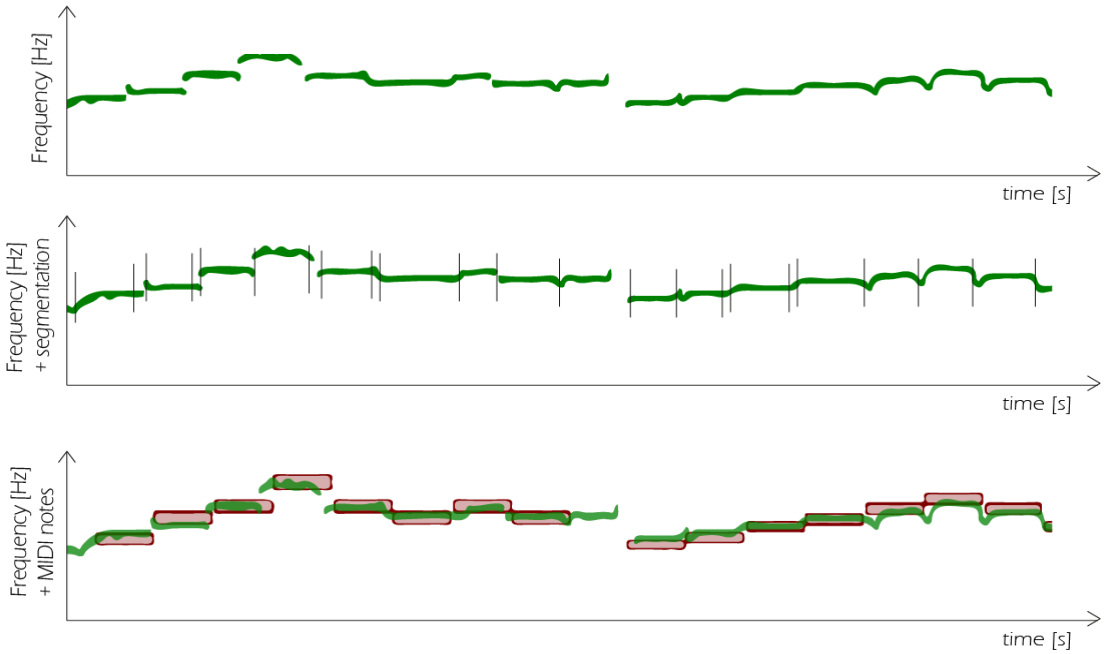


Fig. 3.16 Visualización gráfica de las fases de transcripción.

a) Extracción de melodía

En primer lugar, sabemos que la salida de la fase anterior, en el caso de entrada estéreo, serán dos archivos de audio con las falsetas, uno para cada canal. Por lo tanto, para proceder con más facilidad a la trascripción, haremos la media de los dos canales para crear un audio de un sólo canal, con el que trabajaremos a partir de ahora.

Para la extracción melódica utilizamos *MultiPitchKlapuri [36]*, que tiene como entrada las muestras del audio en formato 44.1Khz y como salida los valores de frecuencia estimados, calculados con un *hop size* de 128 muestras.

Esta estimación calcula la fuerza de un candidato a f0 como la suma ponderada de las amplitudes de sus armónicos parciales. En concreto, la fuerza o *salience* $s(\tau)$ de un periodo candidato τ se calcula como:

$$s(\tau) = \sum_{m=1}^{M} g(\tau, m)\left|Y(f_{\tau,m})\right| \qquad (8)$$

Donde $f_{\tau,m} = mf_s/\tau$ es la frecuencia del armónico parcial m del candidato a F0 f_s/τ, f_s es la frecuencia de muestreo, y la función $g(\tau, m)$ define la ponderación parcial de m en un periodo τ en la suma. $Y(f)$ es la STFT de señal *whitened (*operación de preprocesamiento) del dominio temporal.

La *salience function* es similar a MELODIA [21], pero esta, a diferencia de *Klapuri*, sólo usa los picos espectrales en la suma, para descartar valores espectrales que son menos fiables debido al enmascaramiento o al ruido.

Klapuri ofrece tres métodos [36] dónde el primero es el más sencillo y directo: Evaluar $s(\tau)$ para un rango de valores de τ y extraer el número deseado de máximos locales más altos de este. El segundo método usa un enfoque iterativo de estimación y cancelación, donde la máxima $s(\tau)$ se usa para estimar F0 y luego, antes de pasar a la siguiente estimación, se anula. El tercer método estima todos los F0s a la vez.

La función usada es la misma para transcripción monofónica y polifónica, pero los parámetros y el tratamiento posterior de las muestras es diferente:

- Transcripción monofónica: Como parámetro de *Klapuri*, usaremos un rango acotado de frecuencias entre 80-750Hz, correspondiente al de la guitarra. El valor del *threshold*[9] será de 50 dBs y los demás parámetros están ajustados por defecto:

  binResolution: *real ∈ (0, ∞), default = 10*
  frameSize: *integer ∈ (0, ∞), default = 2048*
  harmonicWeight: *real ∈ (0, 1), default = 0.8*
  hopSize: *integer ∈ (0, ∞), default = 128*
  magnitudeCompression: *real ∈ (0, 1], default = 1*
  magnitudeThreshold *integer: ∈ [0, ∞) = 50*
  maxFrequency: *real ∈ [0, ∞) = 750* Hz
  minFrequency: *real ∈ [0, ∞) = 80* Hz
  numberHarmonics *integer: ∈ [1, ∞), default = 10*
  referenceFrequency: *real ∈ (0, ∞), default = 55*
  sampleRate: *real ∈ (0, ∞), default = 44100*

  La salida de esta función será el vector *guitF0*, con los valores de *pitch* estimados en cada salto de 128 muestras. En este caso, guardaremos solo el primer valor de cada salto, de manera que tengamos solo una nota por muestra analizada.

[9] *Threshold:* Valor umbral o límite

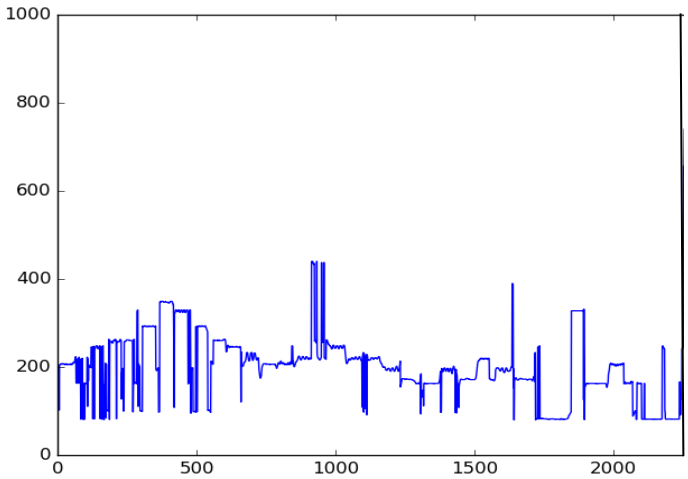


Fig. 3.17 Extracción de la línea melódica (f0) en el caso monofónico mediante Klapuri, usando el rango 80-750Hz.

- Transcripción polifónica: En este caso, usaremos los mismos parámetros que en el anterior, pero no acotaremos las frecuencias en el rango de la guitarra. Lo acotaremos entre 20-2500Hz, para prevenir un posible problema de sincronización con el audio original en la fase de segmentación.

  Cuando *Klapuri* detecta una frecuencia fuera del rango, esta no se guarda en el vector, por lo tanto, en las siguientes fases podemos tener problemas al relacionar las muestras con el original. Entonces, eliminaremos dichas muestras que están fuera de rango posteriormente, o simplemente no las etiquetaremos.

  En esta fase, guardaremos en cada muestra analizada, como mucho, dos valores de estimación de *pitch*: las que estén dentro del rango de 80-750Hz, o en el caso de que solo haya una y esté fuera del rango, la guardamos, pero no la etiquetaremos en la última fase. Con este método, además, eliminaremos posibles *onsets* erróneos, como palmas u otros instrumentos.

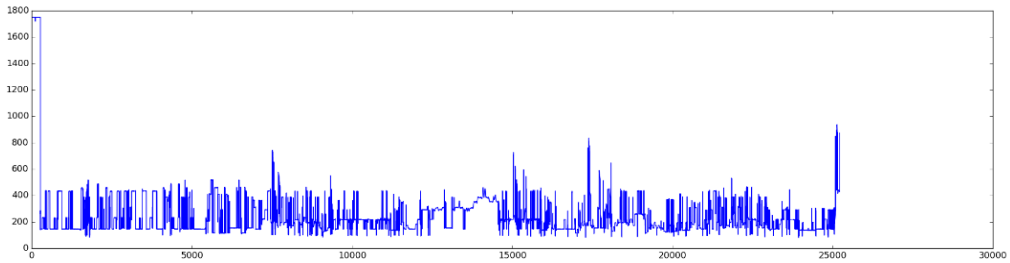


Fig. 3.18 Extracción de la línea melódica (f0) mediante Klapuri

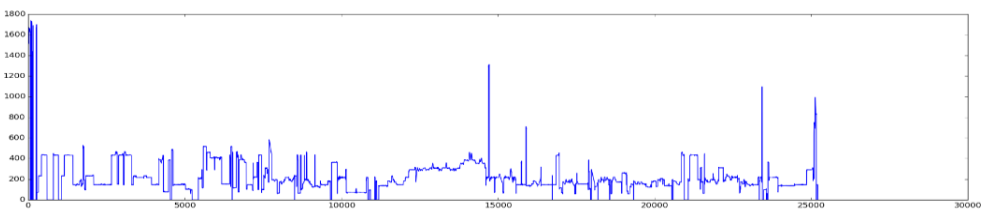


Fig. 3.19 Extracción de la línea melódica (f0) mediante MELODIA

b) Detección de ataques

En esta fase analizaremos los ataques del audio para determinar la existencia de las notas a transcribir. Utilizaremos la función *OnsetDetection [26]* de la librería [20], que dispone de seis métodos diferentes para encontrarlos:

- 'HFC' *High Frequency Content*: Detecta contenido de altas frecuencias, útil para detectar eventos de percusión.

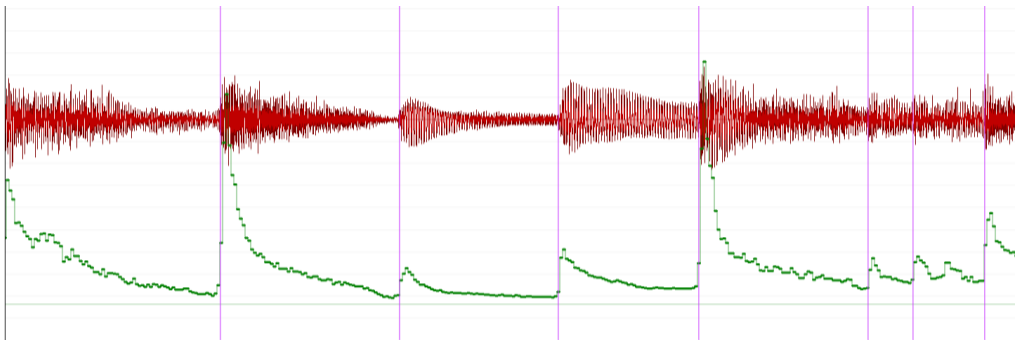


Fig. 3.20 Ataques detectados por características de contenido de altas frecuencias.

- 'complex': Mide diferencias espectrales [27] en el dominio complejo, teniendo en cuenta cambios en magnitud y fase. Causadas por un cambio de tono, se considerarán como ataque, cambios significativos en la energía de la magnitud del espectro y/o desviaciones inesperadas de la fase.

- 'complex_phase': Versión simplificada del método anterior, teniendo en cuenta cambios en la fase ponderados por la magnitud [28]. Útil para sonidos tonales, pero no tanto para eventos percutidos.

- 'flux' *Spectral Flux*: Detección del flujo espectral, teniendo en cuenta los cambios en la magnitud del espectro.

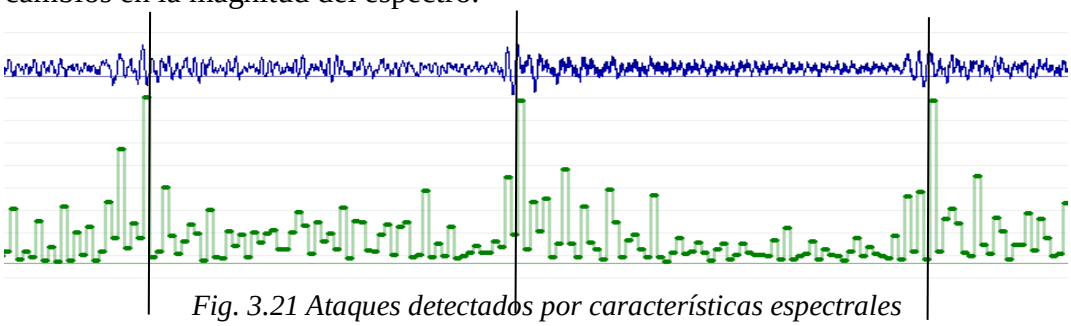


Fig. 3.21 Ataques detectados por características espectrales

- 'melflux': Similar a 'flux' pero usando cambios de energía en las bandas de frecuencia correspondientes a la escala Mel del espectro [29].
- 'rms': Midiendo el cambio del RMS[10] de las magnitudes de los espectros [30].

[10] RMS: (*Root Mean Square): Media cuadrática*

En nuestro caso, aunque las cuerdas de guitarra son percutidas, usaremos un algoritmo que esté centrado en el flujo espectral. De esta manera, si ignoramos los métodos centrados en el contenido en altas frecuencias, la probabilidad de detectar eventos erróneos relacionados con la percusión, como las palmas, será menor. También, debemos tener en cuenta las notas ligadas, que no tienen un cambio especialmente abrupto en RMS, siendo el cambio de tono el protagonista en este caso.

Por lo tanto, evaluamos los métodos '*flux*' y '*complex*' [31], relacionados con las necesidades descritas previamente:

- El flujo espectral se calcula mediante la distancia euclídea entre dos espectros normalizados consecutivos. Es decir, cómo de rápido cambia el *power spectrum* sin tener en cuenta la fase. Está restringido a los cambios positivos y sumados a lo largo de todas las muestras de frecuencia. La función de detección de onsets es la siguiente:

$$SF(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} H(|X(n,k)| - |X(n-1,k)|) \qquad (9)$$

  Donde $H(x) = \frac{x+|x|}{2}$ es la *half-wave rectifier function*

- En el caso de '*complex*', la amplitud y la fase se pueden considerar conjuntamente para determinar cambios en comportamientos estacionarios. Calculando un valor de fase y magnitud esperado en la muestra actual $X(n,k)$, basándose en las dos anteriores $X(n-1,k)$, $X(n-2,k)$. El *target value* se calcula asumiendo amplitud y tasa de cambio de fase constante:

$$X_T(n,k) = |X(n-1,k)|e^{\psi(n-1,k)+\psi'(n-1,k)} \qquad (10)$$

  La función de detección de ataques basado en este método se define como la suma de las desviaciones absolutas respecto a los valores calculados como estimación, es decir *target values*.

$$CD(n) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |X\,(n,k) - X_T\,(n,k)| \qquad (11)$$

En ambos casos, la selección de *onsets* viene se basa en un algoritmo de selección de pico, que encuentra los máximos locales de las funciones de detección de cada caso, sujeto a varias restricciones.

Tanto el *threshold,* como estas restricciones tienen gran impacto en los resultados, por lo tanto, encontrar valores óptimos depende de la aplicación. La selección de pico sigue los siguientes pasos:

Cada función de *onset detection* $f(n)$ se normaliza para tener una media 0 y una desviación estándar de 1. El tiempo $t = \frac{nh}{r}$ se marca como onset si sigue las siguientes condiciones:

I. $f(n) \geq f(k)\ para\ toda\ k\ que\ cumpla\ n - w \leq k \leq n + w$

II. $f(n) \geq \frac{\sum_{k=n-mw}^{n+w} f(k)}{mw+w+1} + \delta$

III. $f(n) \geq g_\alpha\,(n-1)$

Donde $w = 3$ es el tamaño de la ventana usada para encontrar los máximos locales, m=3 es un multiplicador de manera que la media se calcula sobre un rango mayor antes del pico. El *onset* tiene que alcanzar el *threshold* (δ) por encima de la media local. Por último, $g_\alpha\,(n)$ es la función de *threshold* con el parámetro α dado por:

$$g_\alpha\,(n) = \max\big(f(n), \alpha g_\alpha\,(n-1) + (1-\alpha) f(n)\big) \qquad (12)$$

Ambos métodos pueden ser útiles, por lo tanto, evaluaremos empíricamente cada uno de ellos en la fase de evaluación.

c) Segmentación: *onsets* y *offsets*

Además del ataque, es necesario determinar dónde termina la nota. De esta manera, segmentaremos cada posible nota y la acotaremos entre un *onset* y un *offset*, obteniendo así, un segmento con las muestras del audio original por cada nota.

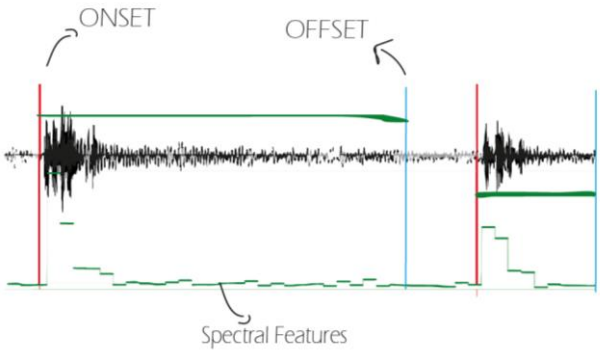


Fig. 3.22 Representación gráfica de la segmentación

Más tarde, en la fase de estimación del *pitch*, tendremos que analizar la relación de cada uno de estos segmentos con sus respectivas muestras de *pitch*, obtenidas en la fase de extracción de melodía.

Para el primer método asumimos un tiempo de ataque mínimo de 0.16 segundos. Consideramos que, si el segmento es menor a este tiempo, el offset será simplemente, la última muestra del segmento analizado. En este caso, estamos considerando que no vale la pena analizar más profundamente el segmento, ya que la nota se acaba cuando empieza otra, pudiendo silenciar la cuerda con la mano, o bien, al tocar en un traste diferente de la misma cuerda.

Para el segundo método, es decir cuando el segmento es mayor al tiempo de ataque mínimo establecido, definimos dos condiciones:

I. Buscamos valores consecutivos de ceros, equivalente a un offset definido. Si esto ocurre, marcaremos como offset la primera muestra nula.

II. En caso contrario, lo asignaremos definiendo un *threshold* correspondiente al 10% de la energía máxima del segmento. Creamos sub-segmentos de 256 muestras y analizamos la energía de cada uno de ellos, buscamos el máximo y calculamos el *threshold*. El sub-segmento que tenga la energía por debajo del 10% del máximo, es marcado como offset. Para calcularla, encontraremos el valor RMS (*root mean square*), mediante Essentia, de cada uno de ellos. Dado un sub-segmento con n valores:

$$x_{rms} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \cdots + x_n^2)} \qquad (13)$$

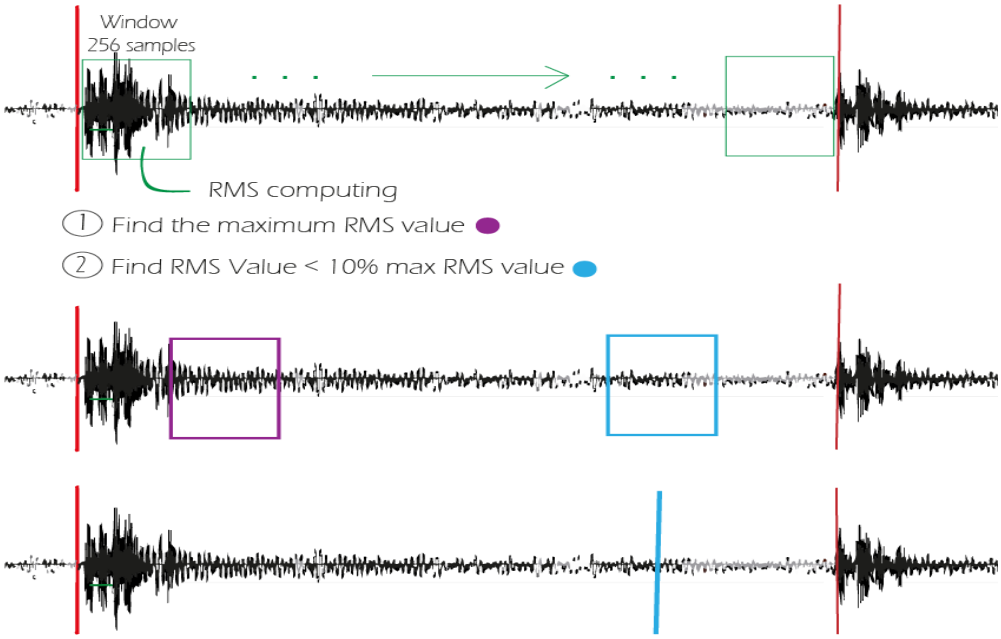


Fig. 3.23 Representación gráfica de la segmentación: offsets

Una vez tenemos los *onsets* y *offsets* determinados, los re-escalamos dividiendo entre 128 (*hop size* usado en la extracción de melodía) para poder evaluar cada segmento respecto a f0.

d) Estimación de *pitch*

En esta sección, el objetivo principal es estimar, para a cada segmento, un valor de *pitch*. Al igual que la asignación del tiempo donde se encuentra, su duración y su valor de energía. Posteriormente, en la fase de post-procesado, todos estos valores se guardarán en un archivo .csv. Paralelamente, crearemos un archivo MIDI con los valores mencionados previamente.

De la misma manera que en la sección de extracción de melodía, consideramos una transcripción monofónica si no se espera acompañamiento, y polifónica si lo hay. Por lo tanto, en el primer caso, obtenemos como máximo una nota por cada valor de tiempo. En cambio, en el caso polifónico podemos obtener como máximo dos notas, siempre y cuando la segunda esté presente en todo el segmente, al igual que la primera.

La estrategia usada para asignar el valor de tono óptimo a cada nota sigue los siguientes pasos:

i. Analizamos cada segmento por separado y ordenamos las frecuencias de F0 que pertenecen a cada uno de ellos, de menor a mayor. En el caso polifónico, analizamos si en todo el segmento hay un segundo valor de F0 en cada una de las muestras. Una vez ordenados, convertiremos todas las frecuencias a cents, asignando como frecuencia de referencia $\mathrm{fT} = 440$ Hz correspondiente a A_4 .

   El cent es la unidad mínima para definir intervalos en notas musicales, equivalente a una centésima de semitono temperado. Debido a que, en acústica musical, todos los intervalos se entienden de forma logarítmica, la conversión nos ayudará a dividir cada una de las octavas en 12 semitonos de 100 cents cada uno. La utilidad reside en expresar intervalos muy pequeños, o bien, comparar intervalos en diferentes sistemas de afinación.

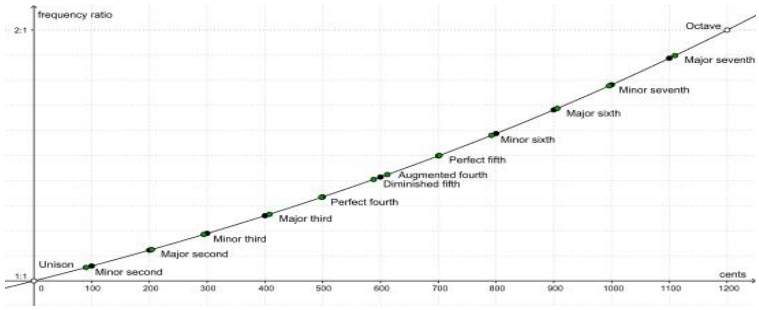


Fig. 3.24 Escala Cents[11]

Dadas dos notas a y b, el valor en cents correspondiente a su intervalo n se calcula de la siguiente forma:

$$n = 1200 \cdot log_2\left(\frac{b}{a}\right) \qquad (14)$$

ii. Con tal de tener un valor que pueda representar la intensidad o volumen de cada nota, computamos la energía de cada uno de los segmentos usando la librería *Essentia*. El rango de volumen MIDI estándar es 0-127, pero después de varias pruebas, observamos que el resultado no es perceptualmente bueno. Por lo tanto, en nuestro caso lo limitaremos a 40-100 para evitar saltos tan abruptos en el volumen relativo de cada nota.

[11]https://upload.wikimedia.org/wikipedia/commons/3/3f/Music_intervals_frequency_ratio_equal_tempered_pythagorean_comparison.svg

La inclusión de este valor ayudará a tener una sensación más "realista" al escuchar y analizar, perceptualmente, el resultado.

$$E_s = \int_{n=-\infty}^{\infty} |x(n)|^2 \, dt \qquad (15)$$

iii. Para crear el archivo MIDI, además de los valores mencionados anteriormente, se precisa indicar el tempo. Servirá para tener un tempo por defecto, aunque después se puede modificar en cualquier secuenciador. En nuestro caso, en vez de asignar un tiempo por defecto para todos, usaremos un detector de *bpm (beats per minute)* para cada audio: *PerceivalBpmEstimator* (Essentia). El algoritmo desarrollado por [32] contiene tres fases:

- Generar una señal de intensidad de inicio: *Onset Strength Signal* (OSS), que contiene una señal de audio en dominio temporal convertida en otra que indique donde los humanos pueden percibir un *onset*.
- Detección de periodo de "*beat*": La señal OSS se divide en ventanas superpuestas de aproximadamente 5.9 segundos y se analizan. Obteniendo un valor de tiempo para cada ventana expresado como un retraso de muestras.
- Las estimaciones de tiempo se acumulan, y mediante estas, se obtiene una estimación global.

iv. Para asignar el valor de *pitch* final a cada, una vez ordenados los valores de cada segmento y convertidos a escala cent, evaluaremos dos métodos:

- Método mediana: representaremos el valor de la variable de posición central del conjunto ordenado de frecuencias.
- Método histograma $H[f_{cent}]$: Dado el conjunto de valores de frecuencia, representaremos el histograma local de cada sub-segmento. De esta forma, asignaremos el valor de *pitch* que aparece más veces como valor final.

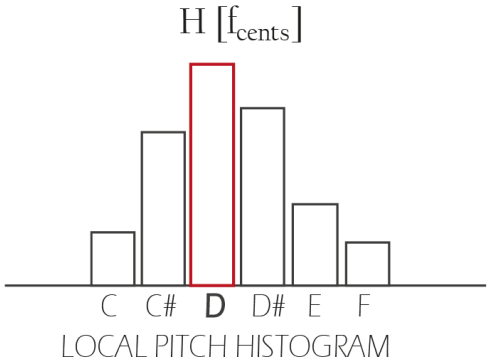


Fig. 3.25 Representación gráfica del histograma local de pitch

Para que el proceso de evaluación y comparación sea más sencillo, el método se podrá escoger como parámetro de entrada del algoritmo.

v. Seguidamente, evaluamos si el valor obtenido está dentro del rango de frecuencias correspondiente a la guitarra. Si se cumple la condición anterior procedemos al cálculo de los demás valores:

- Deshacemos la operación que convierte la frecuencia a escala cent y calculamos el valor de *pitch* escogido como número de nota MIDI mediante:

$$MIDI_{note} = round(12 * \log_2\left(\frac{pitch}{fT}\right) + MIDI_{ref}) \qquad (16)$$

Donde *pitch* es la frecuencia calculada según el método escogido [Hz], fT es la frecuencia de referencia fT = 440 Hz [Hz] y $MIDI_{ref} = 69$ es la frecuencia de referencia, expresada en formato de número de nota MIDI. Para el caso polifónico, antes de escoger como valor de *pitch* final, comprobaremos que la frecuencia está dentro del rango de la guitarra: 80-750Hz.

- Asignamos a cada nota, el tiempo, la duración y la energía de cada una:

$$sTime = onset * \frac{HopSize}{fs} \qquad (17)$$

$$duration = (offset - onset) * \frac{HopSize}{fs} \qquad (18)$$

Donde *hop size* = 128 y la frecuencia de muestreo es 44.1KHz

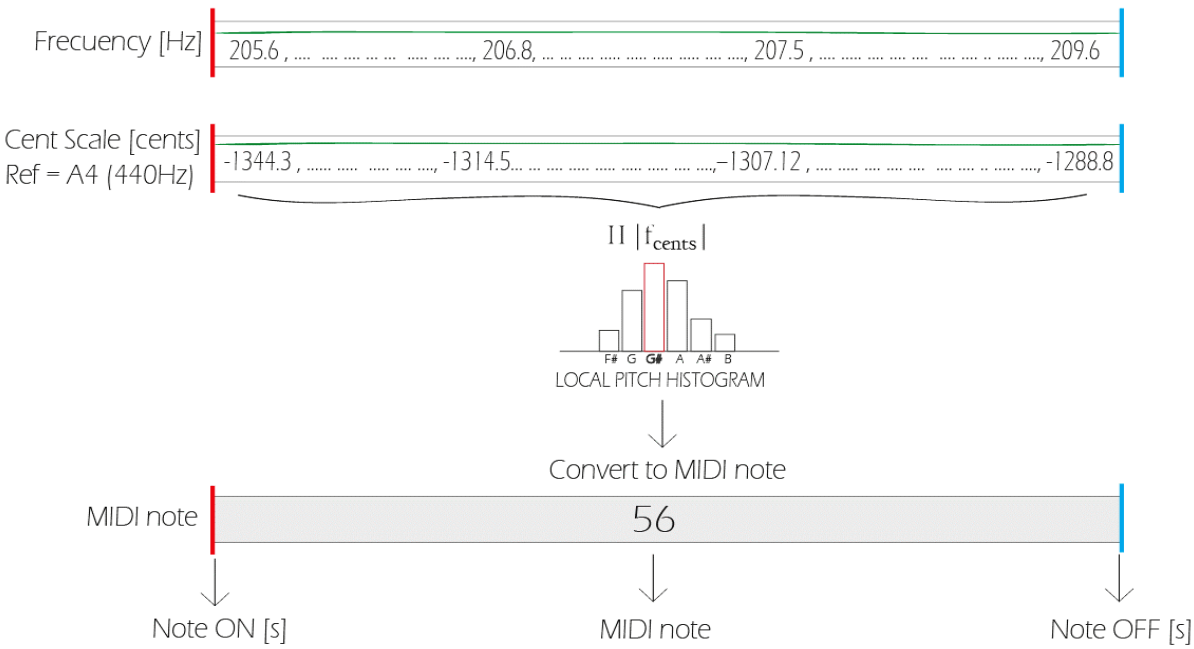


Fig. 3.26 Representación gráfica de la fase de etiquetado de pitch.

Una vez tenemos el valor de *pitch*, junto con el tiempo, duración y valor de energía de cada nota, añadiremos estos datos a una matriz que será post-procesada en el siguiente paso. Paralelamente, añadimos estos valores a la pista MIDI creada anteriormente. Para todo el proceso de creación y exportación de este archivo, se ha usado la librería *MIDIUtil* [33].

3.3 Post-procesado de la transcripción

En esta última fase, utilizaremos algunas técnicas con tal de ajustar o eliminar posibles errores. Básicamente, tendremos en cuenta dos restricciones: la primera se corresponde con la duración mínima de cada nota. La segunda, y más compleja, relacionada con un ajuste teniendo en cuenta las escalas y tonalidades típicas de ciertos palos en el flamenco.

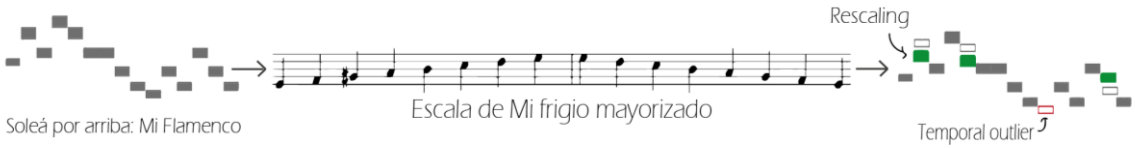


Fig. 3.27 Representación gráfica de la fase de post-procesado.

Para realizar el estudio empírico de la duración de las notas, se han analizado todas las notas pertenecientes al conjunto de falsetas transcritas manualmente para la evaluación. Este paso, además de ayudarnos a eliminar *outliers* con duración menor a la mínima, será útil para ajustar los parámetros utilizados para detectar los *offsets* en la fase de segmentación.

Para ello, además de encontrar la mínima duración, calcularemos la duración media de los segmentos. A continuación, se muestran los resultados, además de un histograma de duración:

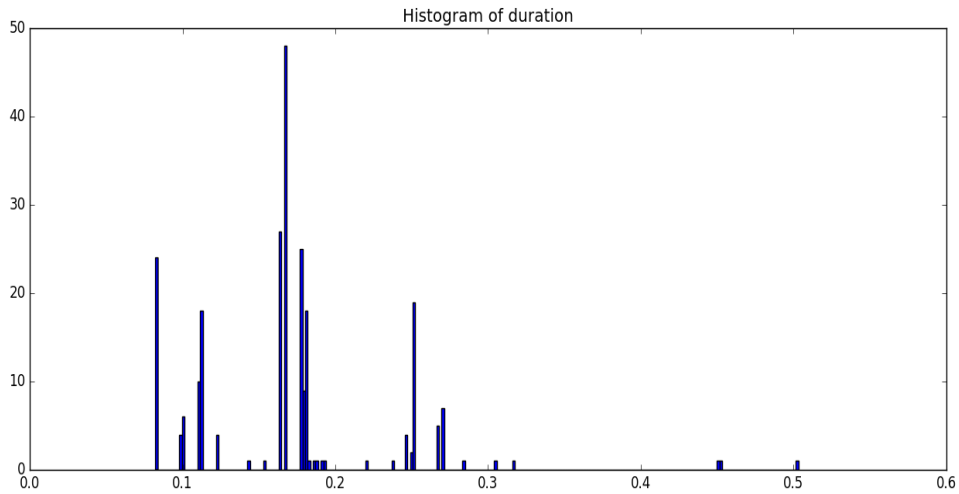


Fig. 3.28 Histograma de duración de las notas de la colección para la transcripción.

- *La duración mínima es 0.08 segundos.*
- *La duración media es de 0.16 segundos*

Una vez eliminados *outliers* temporales, procedemos a la segunda parte del post-procesado: barrido y ajuste de *pitch* teniendo en cuenta aspectos de la teoría musical del flamenco. Para entender esta fase, hay que contextualizar algunos aspectos básicos de melodía y armonía aplicada en el flamenco.

Todos estos conceptos los podemos encontrar el libro de Teoría Musical del Flamenco de Lola Fernández [37]:

1. Sistemas musicales propios de la música flamenca: Antes de concretar que sistemas rigen el lenguaje del Flamenco y así poder detectar errores relacionados con las notas, debemos introducir el sistema modal (Modalidad) y el tonal (Tonalidad): En el sistema modal, predecesor del tonal, encontramos un discurso musical horizontal basado en notas consecutivas.

   De este sistema, surgen siete modos distintos, obtenidos a partir de las siete notas naturales de la escala diatónica. De esta manera, la colocación de los tonos (cinco) y semitonos (dos) varía en función de cada modo, dando como resultado siete escalas en las que no hay alteraciones en ninguno de sus grados.

   Cada uno de los modos tiene un nombre procedente de la nomenclatura en la Edad Media:

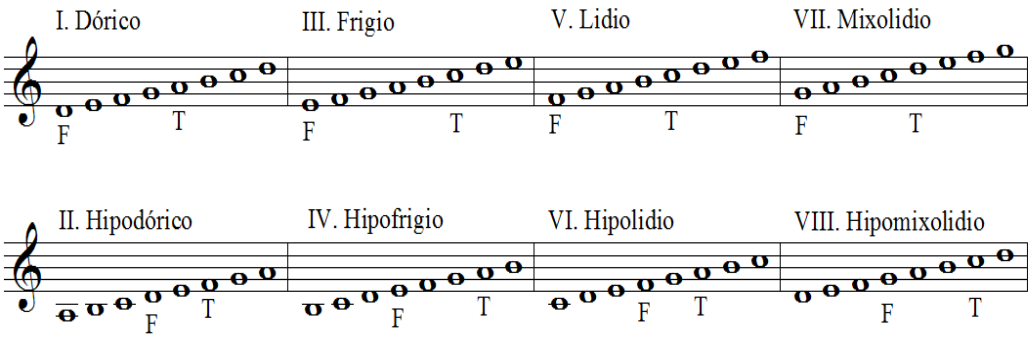


Fig. 3.29 Los ocho modos del sistema modal.

   Más tarde, y con la evolución hacia la polifonía y la verticalidad, aparece el sistema tonal, a su vez, dando lugar a la Armonía. Se establece a partir de dos modos antiguos: jónico y eólico, convirtiéndose en el modo mayor y menor del sistema tonal, y sobre los cuales se superponen terceras sobre cada una de sus notas, dando lugar a los grados de cada tonalidad.

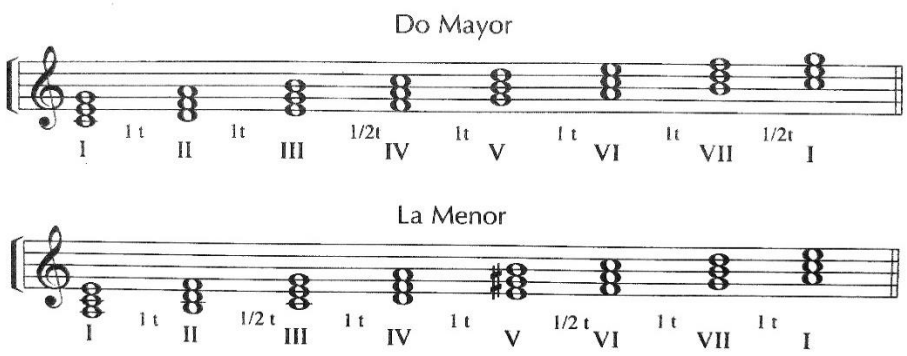


Fig. 3.30 Todos los grados de la tonalidad de Do Mayor y La Menor

Una vez contextualizados ambos modos, especificaremos como conviven y se usan en la música flamenca:

Sistema modal	Modo frigio Modo frigio *mayorizado* Modo jónico Modos mixtos (dos modos en el mismo cante) Modo flamenco
Sistema tonal	Modo mayor Modo menor
Sistema modal + sistema tonal	Modo flamenco + modo mayor (bimodal) Modo flamenco + modo menor

Tabla. 3.1 Los sistemas musicales que se dan en el flamenco [37]

En nuestro caso, hablaremos solo de la parte melódica, ya que solo nos interesa ajustar las notas para que respeten las alteraciones propias de cada escala. A continuación, se concretan algunos modos típicos y sus escalas, para según qué tipo de cante o palo.

Podremos observar que pueden convivir los dos sistemas, o modos (modos mixtos) en un mismo cante, incluso dar pie a otros modos, que surgen de la adaptación de armonizar el modo frigio del sistema modal: modo flamenco.

Uno de los modos más explotados en el flamenco es el modo frigio, utilizado en el toque por arriba (Mi), aunque también existe la variante de modo frigio *mayorizado*. Este último tiene su tercer grado elevado medio tono (G#), y en muchas ocasiones, se combinan ambas escalas.

Mayoritariamente, la tercera elevada aparece en los movimientos melódicos ascendentes, y la tercera propia del modo frigio para los descendentes, incluso produciéndose choques entre las dos variantes entre melodía y acompañamiento.

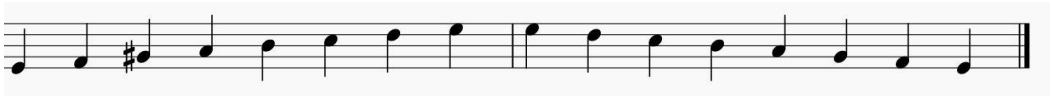


Fig. 3.31 Escala de Mi frigio mayorizado, ascendente y descendente

Esta forma de entender la melodía, y sumando, la armonización del modo frigio, constituyen el modo flamenco.

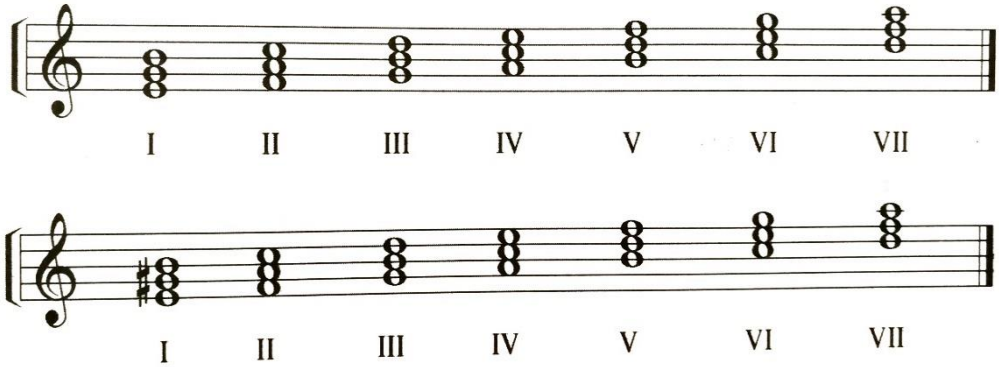


Fig. 3.32 Armonización del modo de Mi frigio y de Mi Flamenco

Otra de las tonalidades más comunes es la de La Flamenco, que surge del modo frigio pero transportado a La, utilizada en el toque por medio. También podemos encontrar, al igual que en la escala *mayorizada*, el tercer grado aumentado medio tono(C#):

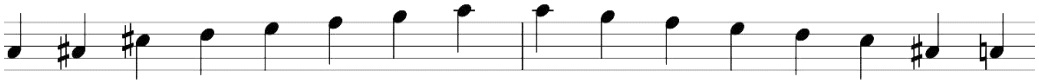


Fig. 3.33 Escala de La en forma frigia mayorizada, ascendente y descendente

En algunas ocasiones podemos encontrar casos de *bimodalidad*, combinando el modo flamenco con modos mayores, sobre todo en los Fandangos o Cantes del Levante. En el anexo 2 podremos encontrar una tabla de tonos y modos propios de cada uno de los cantes.

2. Para implementar estos recursos en el algoritmo, definiremos vectores con la escala apropiada y evaluaremos en casos concretos como es el resultado respecto al caso dónde no hay ajuste de escala. Usaremos siempre un entorno controlado, es decir, falsetas en las que sepamos cual es el tono. En su lugar, si sabemos el palo, podremos definir una tonalidad por defecto, pero en este caso, debemos saber si está transportada o no.

   Como esta información, para el usuario, puede ser difícil de concretar, se omitirá este paso siempre y cuando no se tenga toda la información necesaria. Por otra parte, se podría detectar la tonalidad extrayéndola automáticamente, pero puede

resultar complicado en caso de falseta muy corta. Por lo tanto, esta fase del post-procesado, será usada únicamente para evaluar en entornos controlados.

Se han escogido tres casos a evaluar: escalas frigias *mayorizadas* en Mi y en La, y la escala de Mi Mayor. Estas formas, se usan muy frecuentemente en las piezas de flamenco, pero también es muy común encontrarlas con cierta trasposición, con tal de adaptarse a la tesitura vocal.

Por lo tanto, fijaremos un parámetro de entrada para definir el factor de trasposición, si lo hay, solo cuando el usuario sepa si la pieza a transcribir está traspuesta y cuantos semitonos lo está. Este factor afectará a las escalas de referencia y por defecto será 0. Los vectores, en forma de notas MIDI son los siguientes:

i. *Mi frigio mayorizado: [40, 41, 43, 44, 45, 47, 48, 50]*
ii. *Mi mayor: [40, 41, 42, 44, 45, 47, 48, 49, 51]*
iii. *La frigio mayorizado: [45, 46, 48, 49, 50, 52,53, 55]*

4. RESULTADOS

En este apartado utilizaremos la estrategia de evaluación desarrollada en el capítulo 2. En primer lugar, analizaremos los resultados de las métricas en la fase de extracción de falsetas. En segundo lugar, para la fase de transcripción, evaluaremos varios métodos y parámetros variables que ayudarán a analizar la eficacia y precisión de algunos métodos cuestionados anteriormente.

4.1 Resultados de la fase de extracción de falsetas

Como hemos visto en capítulos anteriores, lo que evaluamos en este caso es:

- Cantidad de falsetas detectadas con el tiempo mínimo fijado a 15 segundos.

Fig. 4.1 Gráfico de falsetas encontradas para cada pieza de la colección

El resultado coincide con numero de falsetas detectadas manualmente, por lo tanto, concluimos que se han encontrado el 100% de falsetas de un mínimo de 15 segundos.

- Precisión en el tiempo de delimitación de éstas. Mediante la colección anotada manualmente, trataremos estas falsetas como segmentos, y evaluaremos la precisión en su delimitación. Resulta interesante transcribir partes como remates o llamadas, puesto que también forman parte de la guitarra. Por este motivo consideraremos correcta la segmentación si está dentro de una ventana de 4 segundos:

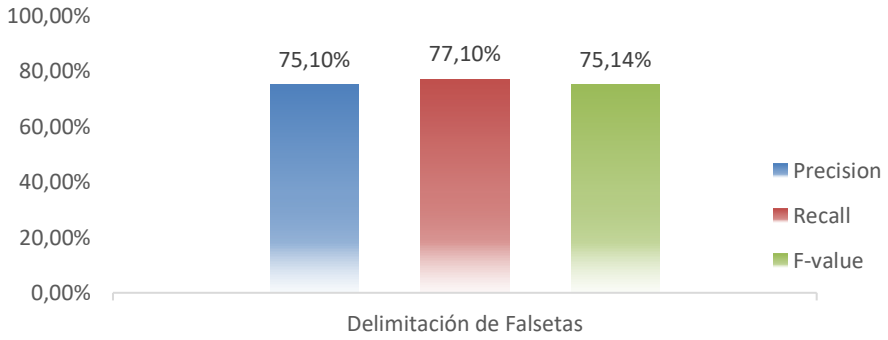


Fig. 4.2 Resultados de la delimitación de falsetas

4.2 Resultados de la fase de transcripción

En primer lugar, evaluaremos los dos métodos de detección de ataque cuestionados en el capítulo 3, con el objetivo de decidir cuál de los dos es más preciso para el caso de la guitarra flamenca. A continuación, se muestran las métricas evaluadas para cada uno de ellos:

1. Método de detección de ataque 'flux', basado en características del flujo espectral. Método de estimación de *pitch* mediante histograma:

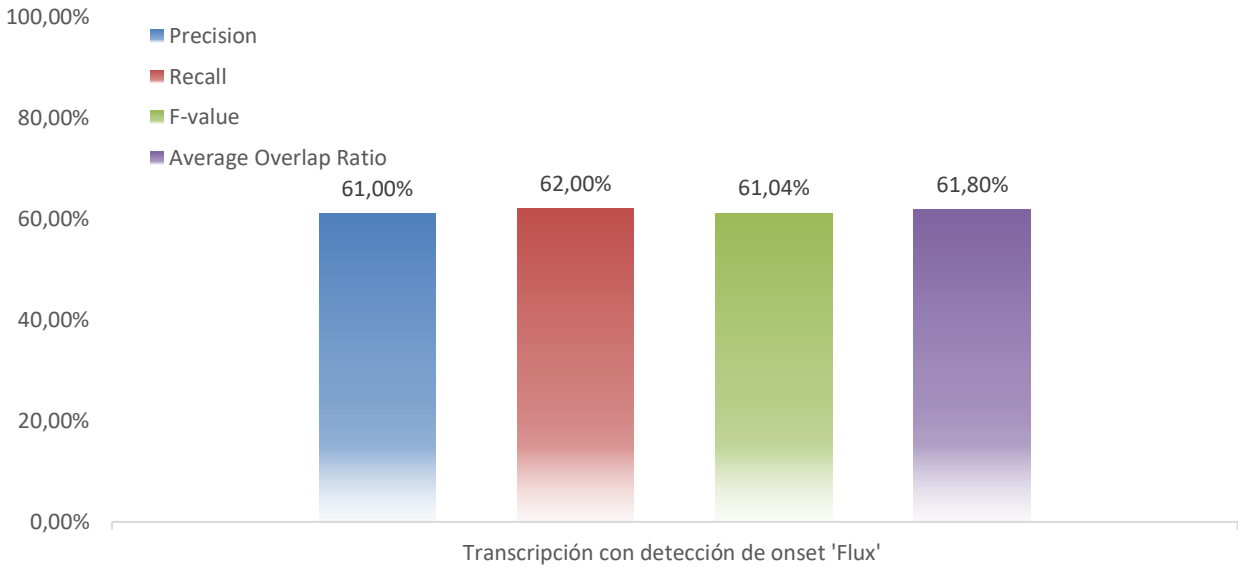


Fig. 4.2 Resultados de la transcripción usando 'flux'

2. Método de detección de ataque 'complex', basado en las diferencias espectrales en el dominio complejo. Método de estimación de *pitch* mediante histograma:

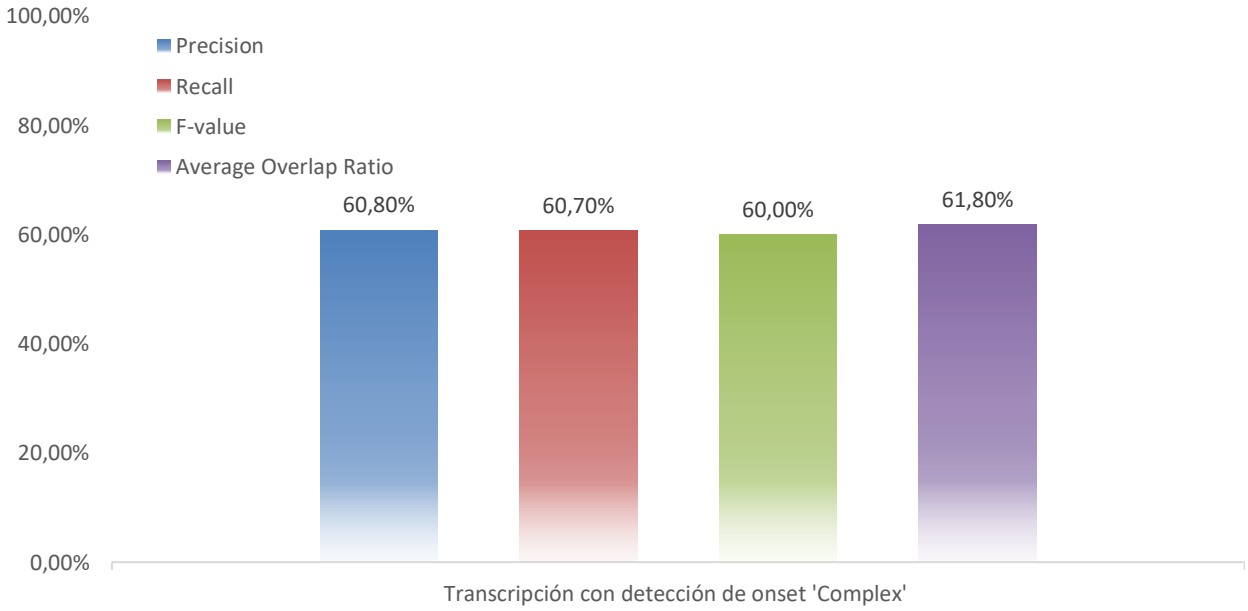


Fig. 4.3 Resultados de la transcripción usando 'complex'

Podemos concluir que el mejor resultado lo tenemos si usamos el método *'flux'*, aunque la diferencia es mínima respecto a *'complex'*. Si evaluamos perceptualmente ambos métodos, observamos que obtenemos mejor resultado para cada uno en ciertas técnicas, lo que después se compensa en los resultados. Por este motivo, una posible mejora seria detectar los ataques de manera con un método adaptativo según se precise según la técnica. Este punto se desarrollará en las conclusiones.

Una vez analizados los métodos de detección de ataque, evaluaremos los dos métodos de selección de *pitch* dentro de cada nota, dado el vector f0, con todas las frecuencias estimadas cada 128 muestras:

- Método mediana (1), escogiendo el valor de la variable de posición central del conjunto ordenado de frecuencias:

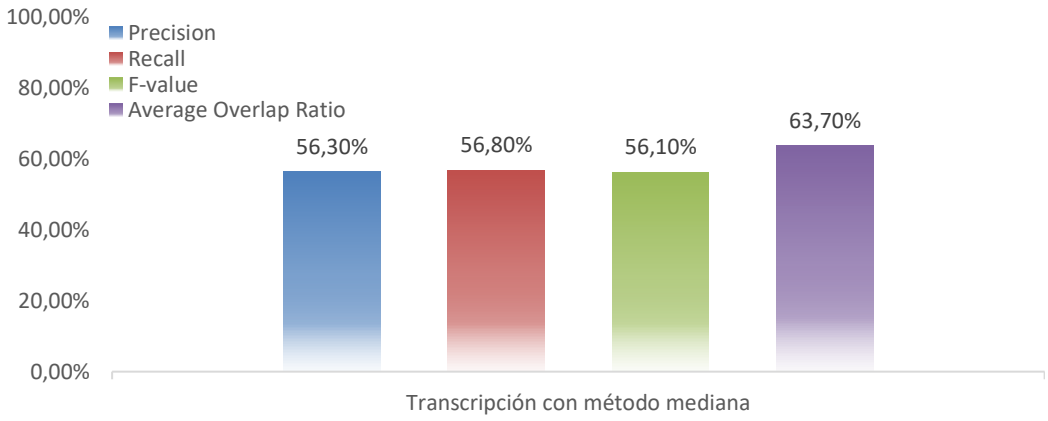


Fig. 4.4 Resultados de la transcripción usando método mediana

- Método histograma (2), que corresponde a la figura 4.2 asignando el valor de *pitch* que aparece más veces como valor final. En el siguiente gráfico observamos la comparativa entre un método y otro:

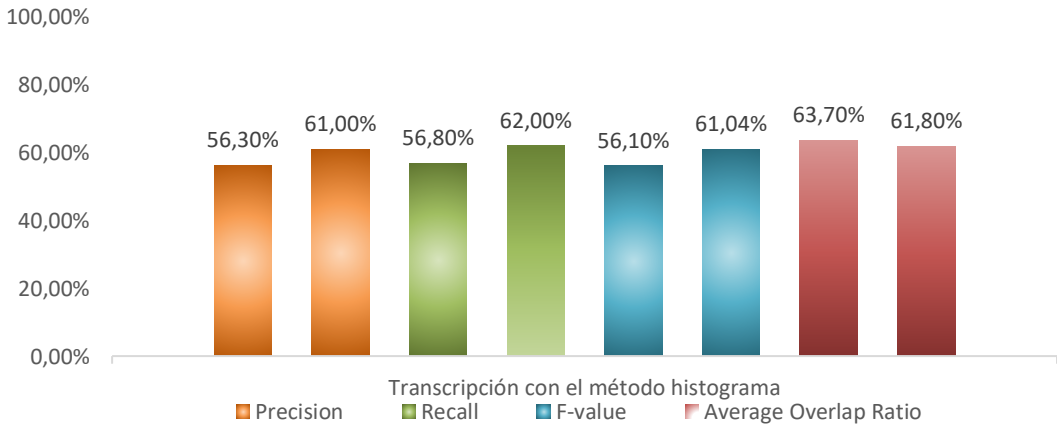


Fig. 4.5 Resultados de la transcripción usando método mediana vs. histograma

Podemos observar, después de analizar los resultados, que el método más preciso es el método que utiliza el cálculo del histograma para definir el *pitch* de cada nota.
Una vez analizados los casos generales, evaluaremos el re-escalado de notas para dos casos concretos. Con tal de observar mejoras, utilizaremos dos falsetas con la tonalidad y la transposición controlada:

- Falseta de Soleá por arriba (Mi) en tonalidad de Mi flamenco, por lo tanto, la escala que le corresponde es la de Mi frigio *mayorizado*. La duración de esta falseta es de 6 segundos. La primera columna de cada métrica corresponde a la evaluación sin re-escalado:

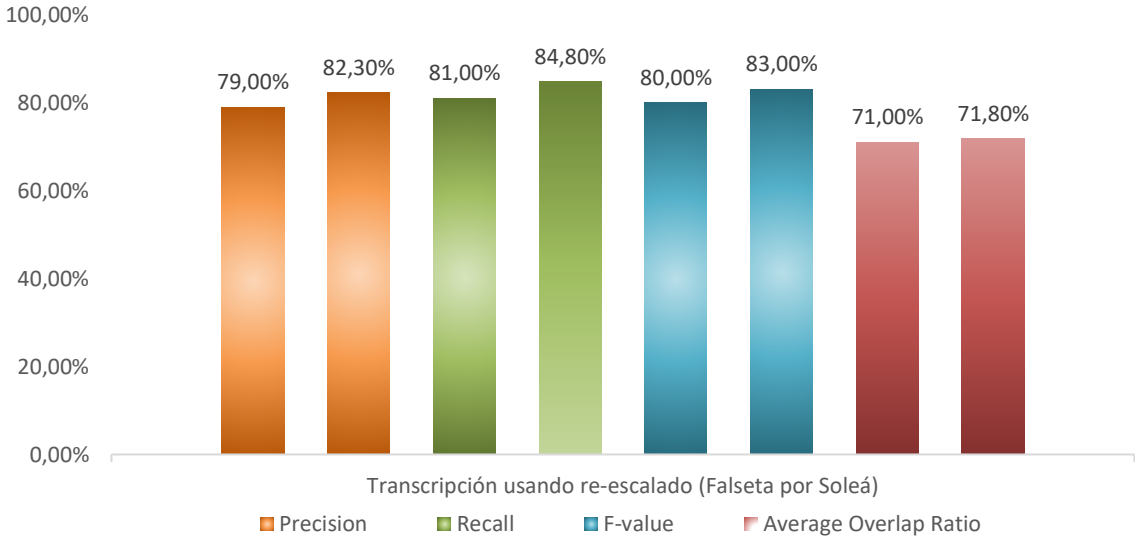


Fig. 4.6 Resultados de la transcripción usando re-escalado (Falseta por Soleá)

- Fragmento de 26 segundos de la pieza por Alegrías "Punta y Tacón" interpretada por Amir-John Haddad. Tonalidad de Mi Mayor sin transposición:

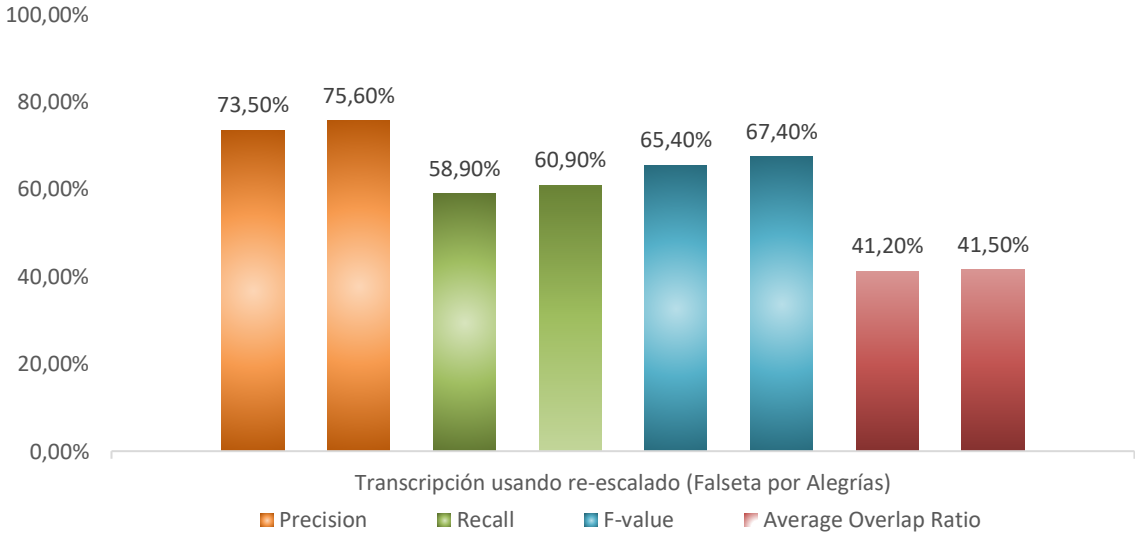


Fig. 4.7 Resultados de la transcripción usando re-escalado (Falseta por Alegrías)

Vistos los resultados, se observa una mejora respecto al caso sin re-escalado, aunque bastante pequeño. Se debería evaluar para cada caso concreto, pero dado que no disponemos de una gran colección para la transcripción, esta fase la usaremos solo en entornos controlados, y como posible mejora para proyectos futuros que den solución a los problemas encontrados en éste. Se desarrollarán estos hechos en el capítulo de las conclusiones.

5. REPRODUCIBILIDAD

Con tal de proporcionar reproducibilidad al algoritmo, se proponen dos vías de experimentación:

1. Disponibilidad del código en un repositorio GitHub: PyToque[12]

2. Disponibilidad de la colección para la transcripción en la plataforma Zenodo[13] [38]: 15 archivos de audio que contienen las falsetas, además de la anotación manual de la transcripción, en formato MIDI, para cada una de ellas.

3. Para usuarios no tan familiarizados con los entornos de programación, se propone un prototipo de interfaz para una herramienta útil para cualquier usuario. Contiene todas las funcionalidades que se proponen cómo parámetros variables de entrada del algoritmo:

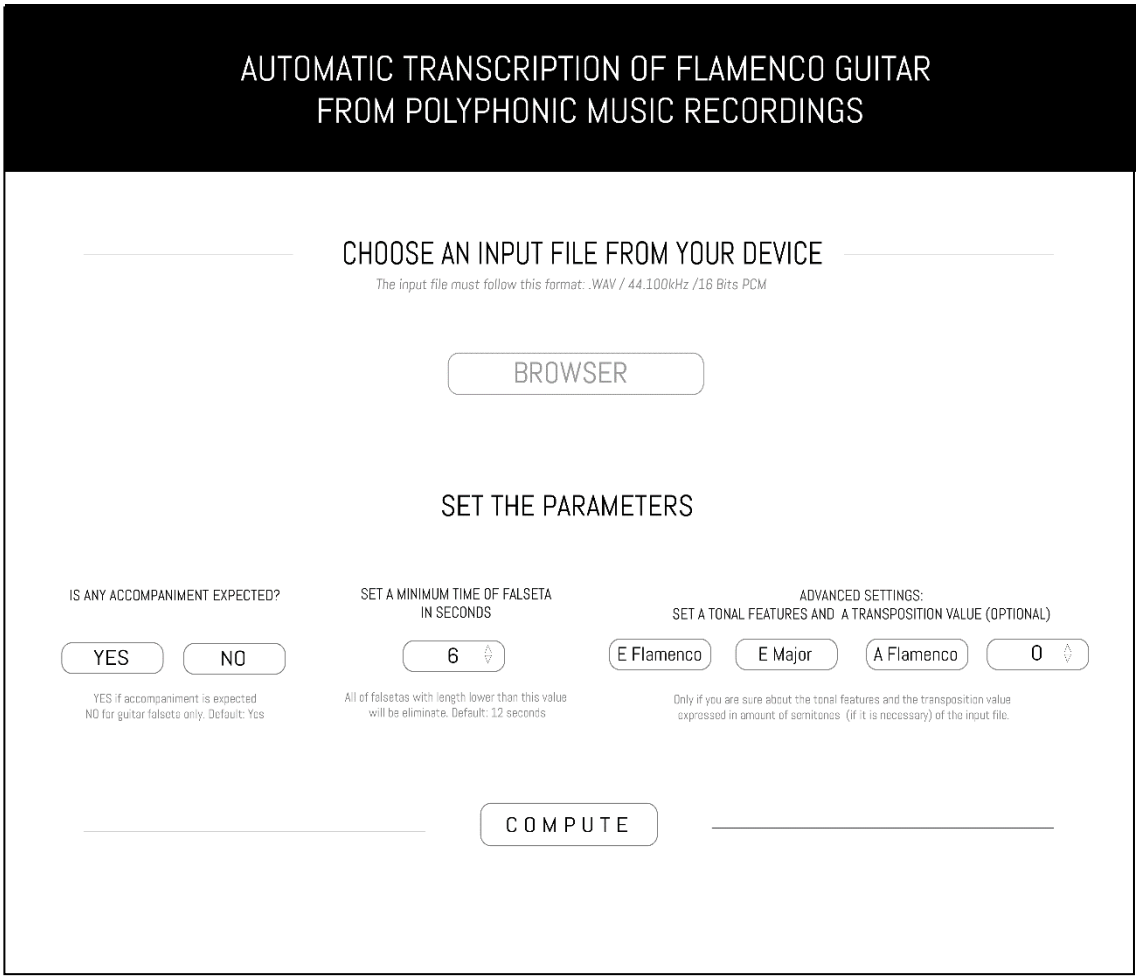


Fig. 5.1 Prototipo de interfaz para la herramienta

[12] https://github.com/SoniaLuque/PyToque
[13] http://doi.org/10.5281/zenodo.804050

Como podemos observar en la figura anterior, el prototipo contiene tres fases: escoger el archivo de entrada, ajuste de parámetros vistos en los capítulos anteriores y computación del algoritmo.

Una vez computado, el resultado se muestra en una segunda interfaz, cuya funcionalidad es poder descargar los archivos MIDI y *.csv* resultantes. Además, incluye un apartado de visualización, dónde encontramos un reproductor con la forma de onda y el archivo MIDI en forma de *piano roll.*

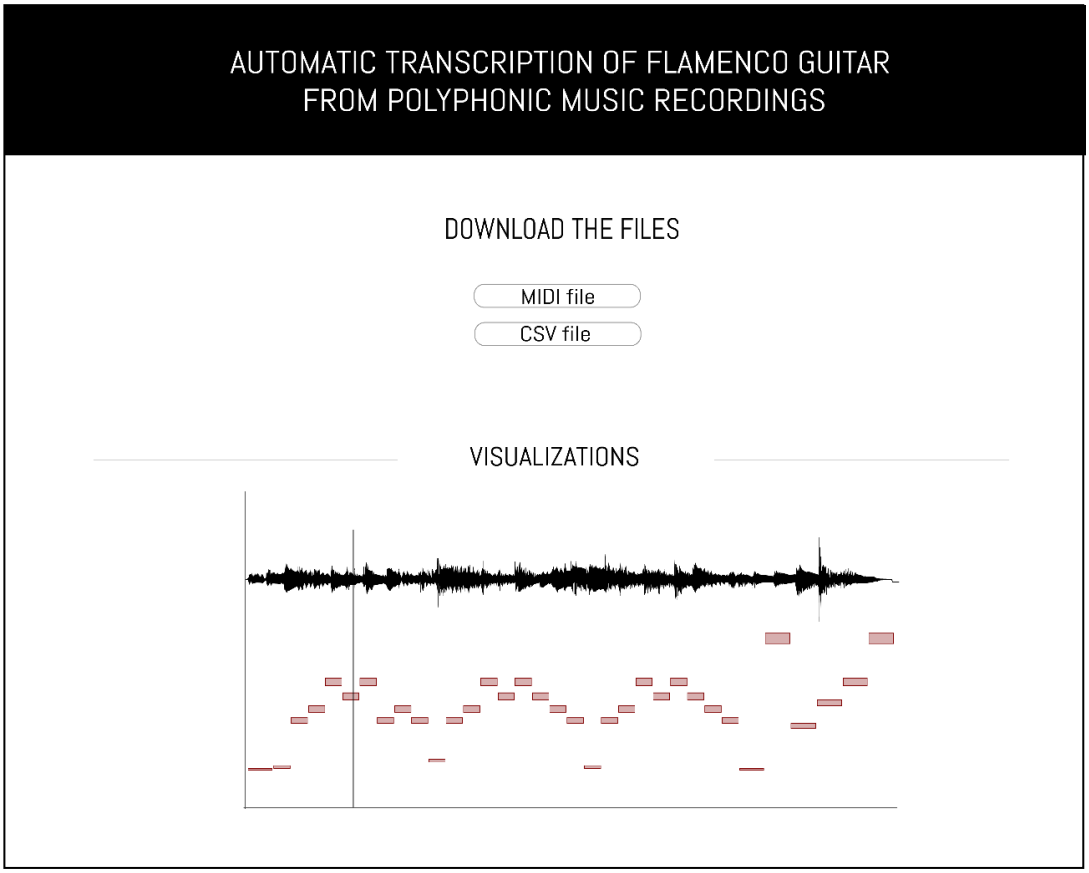


Fig. 5.2 Prototipo de interfaz (2) para la herramienta

6. CONCLUSIONES

Una vez terminado el proceso de desarrollo y evaluación podemos determinar las conclusiones que se extraen del proyecto en general, y particularmente, para ciertas fases que pueden ser replanteadas para más precisión en proyectos futuros. Separaremos las conclusiones en tres fases: conclusiones generales, extracción y transcripción.

1. Conclusiones generales: como hechos que han afectado a todo el proyecto encontramos, en primer lugar, la dificultad para encontrar una colección de anotaciones manuales con sus falsetas correspondientes en los formatos que se requerían. A falta de encontrarlas, se han tenido que crear a partir de partituras y posteriores alineaciones manuales para cada falseta.

   Es un proceso bastante costoso en cuestión temporal, por lo tanto, la consecuencia ha sido que la colección para la transcripción es muy reducida. Esto comporta una evaluación menos fiable encontrando casos dispares. Por otra parte, la colección para la extracción, al tener que delimitar falsetas de manera subjetiva, se pueden encontrar disconformidades entre lo que se considera o no parte potencialmente interesante para la transcripción.

   En términos generales, la colección de falsetas para la transcripción es más sencilla que la que aparece en las canciones comerciales usadas en la extracción: motivo por el cual se trabaja diferente para cada conjunto.

   El objetivo para trabajos futuros, es crear una colección general, que tenga la delimitación de falsetas y además la transcripción de ellas, aunque resultan mucho más complejas para la anotación manual. Además, se podrá trabajar para el caso polifónico, sin tener que comprobar la eficacia de los dos casos, ya realizada en este proyecto.

   Por otra parte, separar las dos colecciones ha servido para saber cómo se trabaja en ambos casos, y mantenerlos aislados, aunque tuviéramos falsetas polifónicas a transcribir, el resultado sería monofónico (para la colección de transcripción).

2. Conclusiones para la extracción: para este caso, obtenemos unos resultados relativamente buenos para la evaluación, pero destacamos dos conclusiones:

   a. La delimitación, se realiza respetando el proceso de dialogo, por lo tanto, observamos que hay una cierta ambigüedad en la segmentación, teniendo en cuenta tanto cierres, como remates, llamadas o introducciones.

      Aunque estos recursos también son interesantes para transcribir, si queremos delimitar únicamente las falsetas, podría incluirse un delimitador más preciso, mediante análisis con descriptores de

características espectrales. Mayoritariamente, los otros recursos incluyen técnicas más cercas a los rasgueos, y aunque no siempre, pero las falsetas suelen incluir contornos más melódicos, entonces, los descriptores ayudarían a discriminar las otras partes si fuera necesario. A modo de conclusión personal, parece interesante que se incluya todo, puesto que también hay que tocarlo.

b. Por otra parte, y relacionado también con el proceso de diálogo, a la vez que resulta muy útil, puede dar problemas si el instrumento que se toca no es la guitarra (aunque la transcripción se haría igualmente, probablemente, con peor resultado). Entonces, este método, como ya avanzamos en el capítulo de estrategia de evaluación, queda reducido a la forma clásica de dialogo entre cante y guitarra. Aunque se podría proponer el análisis de características tímbricas entre la guitarra y otros instrumentos que puedan aparecer, si no respetan la forma clásica.

3. Conclusiones para la transcripción: en el inicio del proyecto, asumimos que los ataques serían relativamente fáciles de detectar. Una vez analizada la evaluación, cambiando tolerancias tanto de *pitch* como de tiempo de ataques, se ha observado que la mayoría de errores vienen dados por la detección de ataques.

   En los resultados percibimos tanto en una colección como en otra, que las técnicas con las que el algoritmo trabaja mejor son las del tipo picado o *alzapúa*. En general técnicas que, aunque se toquen a mucha velocidad, el ataque se marca con suficiente intensidad para que sea correctamente detectado. Por otra parte, técnicas dónde aparecen notas ligadas, o el trémolo, resultan más complejas de transcribir con precisión.

   De esta manera, se concluye que, para una mejora de la eficacia del algoritmo, se debería proponer un tipo de segmentación adaptativa, que tenga más en cuenta las características propias de algunas técnicas flamencas.

   Las escalas Flamencas dotan de una gran complejidad melódica y armónica, que dificulta el caso general para cualquier falseta. El re-escalado de *pitch*, da un mejor resultado en cuanto a precisión, aunque se podría mejorar si se detecta el palo, la tonalidad y la transposición automáticamente, para dar más comodidad al usuario. Para falsetas largas, algunos módulos de *Essentia*, permiten detectar características tonales, pero si son cortas encontramos errores.

   El re-escalado para casos concretos, nos permite escuchar un buen resultado, puesto que puede ser una nota incorrecta comparada con la alineación manual, pero al estar dentro de la escala, perceptualmente es bueno.

En términos generales, realizar una labor de investigación relacionada con la guitarra flamenca, supone una gran responsabilidad a la par que una oportunidad de dar pie a otras investigaciones útiles para el aprendizaje, el análisis, incluso la divulgación de este particular ámbito.

El estudio de esta área a nivel computacional, debería ser una herramienta de ayuda para profesionales y aficionados, sin entorpecer la inmensa labor que realizan los expertos guitarristas flamencos. Ayudando así, a la imparable expansión y auge internacional que está adquiriendo la guitarra flamenca, permitiendo que se profundice en conocimiento y se preserve en forma, como este ámbito merece.

Referencias

[1] Varela Iglesias J L, Tadeo Monge FT, Carreño Rujillo G, Doménech Colón F, Abad León A, et al. (2014). Real Academia Española. Diccionario Usual. *Edición Del Tricentenario*. Retrieved from http://dle.rae.es/srv/fetch?id=I2kiw28

[2] "El cante jondo: Primitivo canto andaluz (Conferencias). García Lorca." [Online]. Available: http://federicogarcialorca.net/obras_lorca/el_cante_jondo.htm.

[3] "El flamenco - patrimonio inmaterial - Sector de Cultura - UNESCO." [Online]. Available: http://www.unesco.org/culture/ich/es/RL/el-flamenco-00363.

[4] G. C. Buend, P. Independiente, and E. F. Modernos, "Guillermo Castro Buendía : La guitarra flamenca , amalgama de lenguas sonoras," pp. 1–9, 2013.

[5] "La guitarra perfecta | Paco Chorobo." [Online]. Available:http://chorobo.com/es/la-guitarra-perfecta/.

[6] Isabel Guirado Morales "El diálogo entre cante, baile y toque" Jornadas sobre el Flamenco 2016.

[7] "Glosario flamenco: de «falseta» a «fuga» - ABC de Sevilla." [Online]. Available: http://sevilla.abc.es/cultura/musica/20140925/sevi-glosario-flamenco-falsete-fuga-201409250402.html.

[8] "CVC. Rinconete. Música y escena.El flamenco hoy. La melodía, por Juan Cruz Palacios."Available:http://cvc.cervantes.es/el_rinconete/anteriores/septiembre_13/18092013_01.htm.

[9] M. Schedl, E. Gomez, and J. Urbano, Music Information Retrieval: Recent Developments and Applications, vol. 8, no. 2–3. 2014.

[10] J.-M. Díaz-Báñez (2017). Mathematics and Flamenco: An Unexpected Partnership. The Mathematical Intelligencer, DOI: 10.1007/s00283-016-9688-4.

[11] T. Recio, "La Columna de Matemática Computacional Sobre problemas de matemáticas en el estudio del cante flamenco Introducción," vol. 16, pp. 513–541, 2013.

[12] E. Gómez, P. Herrera, F. Gómez-Martin (2013) Computational Ethnomusicology: perspectives and challenges, Journal of New Music Research, 42:2, 111-112

[13] "COFLA | COMPUTATIONAL ANALYSIS OF FLAMENCO MUSIC." [Online]. Available: http://www.cofla-project.com/.

[14] J. M. Díaz-Báñez, F. J. Escobar, E. Gómez, F. Gómez, and J. Mora, “COFLA II: Análisis COmputacional de la música FLAmenca.”

[15] Salamon, J., Rocha B., Gómez E. (2012) Musical Genre Classification using Melody Features Extracted from Polyphonic Music Signals. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[16] Gómez, F., Pikrakis A., Mora J., Diaz-Bañez J. M., Gómez E., Escobar F. et. al. (2012) Automatic Detection of Melodic Patterns in Flamenco Singing by Analyzing Polyphonic Music Recordings. III Interdisciplinary Conference on Flamenco Research (INFLA) and II International Workshop of Folk Music Analysis (FMA).

[17] N. Kroher, E. Gómez, C. Guastavino, F. Gómez, “Computational models for perceived Melodic 52 Similarity in a Cappella Flamenco Singing”, Universitat Pompeu Fabra 2013-14, pp. 1-6.

[18] Kroher, N. & Gómez, E. (2016). Automatic Transcription of Flamenco Singing from Polyphonic Music Recordings. IEEE Transactions on Audio, Speech and Language Processing. 24(5), 901-913.

[19] “PyCharm.” [Online]. Available: https://www.jetbrains.com/pycharm/features

[20] Bogdanov, D., Wack N., Gómez E., Gulati S., Herrera P., Mayor O., et al. (2013). ESSENTIA: an Audio Analysis Library for Music Information Retrieval. International Society for Music Information Retrieval Conference (ISMIR'13). 493-498.

[21] J. Salamon and E. Gómez, “Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics“, IEEE Transactions on Audio, Speech and Language Processing, 20(6):1759-1770, Aug. 2012.

[22] Chris Cannam, Christian Landone, and Mark Sandler, Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files, in Proceedings of the ACM Multimedia 2010 International Conference.

[23] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel P. W. Ellis. “mir_eval: A Transparent Implementation of Common MIR Metrics”, 15th International Society for Music Information Retrieval Conference, 2014.

[24] “aubio, a library for audio labelling.” [Online]. Available: https://aubio.org/.

[25] E. Gómez, F. Cañadas, J. Salamon, J. Bonada, P. Vera, P. Cabañas (2012). Predominant fundamental frequency estimation vs singing voice separation for the automatic transcription of accompanied flamenco singing. Proc. 13th International Society forMusic Information Retrieval Conference (ISMIR 2012)

[26] “Algorithm reference: OnsetDetection — Essentia 2.1-dev documentation.” [Online].Available:http://essentia.upf.edu/documentation/reference/std_OnsetDetection.html.

[27] Bello, Juan P., Chris Duxbury, Mike Davies, and Mark Sandler, On the use of phase and energy for musical onset detection in the complex domain, Signal Processing Letters, IEEE 11, no. 6 (2004): 553-556.

[28] P. Brossier, J. P. Bello, and M. D. Plumbley, "Fast labelling of notes in music signals," in International Symposium on Music Information Retrieval (ISMIR’04), 2004, pp. 331–336.

[29] D. P. W. Ellis, "Beat Tracking by Dynamic Programming," Journal of New Music Research, vol. 36, no. 1, pp. 51–60, 2007.

[30] J. Laroche, "Efficient Tempo and Beat Tracking in Audio Recordings," JAES, vol. 51, no. 4, pp. 226–233, 2003.

[31] S. Dixon, "Onset detection revisited", in International Conference on Digital Audio Effects (DAFx'06), 2006, vol. 120, pp. 133-137.

[32] Percival, G., & Tzanetakis, G. (2014). Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22(12), 1765–1776.

[33] “MIDIUtil 1.1.3 : Python Package Index.” [Online]. Available: https://pypi.python.org/pypi/MIDIUtil/

[34] “Basic Soleares Falsetas - Flamenco Guitar Transcriptions.” [Online]. Available: http://canteytoque.es/solsamp.htm

[35] “tabsflamenco.com.” [Online]. Available: http://tabsflamenco.com/

[36] A. Klapuri, "Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes ", International Society for Music Information Retrieval Conference (2006)

[37] L. Fernández, Teoría musical del flamenco, Acordes Concert, 2004.

[38] Advani Aguilar, J.I. (2016). Un algoritmo para la detección automática de falsetas de guitarra flamenca. (Trabajo fin de grado inédito). Universidad de Sevilla, Sevilla.

[39] Sonia Rodríguez Luque. (2017). ToqueFlamenco [Dataset]. Zenodo. http://doi.org/10.5281/zenodo.804050

ANEXO I

A continuación, se detallan las anotaciones manueles de la colección de datos utilizada para la fase de extracción:

1. Donde se divisa el mar (cartagenera) 0- 0:28; 1:52-2:23
2. Hermano mío (seguiriya) 0-41; 2:23- 3:03
3. No dudes de la nobleza (fandango) 0-0:37; 2:07-2:27
4. Y no llegaste a quererme (granaína) 0-0:41; 1:14-1:34
5. Maldito yo estaba (seguiriya) 0-0:30; 2:35-3:00
6. Se me partió la barrena (taranto) 0-0:34; 1:28-2:11
7. Pueblos de la tierra mía (alegrías) 0:50-1:18; 2:14-2:30
8. Que no se quita con ná' (fandangos) 0-0:35; 1:34-2:10
9. Calabosito oscuro (seguiriya) 0-0:31; 2:01-2:22
10. De tus ojos soy cautivo (soleá) 0-0:28; 1:18-1:32; 2:18- 2:41
11. Moral (fandangos) 0-0:22; 1:07-1:26
12. En una piedra me acosté (fandangos) 0-0:29; 1:55-2:11
13. Las penas de mi madre (seguiriya) 0-0:18; 1:44-2:08; 2:29-2:49
14. Camina y dime(tarato) 0-1:12; 3:18-3:31
15. Quisiera volverme pulgar (tangos malagueños): 0-0:28; 1:44-2:06
16. A los santos del cielo (seguiriya) 0-0:36; 2:38-3:02
17. Los dos se juegan la vida (taranto) 0-0:31; 1:52-2:21
18. En tu puerta da la luna(taranto) 0-0:24; 0:44-1:02; 2:23-3:04
19. Ni que me manden a mi (fandangos) 0-0:29; 1:57-2:20
20. Sólo vivo pa'quererte (granaína) 0-0:38; 0:53-1:23

ANEXO II

1. CANTES SIN ACOMPAÑAMIENTO

PALO	MODO
Romances	Frigio mayorizado
Tonás	Frigio, Frigio mayorizado
Martinetes	Jónico
Carcelera	Jónico
Debla	Frigio mayorizado
Saeta	Frigio mayorizado, jónico
Nanas	Frigio mayorizado

2. CANTES BÁSICOS O FUNDAMENTALES

PALO	MODO	TONO
Seguiriya	Flamenco	La flamenco
Cabales	Mayor	La mayor
Livianas	Flamenco	Mi flamenco
Serranas	Flamenco	Mi flamenco
Soleá	Flamenco	Mi flamenco
Caña	Flamenco	Mi flamenco
Polo	Flamenco	Mi flamenco
Bulerías	Flamenco	La flamenco, Mi flamenco
Mayor	La mayor, Mi mayor	
Menor	Mi menor	
Bamberas	Flamenco	Mi flamenco
Alboreás	Flamenco	Mi flamenco
Jaleos	Flamenco	La flamenco
Gilianas	Flamenco	La flamenco
Tango	Flamenco	La flamenco
	Mayor	La mayor
	Menor	La menor
Tientos	Flamenco	La flamenco
Tanguillos	Mayor	La mayor, Mi mayor
Marianas	Flamenco	Mi flamenco, La flamenco

3. CANTES DE CÁDIZ O CANTIÑAS

PALO	MODO	TONO
Alegrías	Mayor	La mayor, Mi mayor, Do mayo
	Menor	La menor, Mi menor, Do menc
Caracoles	Mayor	Do mayor.
Mirabrás	Mayor	Mi mayor
Romeras	Mayor	Mi mayor

4. FANDANGOS

PALO	MODO	TONO
Fandangos de Huelva	Flamenco	Mi flamenco
	Mayor	La mayor
	Menor	La menor
Fandangos naturales	Flamenco	Mi flamenco, La flamenco
Malagueña	Flamenco	Mi flamenco
Verdiales	Flamenco	Mi flamenco
Jaberas	Flamenco	Mi flamenco
Rondeñas cantadas	Flamenco	Mi flamenco
Rondeñas para guitarra solista	Flamenco	Do # flamenco
Granaína y Media Granaína	Flamenco	Si flamenco

5. CANTES MINEROS Y DE LEVANTE

PALO	MODO	TONO
Taranta	Flamenco	Fa # flamenco
Taranto	Flamenco	Fa # flamenco
Cartagenera	Flamenco	Fa # flamenco
Minera	Flamenco	Fa # flamenco, Sol # flamenco
Murciana	Flamenco	Fa # flamenco
Levantica	Flamenco	Fa # flamenco

6. RELACIONADOS CON EL FOLKLORE ANDALUZ

PALO	MODO	TONO
Petenera	Flamenco – menor	Mi flamenco-La menor
Sevillanas	Cualquier modo	Cualquier tonalidad

7. DE IDA Y VUELTA O HISPANOAMERICANOS

PALO	MODO	TONO
Guajira	Mayor	La mayor, Re mayor
Colombiana	Mayor	La mayor, Re mayor
Milonga	Menor	
Rumba	Cualquier modo	Cualquier tonalidad
Vidalita	Menor	
Farruca	Menor	La menor
Garrotín	Mayor	La mayor, Do mayor

Anexo II : Tonalidad y modo de cada uno de los palos