Escuela Politécnica Superior

Estudio de sistemas de estimación automática de acordes en música digitalizada

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:
Pablo González Carrizo

Tutor/es:
José Manuel Iñesta Quereda

Junio 2016

Universitat d'Alacant
Universidad de Alicante

Estudio de sistemas de estimación automática de acordes en música digitalizada

Autor
Pablo González Carrizo

Director
José Manuel Iñesta Quereda
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Escuela Politécnica Superior

ALICANTE, 25 de junio de 2017

Prólogo

El presente Trabajo de Fin de Grado tiene como objetivo, como su propio nombre indica, el análisis de diversos sistemas de estimación automática de acordes a partir de música digitalizada. La obtención de los acordes presentes en una pista de audio es una tarea compleja que requiere, tanto conocimientos de ingeniería como conocimientos musicales. Para muchos músicos, entre los que me incluyo, sería de gran ayuda tener un sistema capaz de realizar esta tarea, en tiempo real, y con una precisión cercana al 100 %. Por ello, cuando la persona que ha tutorizado este trabajo, José Manuel Iñesta, me propuso investigar este tema, no dudé en aceptar.

Con la entrega de este trabajo concluye mi etapa de estudiante del Grado en Ingeniería de Sonido e Imagen en Telecomunicación de la Universidad de Alicante. Una etapa de mi vida que recordaré siempre con un inmenso cariño. Qué mejor forma de cerrarla, que uniendo mis dos pasiones, la ingeniería y la música.

Mi madre suele poner a sus alumnos de Lengua y Literatura, en el último examen que realizan antes de finalizar su paso por el instituto, un poema de Jaime Gil de Biedma que comienza así:

> Que la vida iba en serio
> uno lo empieza a comprender más tarde
> como todos los jóvenes,
> yo vine a llevarme la vida por delante.

En estos años he descubierto que el éxito no es conseguir dinero, ni ascender en el trabajo. Ni siquiera lo es recorrer el mundo o conseguir la fama. El éxito no está ni en las ciencias ni en las letras. El éxito es levantarte cada día sabiendo que amas lo que haces, sea lo que sea.

Quiero agradecer, en primer lugar, a mi tutor José Manuel Iñesta por toda la ayuda prestada, de un valor incalculable. Gracias por contestar a todos mis extensos correos y por ayudarme a poner los pies en la tierra cuando empezaba a tener delirios. Además, esto no podría haber sido posible sin la ayuda de José Javier Valero. De José Javier admiro su humildad y entrega a los demás. Muchísimas gracias por toda la paciencia que tuviste para contestar las largas listas de preguntas que tenía preparadas para hacerte. Suerte con tu tesis, aunque no la necesitas.

También quería agradecer a todos los profesores que he tenido a lo largo de estos años, para los que estoy seguro que la ingeniería era mucho más que sólo un trabajo, era una pasión.

No estaría hoy donde estoy sin el apoyo incondicional de mis padres. A mis padres tengo que agradecerles todo en esta vida. No puedo estar más orgulloso de ellos y me

esfuerzo cada día en que ellos lo puedan estar de mi. A mi hermana Diana le agradezco el saber que siempre podré contar con ella para lo que necesite. Además, estoy seguro de que vas a ser una estupenda médica. Tampoco me olvido de Sancho, era él quien más compañía me hacía en mis largas tardes de estudio. Además, doy las gracias a mis abuelos, mis tíos y mis primos que siempre han estado ahí, alegrándose incluso más que yo de mis propios éxitos.

A mi pareja, Rocío, le agradezco que haya hecho de estos últimos años, los mejores de mi vida.

A Bleda, Lucas y demás amigos tanto de Alicante como de Hellín, les pido perdón por lo pesado que he llegado a ser muchas veces con todas mis *cosas frikis*. Sobretodo estas últimas semanas, creo que teníais más ganas que yo de que terminara el TFG. Sin embargo, siempre habéis estado ahí, y he disfrutado como un niño con todos vosotros.

Por último, me gustaría agradecer a todas las personas con las que alguna vez he compartido escenario, o simplemente he pasado tardes tocando la guitarra junto a ellos. En especial a Pedro Pérez, mi primer profesor de guitarra, pues tú me transmitiste tu amor a la música.

A todos, espero que disfrutéis la lectura

Pablo González Carrizo

San Vicente del Raspeig, Junio de 2017

Abstract

Music transcription systems try to create the original sheets from audio files. This is a difficult task, even from human specialists, and doing it automatically is a work currently done by many researchers. In this project, we will focus on one concrete area of the music transcription: automatic chord estimation (ACE). We will analyze the main subsystems inside an automatic chord estimation system, and how, modifying theirs parameters, changes the result of the estimation.

A lot of researchers of this topic meet together, annually, in the MIREX event, where they share their works. Most new systems are based on deep learning. The problem of deep learning is that the system may work or not, but a set of values of neurons connections weights doesn't give us enough information to know what is happening in the system. In the future, most ACE algorithms will use deep learning, so obtaining more information about other parts of the process, and how they modify the final result of the estimation, will be very useful.

The analyzed subsystems have been divided in some categories depending on the part of the system where they are executed: signal adaptation, feature extraction, prefiltering, classification and postfiltering. We have analyzed the effect on the system of harmonic-percussive separation, signal filtering or beat synchronization. However, as we have seen, the more important parts of the system were the training dataset and the classifier used. We have used two different kind of training datasets. One of them comes from ideal theoretical models. The other one, comes from real samples taken from musical instruments. Three different classifiers have been analyzed: knn, support vector machine and a neural network with some hidden layers.

The system has a configuration file where the user can change all the parameters. Furthermore, a documentation file has been also created, and a report file generator, that will create a document with the results of an estimation. All of this was done to make the use of this system easier for other researchers that may use it as a backend for their projects.

Resumen

La transcripción musical supone la escritura de la partitura que origina una canción, a partir de una pista de audio. Esta es una tarea difícil incluso para los especialistas humanos, y realizarla de manera automática es un área de investigación abierta desde hace muchos años. En este trabajo nos centraremos en un aspecto concreto de la transcripción musical, la estimación de acordes. Realizaremos un análisis de los principales componentes de un sistema de estimación de acordes típico, y veremos cómo la modificación de sus distintos parámetros afecta el resultado de la estimación.

Muchos son los investigadores que han centrado sus esfuerzos en esta tarea. Desde hace más de 10 años, anualmente se suelen poner en común todos los trabajos realizados en este campo en el evento MIREX. Una gran parte de los algoritmos más avanzados de este campo presentados en los últimos años, están basados en el uso del deep learning o aprendizaje profundo. El problema del deep learning es que el sistema puede funcionar, o no, pero un conjunto de valores de pesos de conexiones neuronales no nos proporcionarán la información suficiente como para saber qué está pasando en nuestro sistema. Ya que el futuro de la estimación de acordes pasa inevitablemente por el deep learning, trabajos como éste permitirán arrojar algo de luz al proceso, permitiendo tener una información detallada de cómo afectan al sistema el resto de elementos y sus parámetros.

Los subsistemas analizados se han dividido en función de la fase del proceso en los que estos se encontraban: fase de adaptación de la señal, fase de extracción de características, fase de prefiltering, fase de clasificación y fase de postfiltering. Hemos analizado el efeco de la separación armónico-percusiva, el filtrado de la señal o la sincronización con el pulso. Sin embargo, hemos podido comprobar que los elementos que de verdad suponían importantes cambios en la estimación era el conjunto de datos de entrenamiento escogido y el clasificador usado. Con respecto a los conjuntos de datos de entrenamiento, se han utilizado tanto datos ideales teóricos como datos obtenidos a partir de muestras tomadas de instrumentos musicales. Con respecto al clasificador, se han analizado tres tipos principales, un clasificador knn, una máquina de vector de soporte y una red neuronal con varias capas ocultas.

Además,se ha complementado el sistema con un archivo que permite configurar todos los parámetros del mismo, una extensa documentación y un generador de documentos con los resultados de las ejecuciones del mismo. Todo ello, con la intención de que pueda ser utilizado por otros investigadores como base para la creación de sus propios sistemas de estimación automática de acordes.

Sin música,
la vida sería un error.

Friedrich Nietzsche.

Índice general

1. Introducción **1**
- 1.1. Origen del proyecto . . . 1
- 1.2. Metodología de trabajo . . . 2
- 1.3. Introducción a la extracción de información musical . . . 4
  - 1.3.1. ¿Por qué necesitamos la extración de información musical . . . 4
  - 1.3.2. Dimensiones de la extracción de información musical . . . 5
  - 1.3.3. ISMIR . . . 6
- 1.4. Introducción a la estimación automática de acordes (ACE) . . . 8
- 1.5. Sistemas ACE comerciales . . . 11
  - 1.5.1. Chordify . . . 11
  - 1.5.2. Riffstation . . . 12
  - 1.5.3. Capo . . . 13

2. Fundamentos musicales **15**
- 2.1. Intervalos . . . 15
  - 2.1.1. Consonancia y disonancia . . . 17
- 2.2. Acordes . . . 18
  - 2.2.1. Acordes de más de 3 notas . . . 19
  - 2.2.2. Inversiones . . . 20
- 2.3. Escalas . . . 20
- 2.4. Tonalidad . . . 22
  - 2.4.1. Armonía funcional . . . 23
- 2.5. Notación musical . . . 24
  - 2.5.1. Pentagrama . . . 25
  - 2.5.2. Representación de la altura . . . 25
  - 2.5.3. Representación de la duración . . . 26
  - 2.5.4. Representación de acordes . . . 27

3. Fundamentos físicos **31**
- 3.1. Física del sonido . . . 31
- 3.2. Transformada Discreta de Fourier . . . 33
- 3.3. Parciales y formantes: Timbre . . . 35
- 3.4. Ruido . . . 39
- 3.5. Evolución temporal del espectro . . . 39
- 3.6. Acústica de los instrumentos . . . 44
  - 3.6.1. Excitador . . . 44
  - 3.6.2. Resonador . . . 45

4. Métodos ACE **47**
4.1. Adaptación previa de la señal . . . 47
4.2. Extracción de características . . . 49
4.2.1. Cromagrama . . . 51
4.2.2. Información de la línea de bajo . . . 52
4.2.3. Afinación . . . 52
4.3. Prefiltering . . . 52
4.3.1. Suavizado temporal . . . 52
4.3.2. Sincronización con pulso . . . 55
4.4. Comparación con modelo . . . 55
4.4.1. Patrones ideales . . . 58
4.4.2. Patrones ideales con armónicos . . . 59
4.4.3. Patrones reales . . . 59
4.4.4. Función de distancia . . . 59
4.4.5. Aproximaciones de Aprendizaje Supervisado . . . 61
4.5. Postfiltering . . . 67
4.5.1. Información de tonalidad . . . 67
4.5.2. Sincronización con el pulso . . . 68
4.6. Estimación basada en modelos ocultos de Markov . . . 68
4.7. Evaluación de los resultados . . . 71

5. Diseño del sistema final **73**
5.1. Tecnologías usadas . . . 73
5.1.1. Python . . . 73
5.1.2. Latex . . . 77
5.2. Esquema general del sistema . . . 77
5.2.1. Configuración . . . 77
5.2.2. Estructura del proyecto . . . 78
5.3. Adaptación de la señal . . . 79
5.3.1. Separación armónica percusiva . . . 79
5.3.2. Filtrado de armónicos . . . 80
5.4. Extracción de características . . . 81
5.4.1. stft . . . 81
5.4.2. cqt . . . 83
5.4.3. cens . . . 83
5.5. Prefiltering . . . 83
5.5.1. Filtrado del cromagrama . . . 83
5.5.2. Detección de beats . . . 83
5.5.3. Sincronización con beats . . . 84
5.6. Patrones . . . 84
5.6.1. Patrones ideales . . . 84
5.6.2. Patrones ideales con armónicos . . . 85
5.6.3. Patrones a partir de soundfonts . . . 85

5.7. Estimación 85
5.7.1. KNN 89
5.7.2. Support Vector Machine 89
5.7.3. Red neuronal 89
5.8. Postfiltering: Sincronización con beat 90
5.9. Evaluación 90
5.9.1. Datasets 90
5.10. Visualización de resultados 92
5.11. Documentación 94
5.12. Repositorio de código 94

6. Evaluación del sistema **95**
6.1. Evaluación de los subsistemas 95
6.1.1. Configuración básica 95
6.1.2. Separación armónica percusiva 95
6.1.3. Filtrado de armónicos 98
6.1.4. Extracción de características: Parámetros y tipos de cromagrama . 99
6.1.5. Prefiltering: Filtro de mediana 102
6.1.6. Prefiltering: Sincronización con beat 103
6.1.7. Postfiltering: Sincronización con beat 104
6.1.8. Patrones y clasificadores 104
6.1.9. Clasificadores 105
6.2. Sistema final 107
6.2.1. Velocidad de ejecución 107

7. Conclusiones **109**

Índice de figuras

1.1. Partitura de un fragmento de la canción Country Gardens con anotaciones de acordes . . . 2
1.2. Metodologías tradicionales . . . 3
1.3. Metodologías ágiles . . . 3
1.4. Imagen promocional del software TuneUp, que permite añadir metadatos a archivos MP3. . . . 5
1.5. Wiki MIREX 2016, con resultados de los algoritmos de estimación de acordes . . . 8
1.6. Ejemplo de fichero de salida de un algoritmo de ACE . . . 9
1.7. Ejemplo de cromagrama . . . 10
1.8. Web de Chordify . . . 12
1.9. Chordify player . . . 13

2.1. Intervalos posibles desde la nota Do . . . 16
2.2. Ejemplos de acordes aumentados y disminuidos . . . 19
2.3. Ejemplos de acordes suspendidos . . . 19
2.4. Inversiones del acorde de Do séptima dominante . . . 20
2.5. Escala de Do mayor . . . 21
2.6. Escala de Re mayor . . . 21
2.7. Escala de La menor . . . 21
2.8. Escala de Re menor . . . 22
2.9. Escala de La menor armónica . . . 22
2.10. Escala de La menor melódica . . . 22
2.11. Escala pentatónica mayor de Do . . . 23
2.12. Escala pentatónica menor de La . . . 23
2.13. Do4 representado en las claves más comunes . . . 25
2.14. Duraciones de los distintos símbolos . . . 26
2.15. Ejemplo de cifrado americano . . . 28

3.1. Presión en función del tiempo para un punto determinado . . . 32
3.2. Resultado de la combinación de dos ondas sinusoidales con una diferencia de fase de π . . . 33
3.3. Descomposición de una señal compleja en varias señales sinusoidales . . . 34
3.4. Serie armónica . . . 36
3.5. Nota A4 interpretada por una flauta travesera y por un piano . . . 37
3.6. Espectros de la nota A4 interpretada por una flauta travesera y por un piano . . . 38

3.7. Espectro producido por una voz pronunciando la vocal i 39
3.8. Comparación de espectro de onda sinusoidal sin ruido y onda sinusoidal con ruido 40
3.9. Ventana triangular aplicada a señal 42
3.10. Espectrograma de una señal armónica 43

4.1. Principales tareas de sistema de estimación automática de acordes 48
4.2. Comparación del espectro de la parte armónica y de la parte percusiva de los primeros segundos de la canción "Love Me Do"de The Beatles 50
4.3. Diferencia entre espectrograma (arriba) y cromagrama (abajo) 53
4.4. Sistema de obtención de cromagramas cuantizados, independientes de la afinación 54
4.5. Diferencia entre espectrograma antes del suavizado mediante un filtro de mediana (arriba) y después del mismo (abajo) 56
4.6. Cromagrama antes y después de sincronizar con el beat 57
4.7. Diagrama de sistema de comparación con patrones de acordes 57
4.8. Matriz de patrones de acorde mayores ideales 58
4.9. Pesos de cada nota en el patrón calculado con 6 armónicos del acorde de Do Mayor. 60
4.10. Diagrama básico de uso de conjunto de datos de entrenamiento y conjunto de datos de validación 62
4.11. Efecto de cambiar el parámetro k de un KNN 64
4.12. Ejemplo de SVM 65
4.13. Red neuronal con 3 capas 66
4.14. Diagrama de red neuronal recurrente 66
4.15. Ejemplo de sincronización de la estimación con el pulso en postfiltering. Cada cuadro representa un pulso. En el esquema de abajo vemos el resultado de sincronizar con el pulso el conjunto de acordes mostrados en el diagrama superior. 68
4.16. Ejemplo de Cadena de Markov para 3 acordes 69
4.17. HMM con 4 estados que pueden emitir 2 símbolos discretos y_1 or y_2. a_{ij} es la probabilidad de transición desde el estado s_i al estado s_j. $b_j(y_k)$ es la probabilidad de emitir el símbolo y_k en el estado s_j. 70

5.1. Logo del lenguaje Python 74
5.2. Componentes del sistema ACE implementado 78
5.3. Estructura del proyecto 79
5.4. Diagrama de flujo de la función *harmonic()* 80
5.5. Espectro de filtro Butterworth de orden 1 81
5.6. Diferentes tipos de cromagramas extraídos de un mismo fragmento de audio 82
5.7. Diagrama de flujo de la obtención de los beats 84
5.8. Pesos de cada nota en los 5 patrones ideales implementados 86
5.9. Pesos (normalizados) de cada nota en los 5 patrones con armónicos implementados 87

5.10. Diagrama de generación de patrones a partir de SoundFonts 88
5.11. Sistema de evaluación de los resultados 91
5.12. Fragmento del archivo generado por el sistema implementado 93
5.13. Documentación auto-generada por Sphinx 94

6.1. Configuración básica del sistema 96
6.2. Puntuación *root* en función del margen en el HPSS 97
6.3. Puntuación *root* en función del tamaño del kernel en el HPSS 99
6.4. Configuración del sistema usando filtrado de armónicos 100
6.5. Puntuación *root* en función de la frecuencia de corte del filtrado de armónicos 101
6.6. Configuración del sistema, usando sincronización con beat en prefiltering . 103
6.7. Configuración del sistema final 108

Índice de tablas

2.1. Acordes de séptima 19
2.2. Acordes creados a partir de la escala de Do Mayor 24
2.3. Acordes creados a partir de la escala de Do Mayor 29

5.1. Discos del dataset de The Beatles junto con su duración 91

6.1. Puntuaciones de la configuración básica 96
6.2. Puntuaciones tras modificación margen HPSS 97
6.3. Puntuaciones tras modificación del tamaño del kernel del HPSS 98
6.4. Puntuaciones *root* tras eliminación del bloque HPSS 98
6.5. Puntuaciones tras filtrado de armónicos 101
6.6. Puntuaciones en función del tipo de cromagrama 102
6.7. Puntuaciones en función del hop_length utilizado para el cálculo del cromagrama 102
6.8. Puntuaciones en función del tamaño del kernel usado en el filtro de mediana de prefiltering 102
6.9. Puntuaciones sin el filtro de mediana de prefiltering 103
6.10. Puntuaciones obtenidas usando sincronización con el beat en prefiltering . 104
6.11. Puntuaciones obtenidas usando sincronización con el beat en postfiltering sin subdividr el beat 104
6.12. Puntuaciones en función de los tipos de patrones elegidos 105
6.13. Puntuaciones en función de la k del clasificador KNN 106
6.14. Puntuaciones en función del parámetro C de la SVM 106
6.15. Puntuaciones en función del parámetro α y el número de capas ocultas de la red neuronal 107
6.16. Puntuaciones del sistema final 107

1 Introducción

La **armonía** puede definirse como el equilibrio de las proporciones entre las distintas partes de un todo. En música, la armonía la crean las distintas voces que suenan simultáneamente, es decir, los **acordes**, y la forma en la que éstos progresan a lo largo de una obra. Llamamos acorde al conjunto de 3 o más notas musicales que suenan de forma simultánea y que, juntas, conforman una unidad armónica. Una sucesión de acordes, se denomina progresión armónica. En la figura 1.1 vemos un ejemplo de la partitura de una canción junto con anotaciones de su progresión armónica.

La armonía es considerada como la ciencia que enseña a constituir los acordes y que sugiere la manera de combinarlos de la forma más equilibrada, consiguiendo así sensaciones de relajación y sosiego (armonía consonante), o de tensión (armonía disonante). Es el elemento que permite que la música transmita un mensaje y que permanece inmutable a las distintas variaciones que se puedan interpretar de una obra musical. En muchas ocasiones, únicamente conociendo su armonía, varios músicos serán capaces de interpretar una canción, sin que esta pierda su *esencia*. Por lo tanto, si conseguimos obtener la progresión armónica de una obra musical, estaremos obteniendo información con un gran valor. Además, conocer los acordes de una canción nos será también de utilidad en otras tareas como detección de tonalidad, transcripción de audio a partitura, clasificación del género o alineamiento de audio y letra.

Sin embargo, la obtención de las progresiones de acordes presentes en una canción, puede llegar a ser un trabajo costoso que requiere de varios expertos y con un tiempo aproximado de 8 a 18 minutos por canción. En ciertas canciones este proceso puede durar más de una hora. [Burgoyne et al., 2011] Por ello, este trabajo tiene como objetivo el análisis de sistemas automáticos para la estimación de acordes a partir de música digitalizada. El objetivo final es, a partir de un conjunto de ficheros de audio, generar para cada uno un listado de acordes junto con los tiempos en los que éstos suenan. Se probarán varios algoritmos para cada una de las partes del sistema, evaluando cada vez la capacidad de predicción del sistema a partir de un conjunto de canciones de las que previamente conocemos su *ground-truth*.

1.1. Origen del proyecto

Este proyecto nace como continuación a dos Trabajos de Fin de Grado realizados por Miguel Segura Sogorb [Segura Sogorb, 2015] y Daniel Gil Ortiz [Gil Ortiz, 2016], y tutorizados, al igual que este trabajo, por José Manuel Iñesta. En ambos trabajos, se analizan varios sistemas básicos de estimación de acordes a partir de audio, que serán el punto de partida para la realización de nuestro sistema.

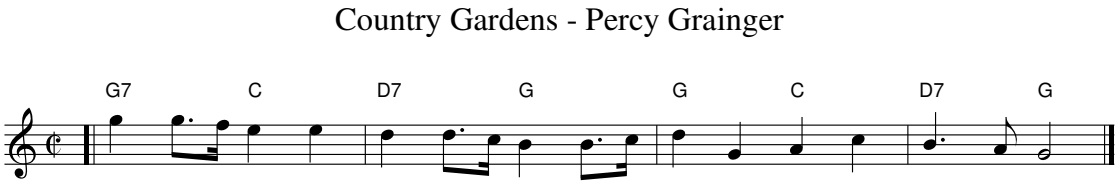


Figura 1.1: Partitura de un fragmento de la canción Country Gardens con anotaciones de acordes

Las ventajas de no partir sobre un *lienzo en blanco* son evidentes, y gracias a ello se ha podido llegar mucho más lejos de lo que se hubiera llegado teniendo que plantearse el diseño del sistema desde cero. Sin embargo, también se corre el riesgo de interpretar todas las suposiciones que se realizan en dichos trabajos como verdades absolutas o de arrastrar ciertos errores de concepto o de programación que existieran en el trabajo original. Por ello, desde un primer momento se decidió no empezar exactamente donde terminaron los trabajos anteriormente citados, sino que se intentaría reproducir sus pasos desde el inicio, planteando y evaluando otras posibles opciones en cada una de las fases del sistema.

1.2. Metodología de trabajo

En este proyecto se ha utilizado una forma de trabajar similar a las **metodologías ágiles** de desarrollo de software. Estas metodologías buscan un desarrollo rápido y eficiente. Están basadas en ciclos de evolución de software incrementales en los que se postponen las decisiones lo más posible hasta haber obtenido un feedback del cliente. En nuestro caso no tenemos un feedback de un cliente, pero sí tenemos un método de evaluación que nos dirá qué tasa de acierto estamos obteniendo con nuestro sistema y nos permitirá evaluar la eficacia de cada uno de los cambios que realicemos en él. Las fases en las que se basaría un proyecto común en este campo (revisión bibliográfica, desarrollo, evaluación y documentación) se han repetido en numerosas ocasiones, para cada iteración en el desarrollo del sistema. Cuando hablamos de iteración nos referimos a cualquier cambio que se realice en el programa que pueda suponer una mejora en la tasa de acierto. Esta forma de trabajar ha permitido tener que evitar hacer una estimación, al comienzo del proyecto, de cuánto tiempo dedicaremos al diseño y desarrollo del sistema, y cuanto a la realización de la memoria. Esta estimación es complicada de realizar, y equivocarnos al realizarla supone no poder dedicarle el tiempo necesario a la memoria del trabajo, o bien, obtener un programa con menos funcionalidades de las que podrían haberse implementado. De esta forma, cada nueva funcionalidad es documentada en la memoria poco después de haberse terminado de programar. En las figuras 1.3 y 1.2 podemos ver las diferencias entre metodologías tradicionales y las metodologías ágiles.

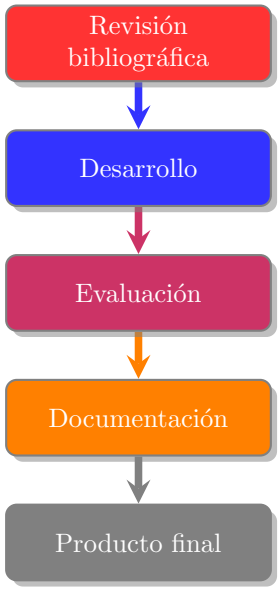


Figura 1.2: Metodologías tradicionales

Figura 1.3: Metodologías ágiles

1.3. Introducción a la extracción de información musical

Vivimos en la **Era de la Información** y, como en muchos otros campos, en la música podemos ver claramente el efecto de la **Revolución Digital**, que supuso el principio de esta era. La cantidad de música digital que se produce cada día crece de forma exponencial. Tenemos dispositivos asequibles, capaces de almacenar miles de canciones. Llevamos estos dispositivos con nosotros a todos sitios, y acompañamos de música muchas de nuestras tareas cotidianas. Al ser capaces de almacenar la música como simples conjuntos de unos y ceros, podemos compartirla o copiarla tantas veces como queramos, obteniendo todas las veces copias idénticas a la original. De ahí, que grandes plataformas de streaming nos permitan acceder al momento a millones de canciones que ni siquiera disponemos, sino que están alojadas en servidores en cualquier parte del mundo.

Además, vivimos en una época de democratización de la música. La grabación de música ha dejado de ser un privilegio de unos pocos y con unos medios muy limitados cualquiera puede ser capaz de grabarse sus propios temas, con una calidad que no tiene por qué alejarse demasiado de las grabaciones de los grandes sellos discográficos.

A partir de esta revolución, nace una nueva área de investigación basada en la extracción de información musical, habitualmente conocida como **MIR (Music Information Retrieval)**. Esta área se engloba dentro de otra área mayor, **Information Retrieval**. Todas las investigaciones y herramientas relacionadas con esta disciplina tiene un enfoque común: Tratar de describir documentos por medio de características obtenidas de ellos. De esa necesidad nace el concepto de **metadatos**, datos sobre los datos. Los metadatos suponen una nueva dimensión que aporta un valor añadido a un documento. En el caso concreto de la extracción de información musical, engloba a especialistas de distintos campos como la musicología, el tratamiento de señal o la inteligencia artificial. Todos ellos trabajan juntos para lograr solventar la gran carencia de metadatos que existe en la música.

Los principales estándares de audio, como son el CD y el MP3, no contemplan la posibilidad de almacenar también información estructurada adicional. Años después del lanzamiento del formato MP3, a raíz de contribuciones externas al estándar, se añadieron las etiquetas ID3, que permitían añadir algunos metadatos básicos. Incluso existe software, como TuneUp 1.4, que permite añadir ciertos metadatos a archivos MP3. Sin embargo, estos no son suficientes, y además no están presentes en muchos archivos. Por lo tanto, toda esa gran cantidad de música que comentábamos antes que se genera actualmente, se está generando con una escasa cantidad de metadatos asociada. Se está perdiendo la oportunidad de multiplicar el valor de esos archivos añadiendo unos pocos bytes, y esto supone un gran reto para el campo de investigación del MIR.

1.3.1. ¿Por qué necesitamos la extración de información musical

Un estudio sobre las necesidades de información en la música de 427 estudiantes universitarios [Lee and Downie, 2004] reveló datos muy interesantes. Según los datos obtenidos, la mayoría de los usuarios buscaban información adicional para complementar su audición de la música. Esta información estaba relacionada con el artista, el género,

Figura 1.4: Imagen promocional del software TuneUp, que permite añadir metadatos a archivos MP3.

la letra de la canción, artistas similares o las recomendaciones de otros usuarios. En el estudio, se sugieren dos tipos de metadatos necesarios.

- **Metadatos de contenido**: Incluyen información obtenida a partir de la propia música (melodía, tiempo, acordes,etc) e información bibliográfica (título, autor, disco, etc)
- **Metadatos de contexto**: Aquí se incluyen todos los metadatos referidos a la relación con otras canciones o artistas similares y los referentes a las asociaciones con otros trabajos o eventos (uso de las canciones en películas, anuncios, etc)

Además de las necesidades de los usuarios comunes que necesitan de los metadatos para tareas como encontrar temas similares, obtener información de los autores u organizar sus bibliotecas de música personales, existen las necesidades concretas de ciertos grupos de personas con una relación de mayor importancia con la música. Ejemplo de ellos pueden ser personas que necesitan encontrar música con ciertas características para sus actividades profesionales o incluso los propios músicos. Para estos últimos, los metadatos de contenido pueden ser imprescindibles. La obtención de una transcripción completa de una obra, de las progresiones de acordes o de simplemente el compás y la tonalidad de una obra son de gran ayuda para ellos.

1.3.2. Dimensiones de la extracción de información musical

En un artículo sobre MIR en el Annual Review of Information Science and Technology [Downie, 2003] se definen siete dimensiones principales dentro de la extracción de información musical.

1. **Tono**: El tono es la altura del sonido percibida, que depende de su frecuencia fundamental, es decir, del número de oscilaciones por segundo. La diferencia de

altura entre dos tonos da lugar a un intervalo, que dependiendo de la distancia entre ambos; medida en semitonos; pertenece a un tipo concreto. Además, dentro de esta dimensión se encontraría la tonalidad, definida como el sistema de sonidos que sirve de fundamento a una composición musical.

2. **Tiempo**: Aquí se incluye la duración de las notas, la velocidad del pulso, el compás, la duración de las armonías o los acentos.

3. **Armonía**: Cuando dos o más alturas suenan a la vez, se produce una armonía. La combinación de armonías en una obra consigue que el autor transmita sus emociones creando en el oyente momentos de tensión y de relajación. La armonía será la base de la melodía y el acompañamiento de la canción. El presente trabajo se desarrollará precisamente sobre esta dimensión.

4. **Timbre**: Es la cualidad del sonido que nos permite conocer el elemento que está produciéndolo. Podemos distinguir una voz o un instrumento porque conocemos su timbre. Físicamente, esta propiedad está basada en el espectro del sonido, y en la presencia en él de armónicos.

5. **Editorial**: La dimensión editorial incluye todas las instrucciones para la interpretación de una obra. Ejemplo de ellas podrían ser las digitaciones, las instrucciones de dinámica o los ornamentos.

6. **Textual**: El elemento principal de esta dimensión son las letras de las canciones.

7. **Bibliográfica**: Contiene la información sobre el título de la obra, el compositor, los intérpretes, la discográfica, etc. Esta es la única dimensión cuyos datos no se pueden obtener de la propia composición.

1.3.3. ISMIR

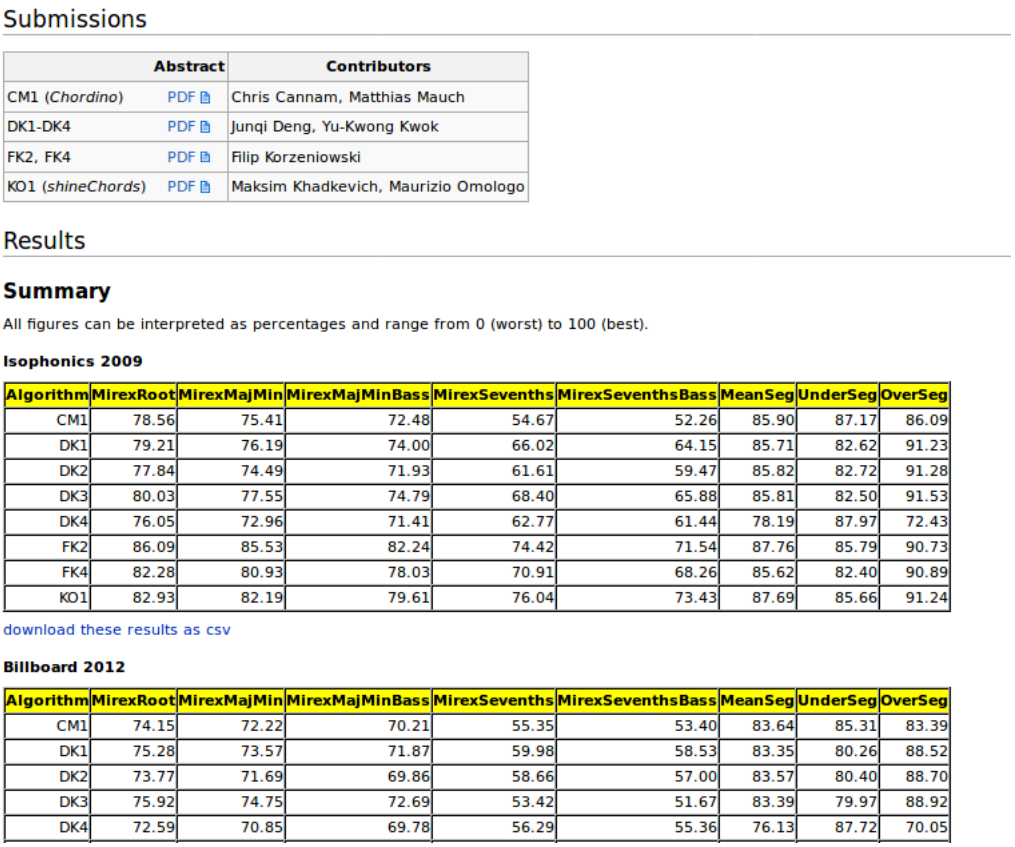
A raíz de esta necesidad de extracción de información en la música nace en el año 2000 las International Society for Music Information Retrieval (ISMIR). ISMIR es una asociación sin ánimo de lucro de investigadores de este campo, que cuenta actualmente con más de 1800 miembros. Una de sus principales tareas es la organización de una conferencia anual, cuya localización es diferente cada año, que reúne a los más importantes investigadores de Music Information Retrieval con la intención de compartir conocimiento. Como parte de estos eventos, desde 2005 se desarrollan los **MIREX** (Music Information Retrieval Evaluation eXchange). En estas sesiones, los algoritmos de MIR de investigadores de todo el mundo *compiten* por ver quien obtiene mejores resultados en las diferentes categorías. El número de categorías ha crecido desde las 10 categorías que se evaluaron la primera vez que se celebró este evento hasta las 24 categorías que existieron en el 2016. Las categorías evaluadas fueron las siguientes:

- Grand Challenge on User Experience (J-DISC)
- Grand Challenge on User Experience

- Audio Classification (Train/Test) Tasks, incorporating:
- Audio K-POP Genre Classification
- Audio Cover Song Identification
- Audio Tag Classification
- Audio Music Similarity and Retrieval
- Symbolic Melodic Similarity
- Audio Onset Detection
- Audio Key Detection
- Real-time Audio to Score Alignment (a.k.a Score Following)
- Query by Singing/Humming
- Audio Melody Extraction
- Multiple Fundamental Frequency Estimation and Tracking
- Audio Chord Estimation
- Query by Tapping
- Audio Beat Tracking
- Structural Segmentation
- Audio Tempo Estimation
- Discovery of Repeated Themes and Sections
- Audio Downbeat Estimation
- Audio Fingerprinting
- Singing Voice Separation
- Set List Identification
- Audio Offset Detection

Además en su web oficial [1] encontramos una wiki con todas las tareas evaluadas en todos los años de celebración del MIREX, con los resultados de los diferentes algoritmos que compitieron en cada una de ellas. En la figura 1.5, vemos una captura de la wiki de

[1] `http://www.music-ir.org/mirex/`



Submissions

	Abstract	Contributors
CM1 (*Chordino*)	PDF	Chris Cannam, Matthias Mauch
DK1-DK4	PDF	Junqi Deng, Yu-Kwong Kwok
FK2, FK4	PDF	Filip Korzeniowski
KO1 (*shineChords*)	PDF	Maksim Khadkevich, Maurizio Omologo

Results

Summary

All figures can be interpreted as percentages and range from 0 (worst) to 100 (best).

Isophonics 2009

Algorithm	MirexRoot	MirexMajMin	MirexMajMinBass	MirexSevenths	MirexSeventhsBass	MeanSeg	UnderSeg	OverSeg
CM1	78.56	75.41	72.48	54.67	52.26	85.90	87.17	86.09
DK1	79.21	76.19	74.00	66.02	64.15	85.71	82.62	91.23
DK2	77.84	74.49	71.93	61.61	59.47	85.82	82.72	91.28
DK3	80.03	77.55	74.79	68.40	65.88	85.81	82.50	91.53
DK4	76.05	72.96	71.41	62.77	61.44	78.19	87.97	72.43
FK2	86.09	85.53	82.24	74.42	71.54	87.76	85.79	90.73
FK4	82.28	80.93	78.03	70.91	68.26	85.62	82.40	90.89
KO1	82.93	82.19	79.61	76.04	73.43	87.69	85.66	91.24

download these results as csv

Billboard 2012

Algorithm	MirexRoot	MirexMajMin	MirexMajMinBass	MirexSevenths	MirexSeventhsBass	MeanSeg	UnderSeg	OverSeg
CM1	74.15	72.22	70.21	55.35	53.40	83.64	85.31	83.39
DK1	75.28	73.57	71.87	59.98	58.53	83.35	80.26	88.52
DK2	73.77	71.69	69.86	58.66	57.00	83.57	80.40	88.70
DK3	75.92	74.75	72.69	53.42	51.67	83.39	79.97	88.92
DK4	72.59	70.85	69.78	56.29	55.36	76.13	87.72	70.05

Figura 1.5: Wiki MIREX 2016, con resultados de los algoritmos de estimación de acordes

MIREX 2016 [2] en la que se muestran los algoritmos de estimación de acordes presentados y sus puntuaciones en los diferentes datasets usados. Además, se proporcionan los papers de estos algoritmos, los datasets usados y una clara explicación del método que se ha seguido para su evaluación, de forma que cualquier investigador podría evaluar su algoritmo y compararse con los algoritmos del estado del arte.

1.4. Introducción a la estimación automática de acordes (ACE)

Este trabajo está enfocado a una de las tareas nombradas anteriormente que ha formado parte de MIREX desde 2008, la estimación automática de acordes a partir de señales de audio (Automatic Chord Estimation, **ACE**).

Como se ha comentado anteriormente, la armonía es uno de los principales elementos

[2] `http://www.music-ir.org/mirex/wiki/2016:Audio_Chord_Estimation_Results`

0.301859410431	1.056984911752	**C:maj**
1.056984911752	2.598655121482	**A:min**
2.598655121482	4.259684892895	**F:maj**
4.259684892895	5.969525225984	**G:maj**
5.969525225984	6.559198562895	**C:maj**
6.559198562895	6.806292985251	**A:min**
6.806292985251	7.055932156412	**G:maj**
7.055932156412	7.458221265228	**F:maj**
7.458221265228	7.951289298292	**C:maj**

Figura 1.6: Ejemplo de fichero de salida de un algoritmo de ACE

que define una canción. Es el elemento que permanece inmutable a las distintas versiones que se realicen de esta e incluso permite a varios músicos interpretarla sin una partitura completa. La meta de un algoritmo de este tipo es obtener, a parir de una pista de audio, un archivo con metadatos con una estructura similar al de la figura 1.6.

Cada fila de este fichero contiene:

- Tiempo de inicio del acorde
- Tiempo de fin del acorde
- Nombre y tipo del acorde

Esta forma de mostrar el resultado de un algoritmo ACE es ampliamente usada, siendo guardado el archivo comúnmente con la extensión ***.lab***. Esta extensión fue creada por los creadores del software **wavesurfer**, que es capaz de leerla y mostrar los acordes en la posición adecuada junto con la pista de audio.

Aunque sea un campo de estudio que lleve ya varios años con nosotros, y exista una gran cantidad de algoritmos distintos, los investigadores no han conseguido dar con la clave para resolver ciertos problemas que surgen a la hora de estimar a qué acorde corresponden una serie de muestras de audio. Las entidades armónicas se crean en nuestro cerebro, el cual integra todos los sonidos que le llegan en un determinado rango de tiempo, incluso si estos no se han interpretado simultáneamente. Uno de los principales problemas es que no todas las notas que suenan en un determinado momento de una canción pertenecen al acorde correspondiente a esa parte. Es muy común el uso, en determinados momentos de una pieza musical, de notas externas al acorde. Estas notas nos permiten crear tensiones, obtener melodías suaves sin cambios bruscos o preparar la transición al siguiente acorde. Este hecho complica la estimación automática, pudiendo confundir al sistema, que piensa que esas notas deben ser parte del acorde. Además, como se analizará más adelante, cuando se ejecuta una nota concreta en un instrumento, sea el que sea, nunca sonará una sola frecuencia; sonarán varias frecuencias distintas, que

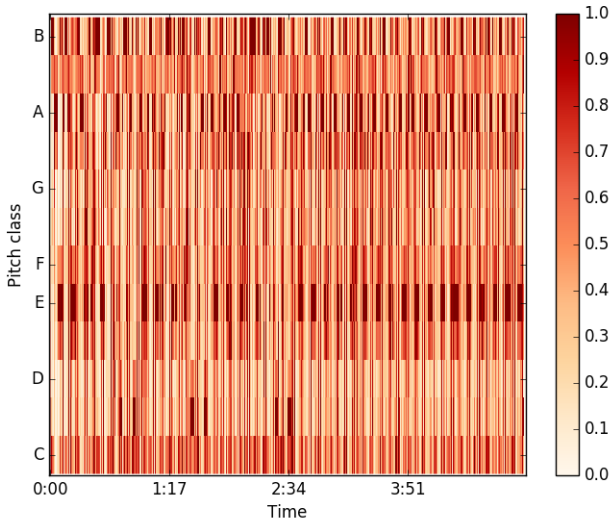


Figura 1.7: Ejemplo de cromagrama

pueden tener o no alguna relación entre sí, y que son una fuente de problemas para estos algoritmos.

Los sistemas ACE suelen combinar técnicas de procesado digital de señal y técnicas de aprendizaje automático. Estos algoritmos, como es común en muchos otros campos, comienzan con una **etapa de extracción de características** de los audios, a partir de las cuales serán capaces de detectar qué acorde suena en cada momento. La principal característica usada por la mayoría de estos algoritmos es el **cromagrama** que no es más que un espectrograma con las octavas y alturas sumadas. Un ejemplo de un cromagrama es el que se puede ver en la figura 1.7 A esto le sigue una **etapa de comparación con un modelo**, que ha sido implementada de formas muy distintas como veremos en capítulos posteriores. En esta etapa se comparan las características extraídas del audio con un modelo que puede ser teórico, o extraído del propio dataset, para así ser capaces de estimar a qué clase pertenece cada una, es decir, a qué acorde corresponden.

A estas dos fases principales, se han ido incorporando a lo largo del tiempo subtareas que suponen mejoras en la estimación como la sincronización de los cromagramas con los beats, el suavizado de los cromagramas, el uso de decodificadores de Viterbi, el uso de redes bayesianas y un largo etcétera de funcionalidades que se analizarán más adelante.

Para la evaluación de estos algoritmos, existen en la red numerosos datasets realizados por expertos que nos sirven como ground truth. Hasta hace no demasiados años, no existía esta amplia cantidad de canciones con sus acordes etiquetados y existía un dataset con 180 canciones de los Beatles de Isophonics [Harte, 2010] [3], considerado durante mu-

[3] `http://www.isophonics.net/content/reference-annotations-beatles`

cho tiempo como el dataset de referencia. La reciente aparición de nuevos datasets como la colección Billboard [Burgoyne et al., 2011] [4] con más de 1000 canciones etiquetadas, ha evitado algoritmos sobreajustados que nos hacían tener una percepción demasiado positiva de cuál es el estado del arte actual.

Desde el 66 % de precisión en la detección de únicamente acordes mayores y menores de los algoritmos presentados a MIREX 2005[Bello and Pickens, 2005], se han ido mejorando progresivamente los algoritmos hasta llegar en MIREX 2016 a un 88 % de eficacia de los algoritmos en algunos datasets[Korzeniowski and Widmer, 2016]. Este último resultado fue conseguido con técnicas de aprendizaje máquina, mediante redes neuronales convolucionales.

Para muchos autores hemos llegado a un límite práctico, lo que se conoce como *glass ceiling*[Benetos et al., 2012]. Estos autores justifican que los métodos de transcripción actuales no son capaces de captar toda la riqueza de las señales de audio de canciones. Las soluciones que proponen pasan por añadir más información al proceso como información de tonalidad, información sobre el género musical o sobre el instrumento. Incluso se proponen algoritmos de transcripción semiautomáticos en los que la interacción del usuario es necesaria para corregir los errores que produce el usuario y que el algoritmo aprenda de estos errores. De aquí nace el concepto de transcripción multimodal, que propone la integración de varias fuentes de información para mejorar la precisión del sistema.

1.5. Sistemas ACE comerciales

Aunque la estimación automática de acordes suele ser un campo más de investigación que de uso comercial, en el mercado existen varios sistemas que realizan esta tarea. En esta sección nos centraremos únicamente en aplicaciones de estimación de acordes con propósito comercial. Existen algunas webs que muestran, con un fin académico, demostraciones de implementaciones de sistemas ACE concretos, pero no serán analizadas en esta sección.

1.5.1. Chordify

Chordify [5] es una de las aplicaciones comerciales de estimación de acordes más conocidas. En la figura 1.8 podemos ver la interfaz principal de su web. Permite obtener los acordes de canciones que pueden ser subidas por el usuario u obtenidas a partir de un enlace a servicios como Youtube, Deezer o Soundcloud. Además, los acordes analizados se muestran, a la vez que se reproduce la canción, de forma que el usuario puede interpretar la canción a la vez que suena, como podemos ver en la figura 1.9.

La detección de acordes en Chordify, es posible gracias al uso de la herramienta Chordino [Mauch and Dixon, 2010a]. Chordino [6] es un plugin VAMP [7], un tipo de plugin que

[4] `http://ddmal.music.mcgill.ca/research/billboard`
[5] `https://chordify.net/`
[6] `http://isophonics.net/nnls-chroma`
[7] `http://www.vamp-plugins.org/`

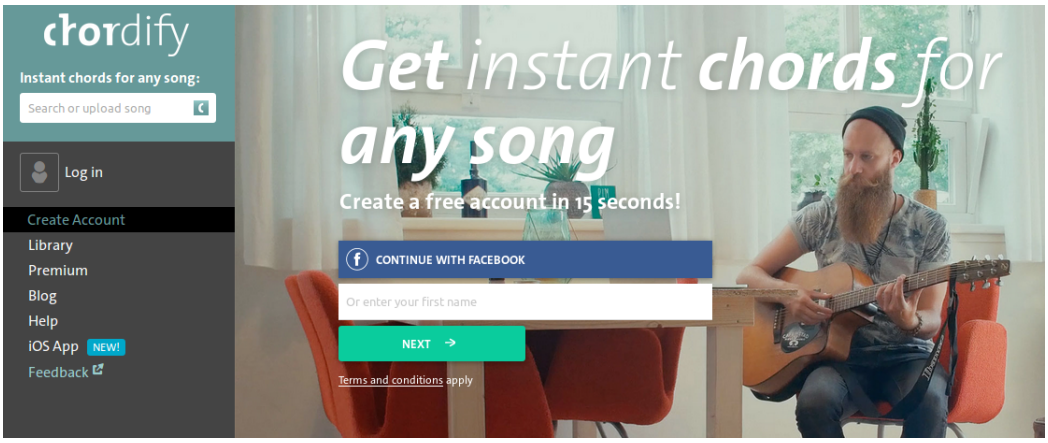


Figura 1.8: Web de Chordify

puede ser ejecutado por herramientas como Audacity o Sonic Annotator, que es capaz de obtener los acordes presentes en una canción. Chordino es capaz de realizar la extracción de características gracias al plugin NNLS Chroma creado por Matthias Mauch [Mauch and Dixon, 2010a]. Como ayuda a la estimación de los acordes, se hace uso del programa HarmTrace [Magalhaes and de Haas, 2011] [8], escrito en Haskell siguiendo el paradigma de programación funcional. Este programa permite aplicar la información de la tonalidad en casos en los que haya cierta incertidumbre sobre cuál es el acorde que está sonando.

Chordify cuenta también con una versión Premium, de pago, que ofrece interesantes características adicionales como la posibilidad de descargar el resultado en formato midi, la eliminación del límite en la subida de archivos o la posibilidad de transponer la canción.

1.5.2. Riffstation

La empresa Riffstation tiene una aplicación web capaz de detectar los acordes de una canción a partir de un enlace a Youtube [9]. Al igual que Chordify, permite ver, mientras que suena la canción, cuáles son sus acordes. Además, la aplicación también permite observar los diagramas de los acordes que suenan.

La fuente de ingresos de Riffstation viene de la integración de este sistema de acordes en un software completo de entrenamiento para guitarristas, que venden a través de su web. Este software, además de la detección de acordes tiene otras utilidades que permiten aislar instrumentos de una canción, cambiar su velocidad o añadir un metrónomo a la reproducción de una canción.

Aunque en su web no dice nada del algoritmo usado, la empresa asegura tener un 85 %

[8] Se puede encontrar en la siguiete url: `https://hackage.haskell.org/package/HarmTrace`
[9] `https://play.riffstation.com/`

Figura 1.9: Chordify player

de acierto en canciones pop/rock.

1.5.3. Capo

Capo es un software de pago creado por la compañía Supermegaultragroovy [10]. Este software contiene una serie de herramientas que ayudarán al músico a interpretar canciones. Entre esas herramientas se encuentra un sistema de detección de acordes. El software puede ser ejecutado en Mac, iPhone o iPad. En un artículo en su blog [11], sus creadores explican cómo ha sido la creación del sistema de detección de acordes. El sistema inicial, basado en un paper de un sistema ACE de la Queen Mary University de Londres [Stark and Plumbley, 2009], ha experimentado diversas modificaciones hasta llegar a la versión actual. Entre esas modificaciones encontramos la sincronización de la detección con los beats, y la incorporación un sistema estadístico para predecir qué acorde es más probable que suene en cada momento.

[10] `http://supermegaultragroovy.com`
[11] `http://supermegaultragroovy.com/2014/07/18/chord-intelligence/`

2 Fundamentos musicales

En este capítulo, sin pretender realizar un manual de referencia sobre teoría de la música, se explicarán algunos conceptos necesarios para comprender la utilidad y el funcionamiento del sistema de estimación automática de acordes propuesto.

2.1. Intervalos

En música, un intervalo se define como la distancia entre dos tonos. El intervalo básico es la **octava**. Una octava es la distancia entre un sonido armónico y otro sonido armónico cuya frecuencia fundamental es el doble o la mitad del otro. La importancia de este relación de frecuencias reside en que, dada una nota, las notas que más se le parecen son su octava superior e inferior. De hecho, estas recibirán el mismo nombre.[1].

Cada octava, en el sistema temperado se divide en 12 escalones equiespaciados en un eje de frecuencias logarítmico. Cada uno de estos escalones será un **semitono** En la música occidental, el semitono es la mínima distancia entre dos notas.[2] Existen 11 notas entre un Do_4 y un Do_5:

$$Do\#_4,\ Re_4,\ Re\#_4,\ Mi_4,\ Fa_4,\ Fa\#_4,\ Sol_4,\ Sol\#_4,\ La_4,\ La\#_4,\ B_4$$

Desde un LA_4, cuya frecuencia es 440Hz, hasta un LA_5, cuya frecuencia es el doble, 880Hz, hay 440HZ de diferencia. Sin embargo entre un LA_3 (220Hz) a un LA_4 existe una diferencia de sólo 220Hz. El intervalo entre LA_3 y LA_4 es el mismo que entre LA_4 y LA_5, una octava. De la misma forma, el semitono no define una diferencia de frecuencias, sino, un ratio. La relación de frecuencia correspondiente a un semitono es de $2^{(1/12)}$

La medida del intervalo entre dos notas la realizamos contando el número de semitonos. Generalmente también usamos el término tono [3]. Un intervalo de un tono no es más que el formado por 2 semitonos. La importancia de conocer los intervalos entre distintas notas se debe a que, intervalos con igual relación de frecuencias, son percibidos como iguales, independientemente de en qué frecuencia se den. De esta forma una melodía

[1] Normalmente, el nombre de la nota se acompaña de un número para hacer referencia a su octava. LA_2 y LA_3 representan a la nota LA en distintas octavas. La frecuencia fundamental de LA_3 será justo el doble que la de LA_2

[2] Esto no es así en muchas músicas étnicas en las que se usan instrumentos capaces de producir sonidos con una distancia de un cuarto de tono

[3] No confundir con la acepción de tono en términos acústicos que se refiere a la altura del sonido.

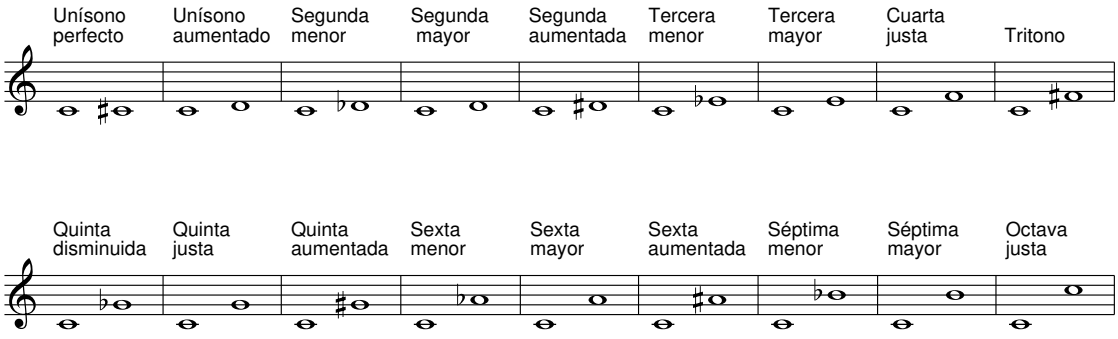


Figura 2.1: Intervalos posibles desde la nota Do

será perfectamente reconocida por un oyente, aunque esta no esté siendo interpretada con las mismas notas que la original, simplemente se tienen que mantener los intervalos entre todas ellas. Gracias a esto, un cantante puede cantar una canción en un registro que sea adecuado a su voz.

En la figura 2.1 podemos ver qué nombre reciben algunos de los intervalos ascendentes que pueden formarse desde la nota Do.

Además del intervalo de octava comentado anteriormente, existen algunos intervalos que son muy utilizados:

- **Unísono**: La diferencia entre las dos notas que forman este intervalo es de 0 semitonos. Para que se dé este caso, las notas no tienen por qué tener obligatoriamente el mismo símbolo. Dos notas con distinto nombre pueden equivaler al mismo sonido. Este es el caso de los llamados **enarmónicos**. Un ejemplo podría ser las notas **Mi#** y **Fa**.

- **Intervalo de quinta**: Este intervalo es fundamental en todos los tipos de música. La relación de frecuencias entre las notas de este intervalo es de 3/2, y corresponde a una distancia de 7 semitonos. Tanto los acordes mayores como los acordes menores contienen este intervalo. Además, aunque la definición estricta de acorde requiere que suenen 3 o más notas, es muy común encontrar en música moderna acompañamientos basados en los llamados **acordes de quintas**. Estos acordes, generalmente interpretados por guitarras eléctricas con distorsión, están compuestos únicamente por un intervalo de quinta.

- **Tritono**: El tritono equivale a un intervalo de cuarta aumentada, formado por 6 semitonos. Este intervalo divide la octava en dos partes iguales. Es un intervalo especialmente disonante. Crea tanta tensión que durante mucho tiempo era conocido como *Diabulus in musica*, llegando a estar incluso prohibido en la Edad Media.

2.1.1. Consonancia y disonancia

La consonancia y disonancia son dos conceptos fundamentales en la armonía, capaces de clasificar distintos intervalos según la tensión que producen. Un intervalo consonante, es percibido como agradable para el oído. Sin embargo, un intervalo disonante provoca cierto rechazo. Mediante la combinación de estas tensiones y relajaciones se crean las obras musicales.

Qué intervalos se consideran consonantes, y cuáles de ellos se consideran disonantes es un tema complejo que, como los cánones de belleza, cambia a lo largo del tiempo.

Actualmente, los intervalos que se consideran consonantes son:

- **3ª Mayor**: Formado por 4 semitonos
- **3ª Menor**: Formado por 3 semitonos
- **4ª Justa**: Formado por 5 semitonos
- **5ª Justa**: Formado por 7 semitonos
- **6ª Mayor**: Formado por 9 semitonos
- **6ª Menor**: Formado por 8 semitonos
- **8ª Justa**: Formado por 12 semitonos

Por el contrario, los intervalos que se consideran como disonantes son:

- **2ª Mayor**: Formado por 2 semitonos
- **2ª Menor**: Formado por 1 semitono
- **4ª Aumentada (Tritono)** : Formado por 6 semitonos
- **7ª Menor**: Formado por 10 semitonos
- **7ª Mayor**: Formado por 11 semitonos

Aunque, como hemos comentado, esta definición sobre qué intervalos son consonantes y cuáles disonantes no está claramente definida, e incluso cambia a lo largo del tiempo, no por ello deja de tener su fundamento físico. En numerosos artículos [Rasch and Plomp, 1999] [Terhardt, 1974] [Cazden, 1945] se estudia cuáles son los factores que afectan a la percepción de un intervalo como consonante o disonante. Estos factores son tanto físicos, como biológicos e incluso culturales.

Desde el punto de vista físico, la consonancia tiene su base en el solapamiento de las series armónicas de las dos notas que forman el intervalo. Cuanto mayor sea este solapamiento, mayor consonancia existirá en el intervalo. El concepto de serie armónica se estudiará más adelante en el siguiente capítulo sobre fundamentos físicos.

En la escala mayor natural, el mayor solapamiento se produce entre los intervalos de cuarta y quinta justa, considerados como consonantes. Uno de cada dos armónicos de la quinta, también está presenta en la tónica, y uno de cada tres armónicos de la cuarta, también lo está en la tónica. De hecho, se suelen considerar al cuarto y quinto grado como **grados tonales**. Al sonar estos, se dice que se consolida la tonalidad.

2.2. Acordes

La armonía consiste en el movimiento de los acordes a lo largo de una obra musical. La definición de acorde que se suele dar es la de *un conjunto de tres o más notas que suenan de forma simultánea y que constituyen una unidad armónica.* Como muchas otras, incluso esta definición da lugar a cierta polémica al considerarse en numerosas ocasiones que un conjunto de dos notas también forman un acorde. Este es el caso de los acordes de quintas formados únicamente por la nota fundamental y su quinta, muy utilizados en música moderna. Según el número de notas que formen el acorde podemos tener **díadas** (formadas por dos notas), **tríadas** (formadas por tres notas), o **tetradas** [4] (formadas por cuatro notas). Sin embargo, en algunos estilos de música, en momentos donde se busque crear mayor tensión, podemos añadir notas, dando lugar a acordes de más de 4 notas.

Los 2 acordes más comunes són las triadas mayores y menores. Estos acordes son la base de la armonía, aunque se le puedan añadir más notas para crear mayores tensiones y dejen, por lo tanto, de ser tríadas. El acorde de tríada mayor está formado por la tónica, su tercera mayor, y su quinta justa. Por ejemplo, el acorde de **Sol Mayor**, estará formado por:

- **Sol**: La tónica. Da nombre al acorde
- **Si**: La tercera mayor
- **Re**: La quinta justa

Si el intervalo de tercera fuera menor en vez de mayor, pasaríamos a tener una tríada menor. Un ejemplo podría ser el acorde de **La menor**, formado por las notas:

- **La**: La tónica. Da nombre al acorde
- **Do**: La tercera menor
- **Mi**: La quinta justa

Otros tipos de tríadas que podemos encontrar son las **aumentadas** y **disminuidas**. Los acordes de tríada aumentados están formados por dos terceras mayores superpuestas, o lo que es lo mismo, desde la tónica, una tercera mayor y una quinta aumentada. Es equivalente a tomar un acorde mayor, y subir un semitono la quinta del acorde. Por el contrario, los acordes disminuidos están formados por dos terceras menores, o lo que es lo mismo, desde la tónica, una tercera menor y una quinta disminuida. Es equivalente a tomar un acorde menor, y bajar un semitono la quinta del acorde.

En la figura 2.2 podemos ver ejemplos de acordes aumentados y disminuidos.

Además, también es común encontrarse los llamados **acordes suspendidos**. La característica que distingue a estos acordes es que no están formados por terceras. También tienen una quinta justa, pero la nota central del acorde en este caso no está a un intervalo

[4] En ocasiones se refiere a estos acordes, erróneamente, como cuatríadas.

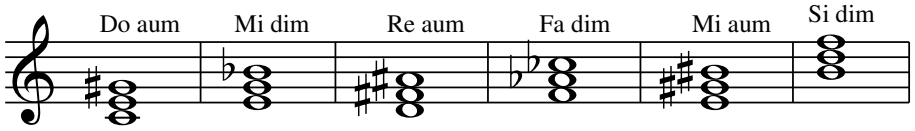


Figura 2.2: Ejemplos de acordes aumentados y disminuidos

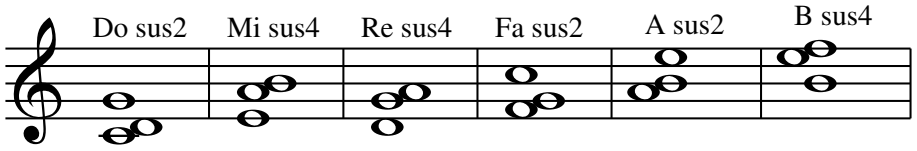


Figura 2.3: Ejemplos de acordes suspendidos

de tercera de la tónica sino a un intervalo de segunda mayor, o de cuarta justa. Este tipo de acordes crean una tensión que se suele resolver en una tríada mayor o menor. En la figura 2.3 podemos ver ejemplos de acordes suspendidos.

2.2.1. Acordes de más de 3 notas

Igual que para construir los acordes de tríadas superoponíamos dos terceres consecutivas a partir de la tónica, para construir los llamados **acordes de séptima**, añadiremos otra tercera más al acorde obteniendo un acorde formado por la nota raíz, su tercera, su quinta y su séptima. El tipo de estos intervalos dará lugar a los distintos acordes de séptima. Estos acordes están muy presentes en todos los estilos de música aportando matices muy interesantes a las tensiones que aportan las tríadas mayores y menores.

Además, en muchas ocasiones, encontramos en estilos como el Jazz acordes que, además de la séptima, incorporan un intervalo de 9ª, 11ª o incluso 13ª. Aunque no los hayamos nombrado hasta ahora, es importante conocer estos intervalos, mayores que la octava, conocidos como **intervalos compuestos**. Por ejemplo, el intervalo de novena desde Do_4 estaría formado por una octava y una segunda, dando lugar a la nota Re_5. Cuando añadimos los intervalos nombrados anteriormente a los acordes de séptima, no

Nombre del acorde	**Tríada**	**Intervalo de séptima**
Séptima mayor	Mayor	Mayor
Séptima menor	Menor	Menor
Séptima dominante	Mayor	Menor
Séptima disminuida	Disminuida	Menor

Tabla 2.1: Acordes de séptima

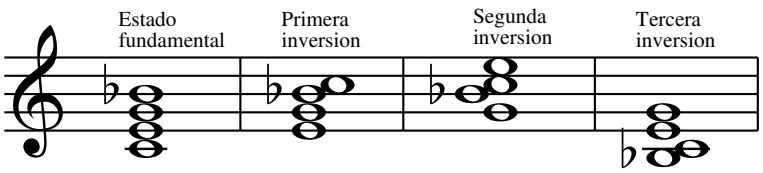


Figura 2.4: Inversiones del acorde de Do séptima dominante

estamos cambiando su función armónica. Su propósito es únicamente el de añadir sutiles matices en la sonoridad del acorde, o incluso crear una segunda línea melódica por debajo de la línea principal. Es por ello por lo que no enfocaremos nuestro esfuerzo en detectar este tipo de acordes y preferimos centrarnos en detectar con eficacia las tríadas mayores y menores y, en la medida de lo posible, las séptimas.

2.2.2. Inversiones

Hasta ahora, se han estudiado los diferentes acordes en su estado fundamental. Esto quiere decir, con la tónica en la voz más grave. Pero los acordes no siempre aparecen así. Es por ello por lo que hablamos de inversiones. Dependiendo de cual sea la nota del acorde que está en la voz más grave podemos tener el acorde en primera, segunda o tercera inversión. En la figura 2.4 podemos ver las distintas inversiones del acorde de séptima dominante de Do.

2.3. Escalas

A partir de los doce sonidos en los que se divide una octava, podemos hacer distintas agrupaciones de ellos formando nuevos subsistemas musicales. Una escala no es más que una secuencia de varios sonidos ordenados por altura (según su frecuencia fundamental) que sirven al autor como base para crear las diferentes melodías. La escala es la paleta de colores del pintor, que contiene todos los posibles colores que usará en su obra. El autor combinará los sonidos de una o varias escalas para crear su obra. Al igual que pasaba con los intervalos, quien determina las sensaciones que provoca una escala no son las frecuencias absolutas que suenan, sino las relaciones entre las distintas notas de la escala. Depende de la cantidad de notas, y de los intervalos que hay entre ellas el nombre que tome la escala. Podemos construir una escala mayor comenzando sobre cualquier nota, siempre y cuando mantengamos el mismo orden de tonos y semitonos.

Aunque existe una gran cantidad de escalas, la mayor parte de la música está basada en las siguientes escalas:

- **Escala mayor**: La escala mayor está compuesta por 7 notas y está definida por la siguiente secuencia de intervalos:

  Tono – Tono – Semitono – Tono – Tono – Tono – Semitono.

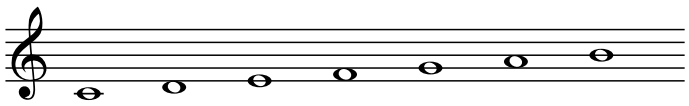


Figura 2.5: Escala de Do mayor

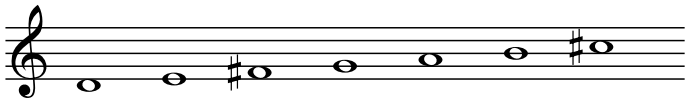


Figura 2.6: Escala de Re mayor

La escala de Do Mayor, es la representación más típica de esta escala, pues no necesita ninguna alteración porque los intervalos entre las notas ya cumplen la secuencia que debe tener una escala mayor. Mediante sostenidos (#) o bemoles (b), que son capaces de aumentar o disminuir respectivamente en un semitono la altura a la que corresponde el nombre de una nota, podemos obtener el resto de escalas mayores. Habitualmente las escalas mayores se suelen percibir como escalas alegres y con gran cantidad de energía. En las figuras 2.5 y 2.6 vemos dos ejemplos de escalas mayores.

- **Escala menor**: La escala menor se obtiene si tomamos las notas sin alterar desde La. Está compuesta por 7 notas con la siguiente secuencia de intervalos:

  Tono – Semitono – Tono – Tono – Semitono – Tono – Tono.

  La escala menor suele considerarse que transmite sensaciones más tristes. En las figuras 2.7 y 2.8 vemos dos ejemplos de escalas menores.

- **Escala menor armónica**: La escala menor armónica es muy similar a la escala menor. Se construye elevando la séptima nota de una escala menor natural un semitono. Esto aporta a la última nota de la escala una tensión especial que se resolverá al ir a la tónica. Un ejemplo de esta escala lo podemos ver en la figura 2.9

- **Escala menor melódica**: La escala menor melódica eleva un semitono los grados VI y VII respecto a la escala menor natural. Sin embargo, esta escala tiene una

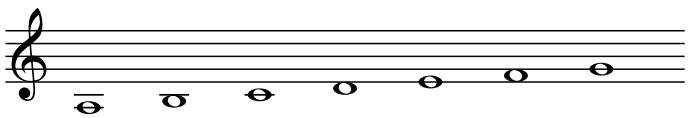


Figura 2.7: Escala de La menor

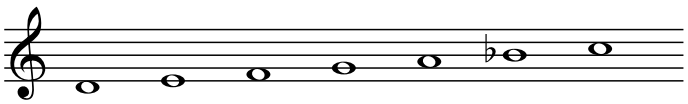


Figura 2.8: Escala de Re menor

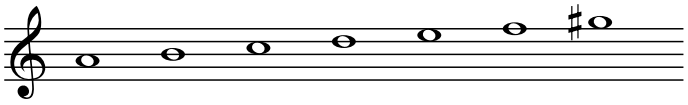


Figura 2.9: Escala de La menor armónica

propiedad que no encontramos en otras escalas y es que sólo alteramos estos semitonos cuando interpretamos en sentido ascendente las notas de la escala. Cuando el movimiento es descendente, se tocan las notas pertenecientes a la escala menor natural. Podemos ver un ejemplo en la figura 2.10

- **Escala pentatónica mayor**: A partir de la escala mayor, si quitamos los intervalos de semitono, nos queda una escala de cinco notas llamada escala pentatónica mayor. Las escalas pentatónicas son ampliamente utilizadas en estilos musicales modernos, además de muy utilizadas para la improvisación en contextos de jazz, pop y rock actuales, debido a que funcionan bien sobre varios acordes diatónicos de la misma tonalidad En la figura 2.11 vemos las 5 notas de la escala pentatónica mayor de Do.
- **Escala pentatónica menor**: Como en la mayor, para obtener la escala pentatónica menor, partimos de la escala menor natural y quitamos los semitonos dando lugar a una escala de 5 notas. En la figura 2.12 vemos un ejemplo de esta escala.

2.4. Tonalidad

Relacionado con el concepto de escala, existe también el de tonalidad. La tonalidad puede entenderse como la organización jerárquica de un conjunto de melodías y acordes alrededor de una nota principal que se denomina **tónica**. Es un concepto que no consiste únicamente en utilizar las notas de una escala concreta sino que e establece una jerarquía

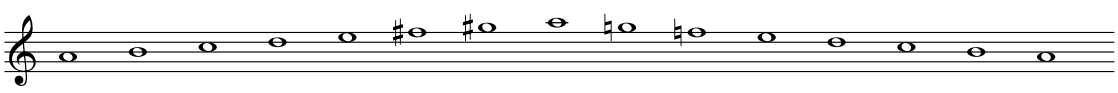


Figura 2.10: Escala de La menor melódica

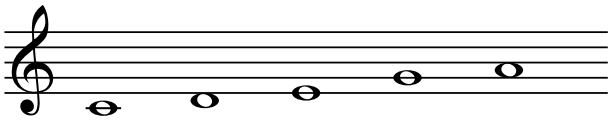


Figura 2.11: Escala pentatónica mayor de Do

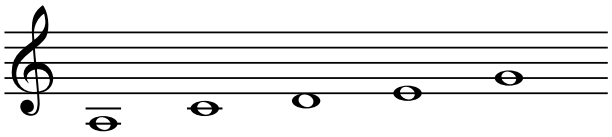


Figura 2.12: Escala pentatónica menor de La

donde la tónica es el centro del conjunto de sonidos que forma la escala asociada a dicha tonalidad.

La sensación de reposo se encuentra precisamente al interpretar la nota o acorde principal de la tonalidad, la nota que da nombre a esta. Además, al igual que podemos decirlo de las escalas, podemos tener tonalidades mayores o menores, dependiendo de si la escala principal usada es mayor o menor. En muchas canciones se produce un fenómeno denominado **modulación** por el cual se modifica la tonalidad de la obra a mitad de esta. Esto requiere de un proceso concreto para que el oyente sea capaz de reconocer la nueva tónica como nota principal que le proporcione reposo. Precisamente, acordes como el de séptima dominante del quinto grado de la nueva tonalidad permiten que pueda asentarse esta, creando una gran tensión que se resuelve al sonar el acorde del nuevo primer grado.

2.4.1. Armonía funcional

De la misma forma que pasa con las escalas, los acordes pueden transmitir distintas emociones según sean mayores o menores, siendo los mayores más relacionados con la *alegría* y los menores más relacionados con la *tristeza*

Además, en la sección anterior, hemos comenzado a ver que ciertos acordes tienen una función en el desarrollo armónico de la obra. Hemos comentado, que por ejemplo, el acorde correspondiente al quinto grado de la tonalidad crea unas tensiones que al resolver en el primer grado, la tónica, son capaces de asentar la tonalidad. Esto es independiente de cuál sea la tónica, lo importante es la posición en la escala de este acorde, que le otorga este comportamiento de **dominante**. Este es el objetivo de la armonía funcional, representar los acordes y estudiar sus relaciones independientemente de la tonalidad seleccionada. Los acordes pertenecientes a una escala, son representados en base a su posición en ella. Usamos números romanos para identificarlos y referirnos a su posición en la escala, con respecto a la tónica.

Como ejemplo, tomaremos la escala de Do Mayor, que tiene las siguientes notas: Do, Re, Mi, Fa, Sol, La, Si. Tomando cada una de estas notas como raíz, podemos crear 7

Grado	Notas del acorde	Nombre del acorde
I	Do - Mi - Sol	Do Mayor
II	Re - Fa - La	Re Menor
IIIm	Mi - Sol - Si	Mi menor
IV	Fa - La - Do	Fa Mayor
V	Sol - Si - Re	Sol Mayor
VIm	La - Do - Mi	La Menor
VII°	Si - Re - Fa	Si disminuido

Tabla 2.2: Acordes creados a partir de la escala de Do Mayor

tríadas a partir de la escala de Do Mayor, usando únicamente las notas de esa escala. En la tabla 2.4.1 podemos ver cuáles son los acordes que se forman a partir de Do Mayor.

Como vemos, en el caso de los grados que dan lugar a acordes menores, lo especificamos, junto a los números romanos, con una m. Si estuviéramos en otra escala, los nombres de los acordes de cada uno de los grados, como es obvio, cambiarían. Sin embargo, el acorde del primer grado seguiría siendo mayor, el acorde del segundo grado seguiría siendo menor, etc. Por lo tanto, lo que verdaderamente nos interesa, desde el punto de vista de la armonía funcional es el grado de la nota que está sonando, no su nombre.

Podemos distinguir tres funciones principales de los acordes. Estas son: **tónica**, **subdominante** y **dominante**. Los nombres de tónica, subdominante y dominante se corresponden, respectivamente, con los grados primero, cuarto y quinto. Sin embargo, existen otros grados de la escala mayor que pueden realizar esta funciones.

- **IIIm y VIm**: Estos acordes, pueden realizar la función de tónica, aunque no pueden llegar a tener el mismo grado de estabilidad que el primer grado.
- **IIm**: El acorde menor formado a partir del segundo grado es capaz de realizar la función de subdominante. Los acordes de subdominante, tienen a ir, con la misma fuerza, bien a la subdominante o bien a la a dominante.
- **VII** Se trata de un acorde disminuido. Su función tonal es la de dominante pero es el acorde más débil e inestable de la tonalidad. Los acordes con función de dominante siempre buscan ir hacia la estabilidad de la tónica.

La información que nos aporta la función armónica de un acorde, presenta un gran valor a la hora de realizar la estimación de acordes. En caso de duda entre varios posibles acordes en un momento concreto, la información de la tonalidad, el acorde anterior y el acorde posterior tiene un gran valor para la elección del acorde adecuado.

2.5. Notación musical

Igual que la escritura jugó un papel fundamental en la evolución del ser humano, permitiendo conservar y transmitir mensajes e ideas, la notación musical permitió transmitir

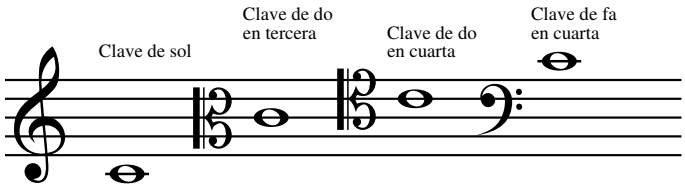


Figura 2.13: Do4 representado en las claves más comunes

y conservar las obras de los compositores a lo largo del tiempo. Antes de existir sistemas de notación, la música sólo podía ser transmitida de forma oral. Tras las creación de estos sistemas, las obras musicales no sólo podrían conservarse durante mucho más tiempo y llegar a lugares mucho más lejanos, sino que ahora estas podrían ser interpretadas por músicos que podrían no haber escuchado nunca esa obra. Los primeros indicios de notación musical datan de hace más de 4000 años. Sin embargo, la notación usada actualmente posee poco más de 400 años. El documento que contiene la transcripción de una obra musical es conocido como **partitura**

2.5.1. Pentagrama

El elemento principal de una partitura, sobra el que escribimos las notas musicales, es el **pentagrama**. Un pentagrama no es más que un conjunto de cinco líneas paralelas, horizontales y equidistantes sobre las cuales dibujamos los símbolos con los que transcribimos la pieza musical. Las notas musicales pueden representarse sobre las propias líneas del pentagrama, o sobre los espacios que existen entre ellas. Dependiendo de sobre qué línea o espacio coloquemos una nota tendrá un sonido más grave (parte inferior del pentagrama) o más agudo (parte superior del pentagrama). Los símbolos que representan las notas musicales son colocados de izquierda a derecha del pentagrama según el orden aparición en la pieza musical. Para representar dos o más sonidos que deben sonar simultáneamente los colocaremos sobre la misma vertical.

2.5.2. Representación de la altura

El primer elemento que vemos al principio de un pentagrama es la **clave**. De la clave dependerá el nombre que tomen las notas de cada línea o espacio. La clave más utilizada habitualmente es la clave se Sol. Esta clave sitúa la nota SOL_4 [5] en la segunda línea del pentagrama (empezando desde abajo). Sin embargo también es común encontrarnos la clave de fa en cuarta o las clave de do en tercera o cuarta.

En una partitura podemos encontrar también unos símbolos que modifican la nota a la que acompañan. Son las llamadas **alteraciones**. Las posibles alteraciones que podemos

[5] Esto es así según el índice acústico científico, en el que nos basamos durante todo el proyecto. Sin embargo, esa misma nota sería SOL_3 en el sistema francobelga usado en Bélgica y en algunas regiones de Francia y España

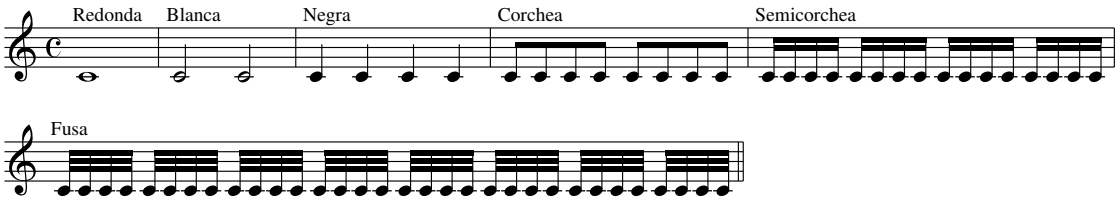


Figura 2.14: Duraciones de los distintos símbolos

encontrar son:

- **Sosteido**: El sostenido eleva, en medio semitono, la nota a la que acompaña. Por lo tanto, si está al lado de un LA_4, la nota pasará a ser $LA\#_4$, un semitono por encima de LA_4.
- **Bemol**: El bemol hace lo contrario al sostenido, baja un semitono la nota a la que acompaña. Si está al lado de un RE_5, la nota pasará a ser REb_5, un semitono por debajo de RE_5.
- **Doble sostenido**: El doble sostenido eleva 2 semitonos la nota a la que acompaña.
- **Doble bemol**: El doble bemol baja 2 semitonos la nota a la que acompaña.
- **Becuadro**: Cuando en un compás aparece una nota con una alteración de la nombradas anteriormente, cada vez que vuelva a aparecer esa alteración durante ese compás, automáticamente deberá ser alterada aunque no esté explícitamente indicado. El becuadro elimina este comportamiento indicando que la nota a la que acompaña no debe ser alterada. Además, el becuadro también elimina restricciones puestas por la armadura.

A continuación de la clave, es común ver un conjunto de símbolos de sostenido o bemol, lo que conocemos como **armadura**. La armadura contiene una serie de alteraciones que deberán ser aplicadas a las notas que correspondan durante todo el desarrollo de la obra. Estas alteraciones dependerán de la **tonalidad**

2.5.3. Representación de la duración

Además de la altura, en una partitura se representa también la duración de las notas. Mediante los siguientes símbolos podemos definir la duración de una nota. En la figura 2.14 vemos los diferentes símbolos que representan la duración de las notas y cuantas de ellas caben en un compás de 4 por 4.

Como ejemplo, la blanca dura 2 tiempos. El **tiempo** es una medida que no tiene una duración concreta sino que en cada obra se indica cuál será su velocidad, en beats por minuto. Por ejemplo, si la obra tiene un **tempo** de 60 beats por minuto, significa que cada tiempo durará un segundo, y que por lo tanto una blanca durará 2 segundos.

Además, los tiempos pueden ser agrupados en lo que conocemos como **compases**. El número de tiempos por compás es una medida que permite incluso reconocer un estilo de música concreto. Por ejemplo, un vals es claramente reconocible por usar un compás de tres tiempos. Esto se debe a que la posición de una nota en el compás, determina en cierto modo su importancia dentro de el. En el compás de 4 por 4, los acentos están en los tiempos primero y tercero. Esto significa que las notas que caigan sobre esos tiempos tendrán un mayor énfasis que las que caigan en los otros dos tiempos.

El tipo de compás que tendrá una obra se definen al principio de la partitura, junto a la clave, mediante dos números colocados uno encima del otro [6]. que representan la nota que define la duración de un **pulso** (número inferior) y cuántos de esos pulsos hay en cada compás (número superior).

Además, podemos encontrar un pequeño punto a la derecha de una figura, denominado **puntillo**, que incrementa la duración de la figura un 50 %. Por ejemplo, si encontramos el puntillo al lado de una redonda (que por defecto duraría 4 tiempos), pasará a durar 6.

Al igual que tenemos símbolos para representar la duración de las notas, también tenemos símbolos para representar la duración de los silencios.

2.5.4. Representación de acordes

De la misma forma que representamos melodías sobre una partitura, también podemos representar acordes. Todas las notas que estén alineadas verticalmente en un pentagrama se deben interpretar a la vez. De esta forma, para representar un acorde simplemente tendremos que anotar en el pentagrama, sobre la misma vertical, cada una de las notas que lo forman.

Cifrados

Como respuesta a la necesidad de representar los acordes de una forma más fácil de leer, surgen los cifrados. En los cifrados se muestran los acordes de una obra mediante combinaciones de números y letras que indican la raíz, el tipo y, en ocasiones, la inversión del acorde que se debe ejecutar. Por lo tanto este tipo de notación deja muchos aspectos al criterio del intérprete.

Como ejemplo, el acorde de Do Menor puede ser representado en un cifrado escribiendo **DOm**, o en cifrado anglosajón (del que se hablará a continuación), **Cm**.

Este tipo de notación es muy común en jazz y estilos de música contemporáneos aunque ya en el barroco se podían encontrar en numerosas ocasiones.

Cifrado anglosajón

Aunque estemos acostumbrados a los nombres de las notas en español[7], **Do Re Mi Fa Sol La Si**, en muchas ocasiones nos encontramos el llamado cifrado anglosajón. Este cifrado utiliza las 7 primeras letras del alfabeto para dar nombre a las notas musicales.

[6] En el caso del compás de 4 por 4 se puede representar simplemente mediante una letra C

[7] Realmente, existen varios idiomas que comparten estos mismos nombres para las notas musicales

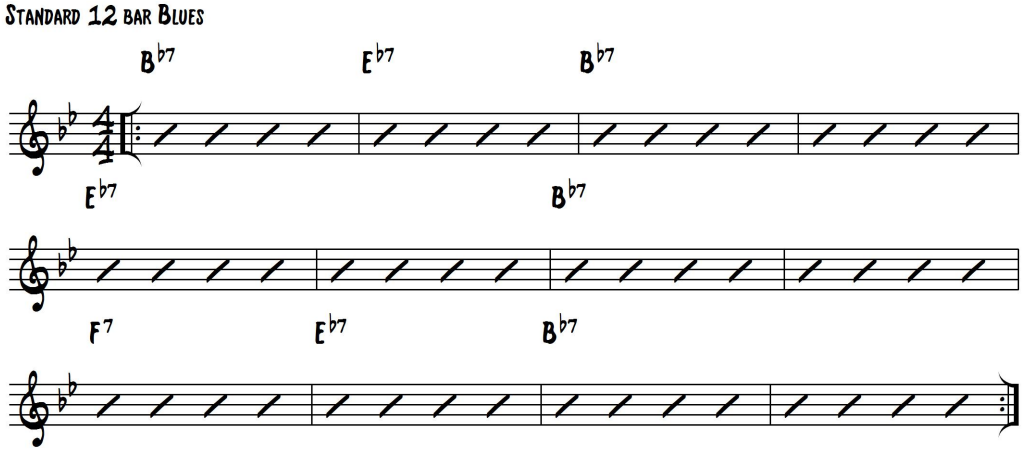


Figura 2.15: Ejemplo de cifrado americano

- **A**: Representa la nota La, o el acorde de La Mayor
- **B**: Representa la nota Si, o el acorde de Si Mayor
- **C**: Representa la nota Do, o el acorde de Do Mayor
- **D**: Representa la nota Re, o el acorde de Re Mayor
- **E**: Representa la nota Mi, o el acorde de Mi Mayor
- **F**: Representa la nota Fa, o el acorde de Fa Mayor
- **G**: Representa la nota Sol, o el acorde de Sol Mayor

Notación del proyecto

Puesto que existen muchas posibles representaciones posibles para los acordes, hay que elegir un criterio claro que será utilizado a la hora de representar los acordes en nuestro proyecto. Es por ello, por lo que en [Harte et al., 2005] se propone un criterio a seguir para la representación de acordes en proyectos de MIR. El fin de este trabajo era crear un criterio único, libre de ambigüedades, que fuera usado en todo este tipo de proyectos. El formato adecuado era el siguiente:

root : (degree1, degree2...) / bass

Comenzamos definiendo la raíz del acorde y ponemos después el intervalo desde la raíz a cada una de las notas del acorde. Añadimos un **b** a estos grados, si queremos indicar que el intervalo es menor, o un # si el intervalo es aumentado. Al final, si el acorde no está en estado fundamental, añadimos el intervalo desde la raíz, a la nota que estará

Tipo de acorde	Lista de componentes	Nombre simplificado
Triada Mayor	(3,5)	maj
Triada Menor	(b3,5)	min
Triada Aumentada	(3,#5)	aug
Triada Disminuida	(b3,b5)	dim
Séptima mayor	(3,5,7)	maj7
Séptima menor	(b3,5,b7)	min7
Séptima dominante	(3,5,b7)	7
Séptima disminuida	(b3,b5,bb7)	dim7
Séptima semidisminuida	(b3,b5,b7)	hdim7
Sexta mayor	(3,5,6)	maj6
Sexta menor	(b3,5,6)	min6
Novena	(3,5,b7,9)	9
Novena mayor	(3,5,7,9)	maj9
Novena menor	(b3,5,b7,9)	min9
Cuarta suspendida	(4,5)	sus4

Tabla 2.3: Acordes creados a partir de la escala de Do Mayor

en la posición más grave en el acorde, definiendo así su inversión. Sin embargo, no es esta la notación más usada, sino que es la simplificación de esta notación la que mayor aceptación ha tenido. El formato simplificado permite sustituir las listas de intervalos de los acordes más comunes por una palabra (shorthand) que los represente. La forma de representar el acorde es la siguiente: *root : shorthand(extra-degrees) / bass*.

En la tabla 2.5.4 se pueden ver las equivalencias entre estas notaciones.

3 Fundamentos físicos

En este capítulo se estudiará el sonido desde el punto de vista de la física y las matemáticas. Saber en qué consiste el sonido, llegando hasta el nivel de las partículas, será de gran utilidad a la hora de abordar el problema de la estimación automática de acordes con una perspectiva más amplia y siendo capaces de detectar problemas que, de otra forma, no sería posible.

3.1. Física del sonido

La vibración de algunos elementos se transmite a través de un medio elástico, formando una onda que, dependiendo de la frecuencia, puede hacer vibrar a ciertas estructuras anatómicas provocando respuestas neurológicas y psicológicas implicadas en la percepción del sonido. El sonido se transmite mediante ondas longitudinales, esto quiere decir que la vibración de la onda es paralela a su dirección de propagación. A diferencia de la luz, el sonido siempre requiere un medio para transmitirse, y nunca podrá transmitirse por el vacío.

Si medimos en un punto fijo la presión atmosférica, veremos que, cuando se está produciendo un sonido, esta aumenta y disminuye de forma periódica, como podemos ver en la figura 3.1.

A nivel de partículas, lo que está sucediendo es que se están produciendo una serie de compresiones y rarefacciones del aire. La presión atmosférica se mide en pascales y se sitúa en torno a los 100 000 Pa al nivel del mar. Sin embargo, el aumento de la presión atmosférica debido al sonido es mínimo, situándose entre 20 micropascales y 20 pascales. Esto es lo que definimos como presión sonora.

Como se puede observar, existe una gran diferencia entre el mínimo y el máximo valor de presión sonora que podemos percibir los humanos. Es por ello, y por cómo detecta esas presiones nuestro oído, por lo que es común hablar de **Nivel de Presión Sonora**. El nivel de presión sonora se define mediante la siguiente fórmula:

$$L_p = 20 \cdot log(\frac{P_1}{P_0}) \tag{3.1}$$

donde:

- P_1: Presión sonora eficaz(RMS)
- P_0: Presión de referencia: $20\mu Pa$

Para que los humanos podamos percibir un sonido, la frecuencia de la onda este tiene que estar entre 20 y 20000 Hz. Los sonidos con frecuencias superiores a los 20 kHz son

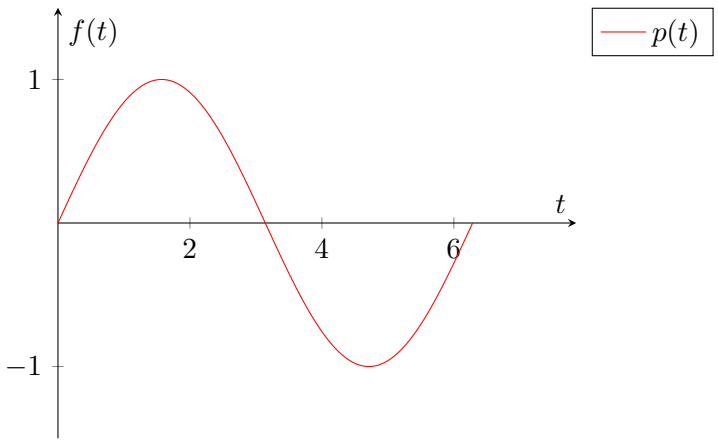


Figura 3.1: Presión en función del tiempo para un punto determinado

conocidos como ultrasonidos mientras que los sonidos con frecuencias inferiores a los 20 Hz son conocidos como infrasonidos.

Si se analizan las ondas sonoras podemos ver que, generalmente, estas son combinaciones de varias ondas sinusoidales. Una onda sinusoidal es el patrón de vibración más simple. Este tipo de onda se describe mediante la siguiente ecuación:

$$y(t) = A \cdot sin(2 \cdot \pi \cdot f \cdot t + \phi) \tag{3.2}$$

La fórmula anterior define la onda mediante tres parámetros:

- A: **Amplitud**: Es el máximo valor de la onda en cada ciclo, en valor absoluto.
- f: **Frecuencia**: Determina el número de ciclos por segundo que realiza la onda. El valor inverso de la frecuencia es el **Periodo (T)**: El periodo mide el tiempo entre dos puntos equivalentes de una onda.
- ϕ**: Fase**: La fase mide la porción del ciclo que ha transcurrido en cada instante. Añadiendo una fase a la fórmula de la onda sinusoidal desplazaremos el origen hasta el punto que deseemos. Los humanos no somos capaces de percibir la fase de una onda sinusoidal. Sin embargo, es un elemento que se tiene siempre en cuenta, pues puede provocar problemas a la hora de combinar varias ondas sinusoidales. El resultado de la combinación de estas dependerá de la fase de cada uno, llegando a ser incluso nulo si existe un desfase entre todas ellas que haga que la suma sea 0. Esto se puede ver en la figura 3.2.

Como se analizará más adelante, en muchas ocasiones, existe una onda principal que provoca la periodicidad de la onda, a la que se suman, con menor aportación, otras ondas sinusoidales con frecuencias múltiplos de la frecuencia de la onda principal, conocidas como frecuencias parciales.

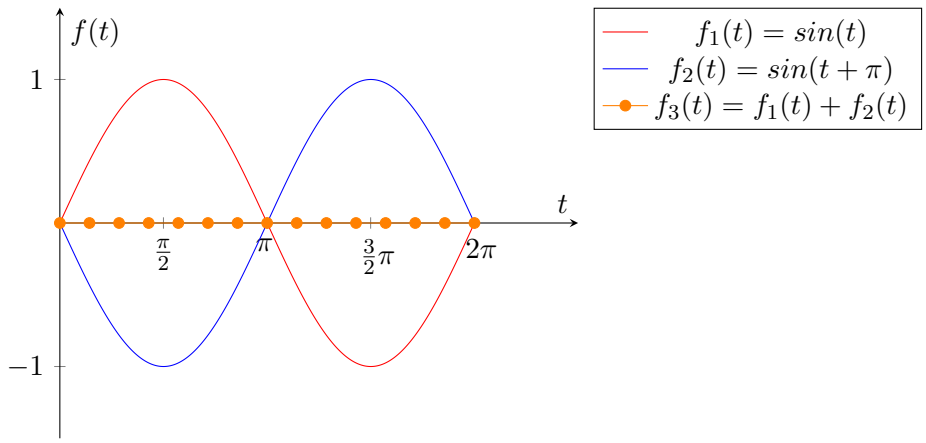


Figura 3.2: Resultado de la combinación de dos ondas sinusoidales con una diferencia de fase de π

3.2. Transformada Discreta de Fourier

La información en bruto de las muestras de un audio no resulta de gran utilidad a la hora de extraer característica de éste, pues no existe una relación directa con el contenido armónico existente. Sin embargo, hay evidencias de que el oído humano, realiza una transformación del dominio del tiempo al dominio de la frecuencia, siendo más sensible a la información frecuencial que a la información de fase. [Deutsch, 2013]

Todas las ondas sonoras pueden describirse como la combinación de varias ondas sinusoidales. El Teorema de Fourier, aplicado al tratamiento de señal, nos dice que toda función periódica de período P puede descomponerse en una suma de sinusoides armónicas, de amplitudes y fases adecuadas, cuyo primer armónico posea período P. En la figura 3.3 podemos ver un ejemplo de cómo una señal compleja se puede descomponer como la suma de varias señales sinusoidales simples.

Además, las frecuencias de estos armónicos son múltiplos enteros de una frecuencia conocida como frecuencia fundamental.

Este campo de estudio es conocido como **análisis de Fourier**. Sin embargo, dependiendo de la naturaleza de la onda que se estudia las herramientas utilizadas son distintas.

- **Señales periódicas y continuas** Ejemplos de estas señales son por ejemplo las ondas sinusoidales o las ondas cuadradas. Para este tipo de señales se usan las **Series de Fourier**.

$$s_N(x) = \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \sin(\frac{2\pi n x}{P} + \phi_n) \tag{3.3}$$

  Asumimos que $s_N(x)$ es una señal periódica de periodo P.

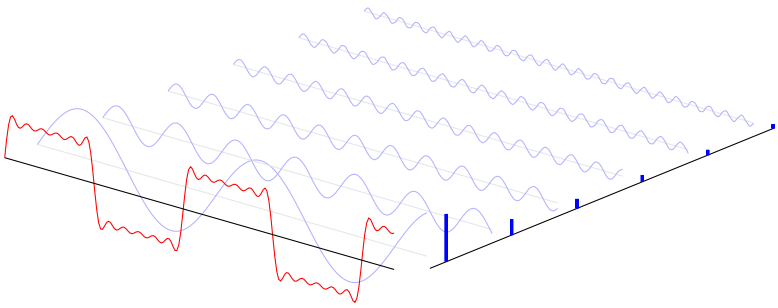


Figura 3.3: Descomposición de una señal compleja en varias señales sinusoidales

- **Señales periódicas y discretas** Este tipo de señales se repiten continuamente, pero solo están definidas en una serie de puntos concretos. La herramienta utilizada para estas es la **Transformada Discreta de Fourier (DFT)**

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1 \tag{3.4}$$

- **Señales no periódicas y continuas** Ejemplos de estas señales son las señales exponenciales o las curvas Gaussianas. La herramienta utilizada es la **Transformada de Fourier (TF)**.

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx, \tag{3.5}$$

  Se define para cualquier número real ξ.

- **Señales no periódicas y discretas** Estas señales solo están definidas en una serie de puntos discretos. La herramienta utilizada es la **Transformada de Fourier de Tiempo Discreto (DTFT)**.

$$X_{2\pi}(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-i\omega n} \tag{3.6}$$

Cuando tratamos con señales digitalizadas, la herramienta que tendremos que usar es la **Transformada Discreta de Fourier**. La DFT será capaz de descomponer las señales en una serie de ondas sinusoidales con distintas frecuencias y fases, cada una de ellas con un peso concreto. Por lo tanto, el resultado de la DFT será la descomposición en frecuencia de la onda, obteniendo una representación de la señal en el dominio de la frecuencia. En ella, podremos ver qué frecuencias destacan en cada sonido. No debemos olvidar que nuestro propósito final es ser capaces de detectar qué acordes suenan en una serie de muestras de audio. Como sabemos, cada nota tiene una frecuencia única

y por lo tanto, una primera aproximación a la detección de acordes sería analizar la respuesta de la DFT para ver qué frecuencias tienen más peso, y con qué nota musical se corresponderían, y a partir de ahí, ver si ese patrón se corresponde con un patrón de un acorde concreto.

La representación de una señal en el dominio frecuencial es capaz de revelar una gran cantidad de información acerca de la estructura interna de un sonido. Además, es una representación mucho más similar a nuestra percepción pues nuestro oído percibe el sonido gracias a que las variaciones de presión se producen a unas frecuencias concretas.

3.3. Parciales y formantes: Timbre

Cuando se excita un sistema resonante, el sistema comienza a vibrar. Dentro de esa vibración podemos encontrar varias frecuencias. La principal, será la frecuencia de resonancia. Sin embargo, podremos encontrar varias frecuencias mayores que esta.

Los instrumentos afinados suelen estar basados en un oscilador armónico, como una cuerda o una columna de aire. Estos instrumentos generan un sonido conocido como **sonido armónico**. El espectro de la señal de estos instrumentos está formado por varias frecuencias, denominadas frecuencias parciales o simplemente parciales.

En las señales periódicas producidas por los osciladores armónicos existe una frecuencia parcial que tiene gran relevancia, la **frecuencia fundamental**. Esta frecuencia es la inversa del periodo fundamental, tiempo en el que se repite el ciclo principal de la onda. El resto de parciales serán conocidos como **armónicos** Además, todos los armónicos son múltiplos enteros de esta frecuencia fundamental. Por lo tanto, el máximo común divisor de todos los parciales será la frecuencia fundamental. La frecuencia fundamental está muy ligada a nuestra percepción, pues será la frecuencia que detectemos en un sonido concreto. Sin embargo, esto sólo podrá ser así cuando la frecuencia fundamental esté por encima del límite inferior de frecuencia del oído humano, 20 Hz.

Existe un fenómeno curioso relacionado con la frecuencia fundamental, conocido como **frecuencia ausente**. Existen ocasiones, en las que todas las frecuencias parciales son múltiplos de una frecuencia, pero esta no aparece en el espectro de la onda. En estos casos, aunque no exista este parcial, el oído sí la percibirá debido al la interferencia producida entre parciales vecinos.

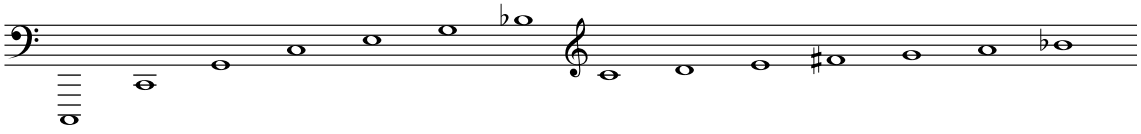


Figura 3.4: Serie armónica

La serie armónica es la representación sobre el pentagrama de todos los armónicos que genera un oscilador armónico ideal. En la figura 3.4 podemos ver, cuando se interpreta una nota Do_1, cuáles son los armónicos que oiremos junto con esa nota. Como propuso Hermann von Helmholtz, existe una clara correlación entre los intervalos que comparten varios armónicos y los intervalos que percibimos como consonantes. [Helmholtz and Ellis, 1886]. Además, fijándonos en la serie anterior,podemos ver que hay notas que se repiten en diversas ocasiones:

- **Do**: Se repite en 4 ocasiones
- **Sol**: Se repite en 3 ocasiones
- **Mi**: Se repite en 2 ocasiones
- **Si bemol**: Se repite en 2 ocasiones

Se puede concluir, por lo tanto, que al sonar una nota con todos sus armónicos, en cierta medida está sonando su acorde mayor, o si tenemos en cuenta el Si bemol, su acorde de séptima dominante. Podría afirmarse que el acorde mayor es intrínseco a la serie armónica

Cuando escuchamos sonidos armónicos, percibimos claramente la frecuencia fundamental, y estos armónicos únicamente *colorean* el sonido. Este fenómeno de *coloreo* del sonido que realizan los armónicos, es lo que conocemos como **timbre**. El timbre es la propiedad que nos permite distinguir dos instrumentos distintos que interpretan la misma nota. Aunque la frecuencia fundamental sea la misma, el peso de los distintos parciales no lo es, permitiendo al oído diferenciar ambos sonidos. En la figura 3.5 podemos ver ejemplos de 2 instrumentos distintos interpretando la misma nota. Al tener diferente energía en cada armónico su forma de onda será totalmente diferente.

Si nos fijamos ahora en la figura 3.6, que representa los espectros de los dos sonidos anteriores, vemos que ambos tienen un parcial a 440 que destaca sobre todos los demás, pero además tienen más armónicos.

Además de los parciales, existen elevaciones en el espectro, formadas por varios parciales, conocidas como **formantes**. Los formantes juegan un papel muy importante en el reconocimiento de voz, pues permiten distinguir los sonidos vocálicos, que se caracterizan por tener estos formantes a unas frecuencias concretas. Por ejemplo, en la figura 3.7, vemos los formantes que produce la vocal i.

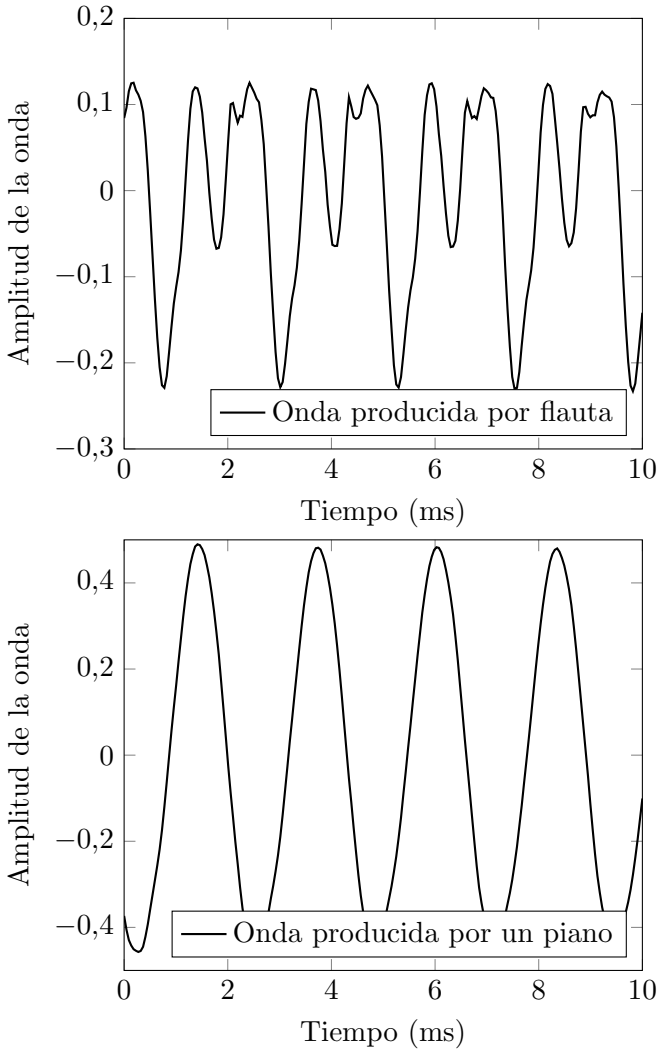


Figura 3.5: Nota A4 interpretada por una flauta travesera y por un piano

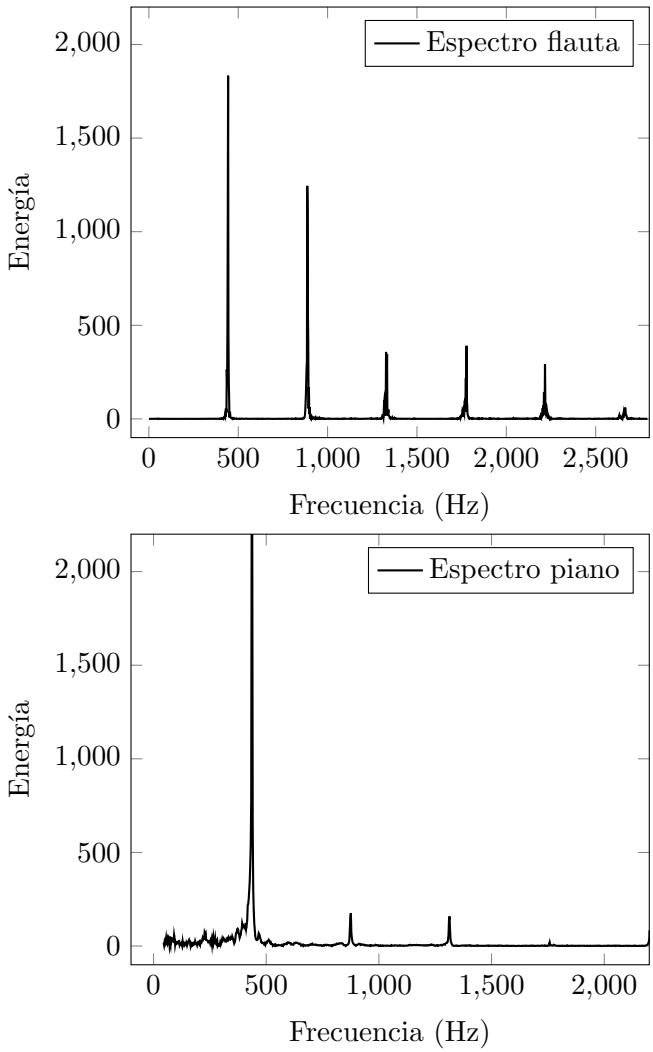


Figura 3.6: Espectros de la nota A4 interpretada por una flauta travesera y por un piano

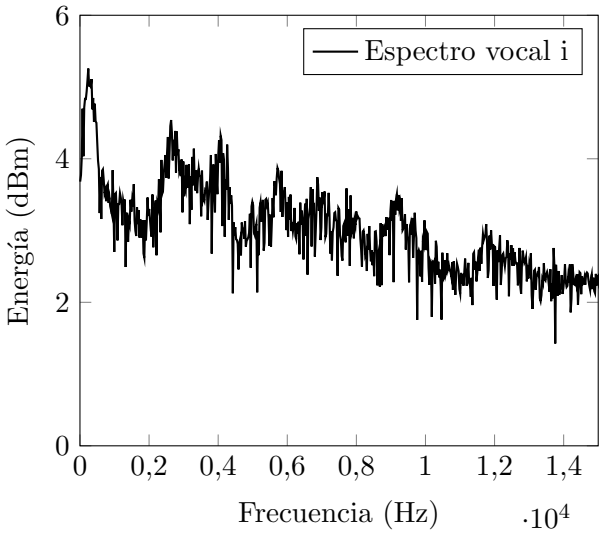


Figura 3.7: Espectro producido por una voz pronunciando la vocal i

3.4. Ruido

La definición de ruido depende en gran medida del contexto en el que se realice. El ruido no tiene por qué presentar ninguna característica física concreta, sino que se llama ruido a cualquier sonido no deseado. Sin embargo, habitualmente este sonido suele ser un sonido inarmónico con parciales en casi todas las frecuencias del espectro.

Toda señal no generada artificialmente, siempre tendrá una cierta cantidad de ruido. Por lo tanto, ya no encontraremos los espectros armónicos perfectos que hemos analizado. Tendremos un espectro con energía en todas las frecuencias, resultado de la suma de la señal que se desea captar y el ruido. Este es uno de los primeros problemas que encontramos al querer analizar las notas musicales presentes en un audio, no podemos saber si la presencia de algunas frecuencias en el espectro se deben a armónicos de la señal, o al ruido. Separar señal y ruido es una tarea difícil, y nunca se puede obtener un resultado totalmente perfecto, por lo que habrá que lidiar siempre con él. En la figura 3.8 vemos la diferencia del espectro de una onda sinusoidal con una frecuencia de 500 Hz y la misma onda a la que se le ha añadido ruido con energía en todas las frecuencias.

3.5. Evolución temporal del espectro

La DFT nos proporciona el espectro total de toda la señal. Sin embargo, no es capaz de darnos la evolución en el tiempo de este espectro. La solución lógica para este problema sería ir tomando progresivamente conjuntos de muestras y aplicar la DFT únicamente a esas muestras. Esto es equivalente a aplicar la DFT sobre sucesivas ventanas que

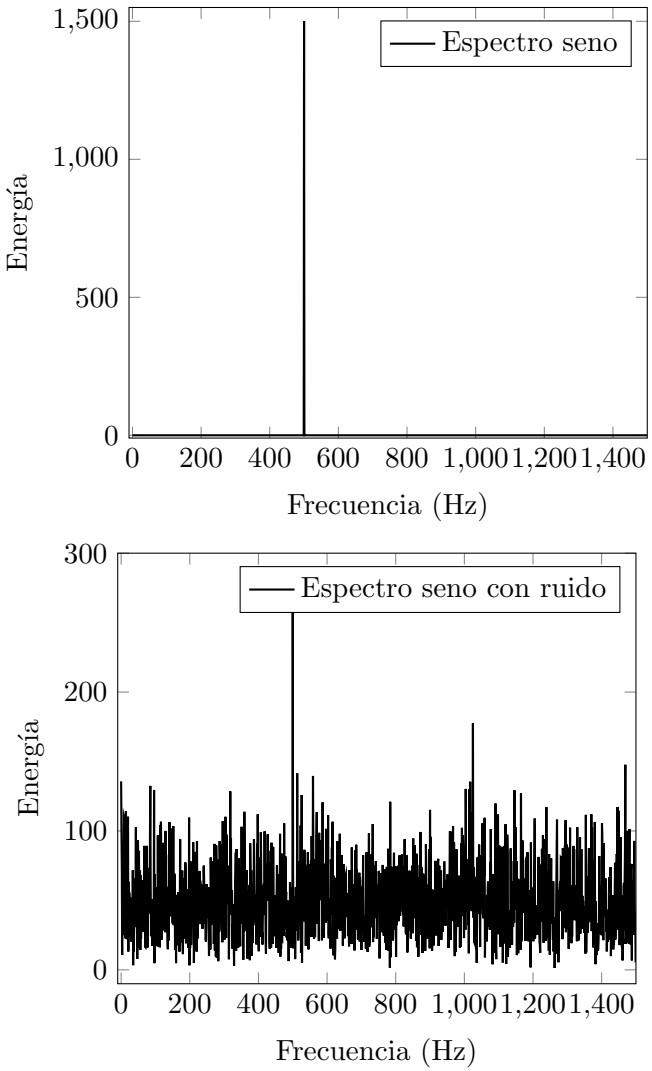


Figura 3.8: Comparación de espectro de onda sinusoidal sin ruido y onda sinusoidal con ruido

multiplican a la señal. En esto se basa la Transformada de Fourier de Tiempo Reducido (Short-time Fourier transform, **STFT**) La fórmula de la STFT en tiempo discreto es:

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{3.7}$$

En la figura 3.9 vemos un ejemplo del efecto de la multiplicación de una ventana triangular a una señal para obtener un fragmento de esa señal para que posteriormente se aplique la DFT a ese fragmento. La ventana se desplazará hasta haber recorrido toda la señal

A partir de todos los espectros locales obtenemos el diagrama conocido como **espectrograma**, (Figura 3.10) que muestra la evolución del espectro en el tiempo. El espectrograma es una representación en 3 dimensiones. Sin embargo, la mayoría de veces se realiza una representación en 2 dimensiones, usando el color para representar la dimensión restante. En un espectrograma de este tipo, el eje X contendrá los valores temporales, y el eje Y contendrá las diferentes frecuencias. Mediante colores, representaremos los pesos de cada una de esta frecuencia en el espectro.

Antes hemos dicho que para realizar la STFT tendremos que aplicar una ventana. Una ventana no es más que una función que únicamente es distinta de cero en un rango limitado de tiempo. Por lo tanto, existen diversas funciones que pueden dar lugar a una ventana. Ejemplos de ellas son la ventana rectangular, la ventana triangular, la ventana de hamming o la ventana de Blackman, cada una de ellas definida por una función concreta. No existe una ventana que sea *la mejor*, por lo que la elección dependerá del problema en el que se aplique.

Las ventanas nos permiten evitar las discontinuidades al principio y al final de los bloques analizados. Además de la elección de la función, existen dos parámetros, relacionados con la ventana, que determinarán en gran medida el resultado de la STFT: la duración y el solapamiento.

- **Duración de la ventana** La duración de la ventana suele darse en número de muestras, y de ella depende la resolución que podemos obtener en frecuencia. Por razones de optimización de cálculo, este valor suele ser una potencia de 2. Cuanto mayor sea la longitud de la ventana, mayor será la resolución del espectro, y viceversa. Sin embargo, usar una ventana grande supondrá una mala resolución temporal. Por otra parte, tamaños pequeños de ventana, tendrán serios problemas para analizar las bajas frecuencias.

- **Solapamiento** Si los desplazamientos que realiza la ventana son de un número de muestras igual al número de muestras de la ventana decimos que no hay solapamiento. Este hecho lleva asociado que si aumentamos la resolución temporal, disminuirá la resolución en frecuencia, y viceversa. Además, tenemos el problema del *leakage*: discontinuidades en los bordes de la ventana de análisis producen esparcimiento de ruido en el resto del espectro. Muchas de las funciones utilizadas en las ventanas intentan reducir el leakage eliminando las discontinuidades y realizando un solapamiento entre las distintas ventanas del análisis, es decir, la ventana avanza un número de muestras menor a su propia longitud.

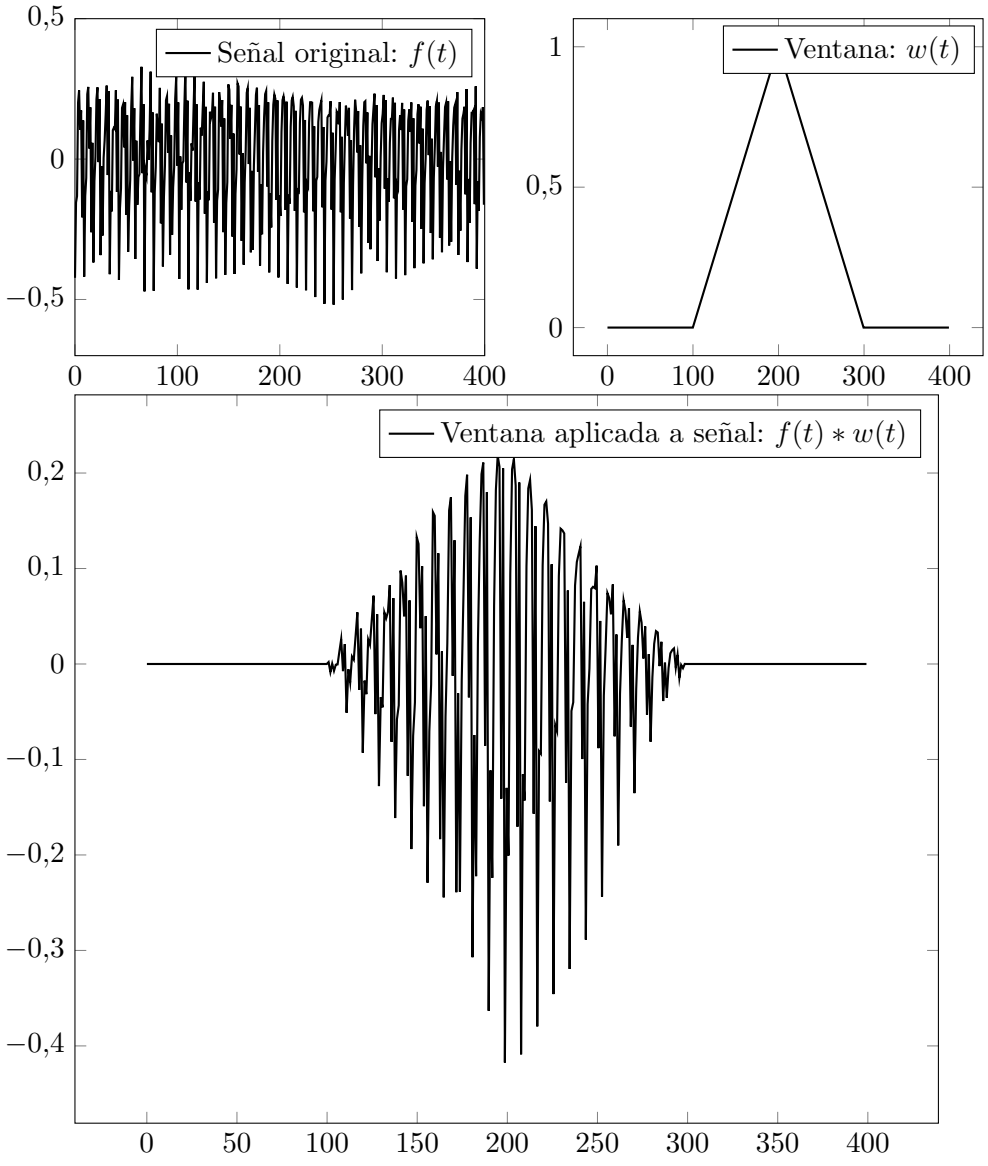


Figura 3.9: Ventana triangular aplicada a señal

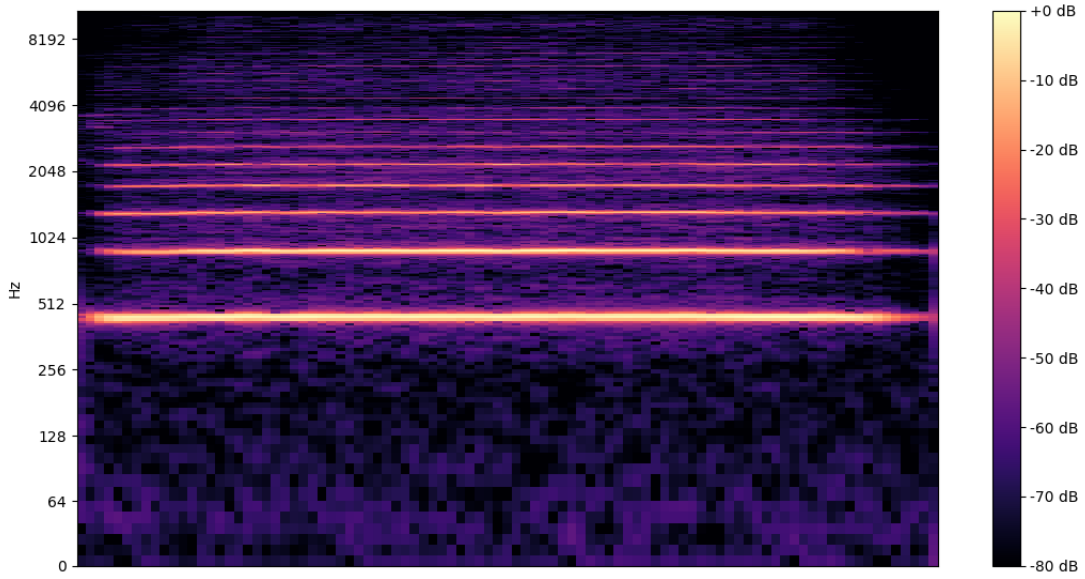


Figura 3.10: Espectrograma de una señal armónica

3.6. Acústica de los instrumentos

(Revision)En la estructura de cualquier instrumento musical encontramos siempre dos partes diferenciadas: uno o varios **excitadores** y uno o varios **resonadores**.

El intérprete suministra una energía al excitador, que empieza a vibrar emitiendo un sonido. Parte de este sonido se transmite al resonador, el cual filtra y modifica esta señal y la acondiciona para ser escuchada. Las familias que conocemos de instrumentos: percusión, cuerda, viento, etc, dependen precisamente del tipo de excitador que usen estos.

Esta combinación de excitadores y resonadores hace que cada instrumento tenga un espectro característico. Dependiendo del instrumento se excitarán unos modos de vibración concretos, dando lugar a ciertos armónicos.

3.6.1. Excitador

El excitador es el elemento vibrante al que el intérprete le suministra energía. Cada familia de instrumentos tiene un excitador.

- **Instrumentos de cuerda**: El excitador es una o varias cuerdas que vibran transversalmente a su eje. La excitación se puede producir, dependiendo del instrumento, pulsando, frotando , golpeando, o incluso soplando a la cuerda. La frecuencia del sonido que se genera depende de la tensión y de la longitud de las cuerdas. De hecho, el intérprete ejerce presión en determinados puntos de la cuerda para *acortarla* y conseguir interpretar una nota más aguda que la de la cuerda *al aire*.

- **Instrumentos de viento**: El excitador depende del instrumento concreto, pero siempre es un elemento que mueve el aire en el interior de un tubo. En los instrumentos de viento madera la vibración se produce al soplar a una o dos lengüetas móviles, dependiendo del instrumento. En los de viento metal, el sonido se produce mediante la resonancia del aire con la vibración de los labios del intérprete que los aprieta contra la boquilla forzando el aire entre ellos para que vibren. En los instrumentos de viento, el excitador no es el responsable de la frecuencia de la nota interpretada, sino que depende del resonador.

- **Instrumentos de percusión**: El excitador es un elemento que está preparado para vibrar al recibir un golpe por parte del intérprete. Si el elemento excitado es una membrana tensada por sus extremos, hablamos de **membranófonos**, mientras que si el excitador es una o varias piezas de material sólido, estaremos hablando de **ideófonos**. Los elementos que vibran en este caso ya no son unidimensionales. Debido a la gran cantidad de frecuencias que se excitan, y la gran cantidad de inarmónicos, en muchas ocasiones es muy difícil percibir una altura definida en el sonido que estos producen. Este hecho se produce, en mayor medida, en membranófonos.

Los excitadores son elementos muy difíciles de modelar, pues su comportamiento es no lineal puesto que su respuesta no es proporcional a la energía que se le suministra. Es

habitual recurrir a métodos empíricos para modelar su comportamiento.

3.6.2. Resonador

El resonador filtra y amplifica la señal producida por el excitador. EL objeto que actúa como resonador depende de la familia de instrumentos.

- **Instrumentos de cuerda:** El elemento resonador suele ser, en los instrumentos de cuerda, la caja de resonancia. Dependiendo de las dimensiones, la caja de resonancia potencia ciertas zonas del espectro, además de amplificar la señal.
- **Instrumentos de viento:** El resonador en los instrumentos de viento suele ser un tubo dentro del cual se forma una onda estacionaria. Será la longitud de este tubo la que determine la frecuencia emitida. Mediante los agujeros que estos tubos suelen tener, introducimos nodos en la onda estacionaria modificando así la frecuencia emitida. En los instrumentos de viento metal se puede modificar la frecuencia emitida abriendo y cerrando válvulas que dejan que el aire pase a diferentes tubos de distinta longitud. En otras ocasiones, como en el trombón de varas, se consigue modificar la frecuencia producida variando la longitud del tubo.
- **Instrumentos de percusión:** En este caso el elemento de resonador suele ser una caja que únicamente filtra y amplifica el sonido producido por la membrana o por el elemento vibrante.

La información del instrumento, por lo tanto, será de gran valor para la estimación de acordes, pues sabiendo qué armónicos produce el instrumento seremos capaces de estimar cómo sería el espectro del sonido generado al interpretar un acorde o nota concretos.

4 Métodos ACE

En este capítulo se estudiarán los principales componentes de los métodos de estimación automática de acordes, desde sus inicios en la década de los 80 hasta la actualidad. Los primeros métodos no eran más que pequeñas características añadidas a los sistemas transcripción polifónica [Chafe et al., 1985] [Chafe and Jaffe, 1986] [Martin, 1996] [Kashino and Hagita, 1996]. Fue [Fujishima, 1999] quien consideró por primera vez la estimación automática de acordes como una tarea independiente. La mayoría de los métodos de estimación automática de acordes tienen una estructura similar. El primer paso principal de todos ellos consiste en separar los audios en varias porciones, y convertir cada una de ellas en un vector de características. El segundo paso consiste en la utilización de técnicas de similitud de patrones para comparar cada uno de estos vectores con un conjunto de modelos predefinidos de acordes. Prácticamente cualquier algoritmo de estimación de acordes realizará estas dos tareas.

El conjunto de posibles acordes que un algoritmo ACE será capaz de estimar es un conjunto limitado. Por lo tanto, un primer paso muy importante para este proceso será definir el conjunto Λ de posibles acordes. Por ejemplo, es común el uso del siguiente conjunto, que contiene las triadas mayores y menores:

$$\Lambda = \{C, C\#, D, ..., B, Cm, C\#m, Dm, ..., Bm\} \tag{4.1}$$

Además, existen técnicas de procesado que permiten mejorar la eficacia en la detección. Este tipo de técnicas son conocidas como técnicas de **prefiltering** o de **postfiltering**, dependiendo de si son realizadas antes o después de la comparación con los patrones modelo. En la figura 4.1 vemos las principales tareas que se realizan en un sistema ACE.

En este capítulo, se detallarán todas estas técnicas. Muchas de ellas serán utilizadas por nuestro algoritmo, siendo capaces con ello de cuantificar hasta qué punto su uso y combinación benefician a la estimación.

4.1. Adaptación previa de la señal

Existe un conjunto de herramientas que adaptan y filtran la señal para obtener únicamente la información que nos será útil en la detección de acordes, eliminando componentes no deseadas que podrán entorpecer el proceso. Algunas de estas técnicas son aplicadas en el dominio del tiempo y otras, en el dominio de la frecuencia. Este tema ha sido tratado por diferentes autores [Pauws, 2004],[Ono et al., 2008] [Reed et al., 2009] que han intentado encontrar qué partes de la señal sería beneficioso eliminar para favorecer la detección. Este tipo de técnicas siempre se aplican antes de realizar la extracción de características.

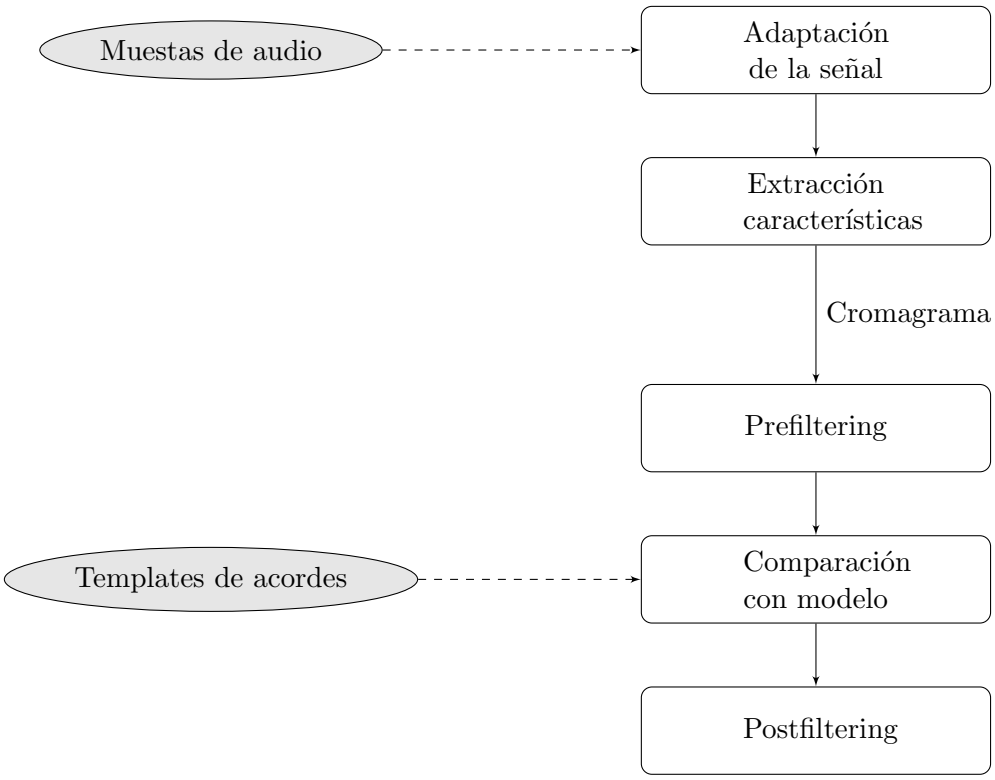


Figura 4.1: Principales tareas de sistema de estimación automática de acordes

Separación de parte armónica y percusiva

Un ejemplo de la adaptación de la señal para la estimación de acordes es la separación de las partes armónica y percusiva. Los sonidos percusivos, que no aportan ningún tipo de información sobre el acorde; habitualmente ocupan un gran ancho de banda pero, sin embargo, son estrechos en tiempo. Este gran ancho de banda, que no contiene ningún tipo de información de armonía, *ensuciará* el espectro, haciendo más difícil analizar las componentes armónicas. Esta separación de parte armónica y percusiva es tratada por primera vez en [Ono et al., 2008] y utilizado para sistemas ACE por [Reed et al., 2009], [Ni et al., 2012] y [Chen et al., 2012] entre muchos otros. Además hay que destacar los algoritmos de separación de parte armónica y percusiva propuestos por [Fitzgerald, 2010] y [Driedger et al., 2014] en los que basaremos nuestra implementación.

Puesto que los sonidos percusivos son muy estrechos en tiempo, los métodos para eliminar la parte percusiva estarán basados en el **filtro de mediana**. En un espectrograma la parte armónica de la señal aparecerá, en gran medida, con lineas horizontales mientras que la parte percusiva aparecerá mediante líneas verticales, al ser sonidos de muy corta duración pero con un gran ancho de banda. En la figura 4.2 vemos un ejemplo de este efecto:

Filtrado de armónicos

Como se ha analizado previamente, cuando un instrumento emite una nota concreta, además de la frecuencia fundamental se producirán una serie de armónicos en frecuencias superiores y subarmónicos en frecuencias inferiores. En la figura 3.6 veíamos los espectros producidos tanto por una flauta como por un piano. Estos armónicos pueden confundir a los sistemas ACE, y por ello, varios autores [Pauws, 2004] [Ueda et al., 2010] [Lee and Slaney, 2006] [Papadopoulos and Peeters, 2007] [Mauch and Dixon, 2010c] [Varewyck et al., 2008] han intentado eliminar estos armónicos. Este filtrado de armónicos no es más que la implementación de un **filtro paso bajo**. Por ejemplo, en [Pauws, 2004] se realiza un filtro paso bajo con una frecuencia de corte de 5 kHz. De esta forma, eliminamos los armónicos de la parte alta del espectro que no tienen ninguna utilidad en el proceso de estimación de acordes. Teniendo en cuenta que la nota más aguda de un piano, un C_8 (que rara vez escucharemos) tiene una frecuencia alrededor de los 4200Hz, cortar a esta frecuencia nos asegurará no perder ninguna frecuencia fundamental de este instrumento (uno de los que puede llegar a octavas más altas), pero quitarnos la mayor cantidad posible de armónicos. Incluso se podría disminuir esa frecuencia de corte y, aunque comenzaríamos a perder algunas notas muy agudas de ciertos instrumentos, la mejora en la estimación que supone el quitarnos esa gran cantidad de armónicos haría que mereciera la pena.

4.2. Extracción de características

De la misma forma que el oído humano realiza una transformación del dominio del tiempo al dominio frecuencial, por proporcionarle este último una mayor cantidad de

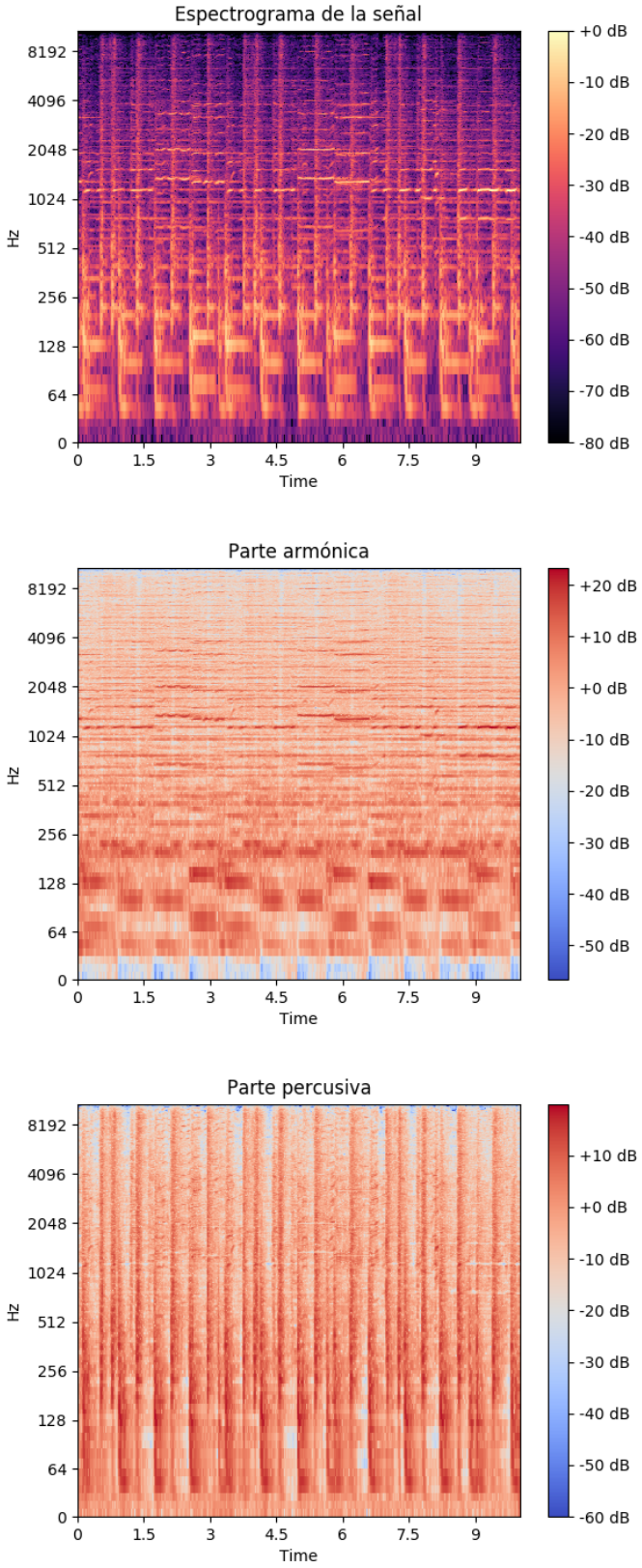


Figura 4.2: Comparación del espectro de la parte armónica y de la parte percusiva de los primeros segundos de la canción "Love Me Do"de The Beatles

información, los sistemas ACE necesitarán extraer una serie de características presentes en el dominio de la frecuencia.

4.2.1. Cromagrama

La principal representación del audio utilizada en los sistemas ACE es el cromagrama. Un cromagrama consiste en una matriz en la que cada fila representa una de las 12 posibles notas musicales y cada columna representa un punto discreto del tiempo. Cada columna del cromagrama se conoce como **vector de croma**, y representa la potencia de cada una de las 12 notas para un instante concreto.

La primera referencia a este representación fue realizada por [Shepard, 1964], donde se definían dos dimensiones (altura del tono y croma) que eran de utilidad para explicar cómo funciona el sistema auditivo humano. En este trabajo, la palabra croma se refería a la nota musical mientras que la altura del tono hacía referencia a la información de la octava.

Fue Fujishima [Fujishima, 1999] quien, en 1999, utilizó por primera vez el cromagrama para realizar la estimación de acordes. En dicho trabajo, se habla de los PCP, **Pitch Class Profile**, que no son más que vectores de croma. Para su obtención, se comenzaba realizando Transformada Discreta De Fourier a un segmento de audio y a partir de esta, se calculaba la potencia en un conjunto de bandas de frecuencia cercanas a cada una de las notas musicales. Posteriormente se plegaba el vector de forma que obteníamos un vector de 12 dimensiones, correspondientes a cada una de las notas. En cada una de ellas, estaría la información de todas las octavas de esa nota.

La base de la obtención de los cromagramas se ha mantenido desde entonces, aunque existen ciertos elementos añadidos posteriormente que han supuesto importantes mejoras.

El paso principal para obtener un cromagrama es el cambio del dominio del tiempo al dominio de la frecuencia. Para ello, hacemos uso del análisis de Fourier. Puesto que queremos una representación de cómo varía el espectro con el tiempo, es común en los algoritmos de obtención de cromagramas el uso de la **STFT** o Transformada de Fourier de Tiempo Reducido, la cual hace uso de ventanas deslizantes a través de la señal para calcular los diferentes espectros. Este método tiene un problema: la STFT usa un tamaño fijo de ventana. Como se ha analizado previamente, tamaños pequeños de ventana supondrán problemas a la hora de detectar frecuencias más graves. Por el contrario, tamaños grandes de ventana, supondrán una baja resolución temporal.

Como solución al problema del tamaño fijo de la ventana en la SFTF, surge la transformada Q constante. Esta solución, usada por primera por [Brown, 1991], hace uso de un tamaño de ventana variable, dependiendo de la frecuencia. Así, frecuencias bajas podrán usar ventanas más grandes en tiempo y viceversa. Esta técnica se ha utilizado en diversas ocasiones recientemente, [Yoshioka et al., 2004] [Bello and Pickens, 2005] [Mauch et al., 2009] [Ni et al., 2012] [Chen et al., 2012]

Tras haberse obtenido una representación en frecuencia de la señal, hay que *plegar* el espectrograma para obtener un vector de 12 dimensiones, una por cada nota, puesto que no nos interesa la información de la octava. Además, se aplica un proceso de nor-

malización que hace al cromagrama independiente de la intensidad general de la obra musical.

En la figura 4.3 vemos la diferencia entre el espectrograma y el cromagrama de una misma canción. En cierto modo, podría decirse que el cromagrama es un espectrograma que ha sufrido un proceso de cuantificación.

4.2.2. Información de la línea de bajo

El cromagrama no proporciona información sobre las octavas, puesto que sólo es capaz de mostrar la energía en 12 posibles notas. En ocasiones, es de gran utilidad obtener un cromagrama distinto para las frecuencias bajas. Este método, propuesto originalmente por[Sumi et al., 2008] permite incorporar información sobre la línea de bajo al algoritmo de estimación permitiendo mejorar la eficacia de la estimación general. Además, obtener un cromagrama exclusivo para las bajas frecuencias, permitirá identificar la **inversión** en la que se encuentran los acordes.

4.2.3. Afinación

Aunque no es común, existen algunas canciones en la música popular que no usan la afinación estándar (**A4=440Hz**). Por ello, se calcula el espectrograma con una granularidad de 3 o más bandas de frecuencia por semitono, permitiendo una cierta flexibilidad con respecto a la afinación utilizada. [Harte and Sandler, 2005] [Harte et al., 2006]. En la figura 4.4 vemos la forma de obtener el cromagrama cuantizado en[Harte and Sandler, 2005], que es capaz de lidiar con este problema

4.3. Prefiltering

Tras haber obtenido el cromagrama, en ocasiones se realizan ciertas modificaciones sobre él de forma previa a la comparación con los patrones. Estas técnicas tienen como fin suavizar los cromagramas, mediante el uso de diversos filtros.

4.3.1. Suavizado temporal

Tras la fase de extracción de características, obtenemos una gran cantidad de vectores de croma muy diferentes. Esto supondría, al analizar estos patrones, varios cambios de acorde por segundo. Pretendemos evitar esa gran cantidad de cambios, pues en realidad los cambios de acorde en las canciones no se producen a tanta velocidad. Además, la extracción de cromagramas está basada en la **STFT**, la cual usa ventanas de un tamaño fijo para recorrer la señal y obtener su espectro. En muchas obras musicales encontramos lo que conocemos como arpegios, que consiste en un acorde en que las notas se interpretan de forma sucesiva. Si nuestra ventana no es lo suficientemente grande, no será capaz de abarcar todo el arpegio, sino únicamente una parte de él, lo que podrá dar lugar a errores en la estimación al no tener la información completa de todas las notas del acorde.

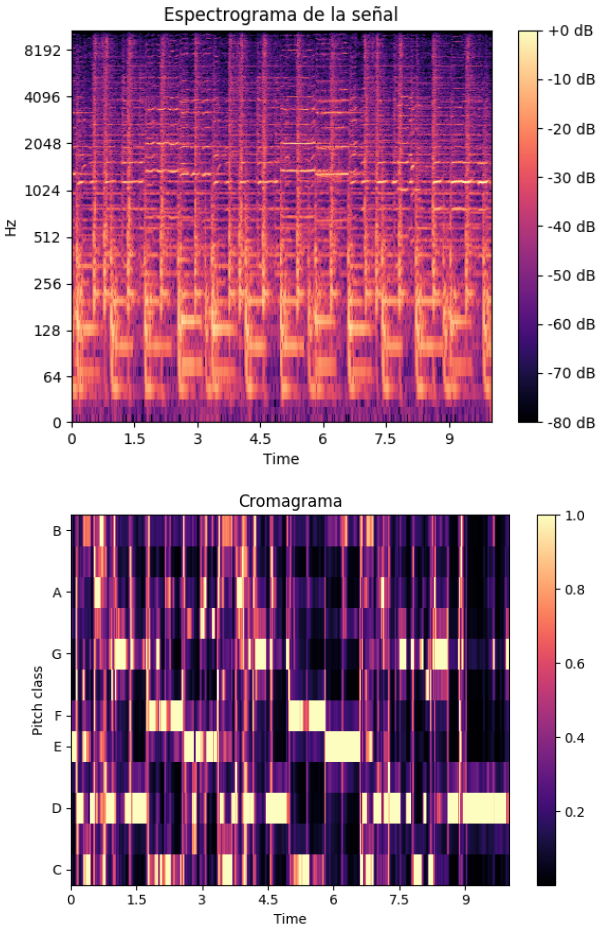


Figura 4.3: Diferencia entre espectrograma (arriba) y cromagrama (abajo)

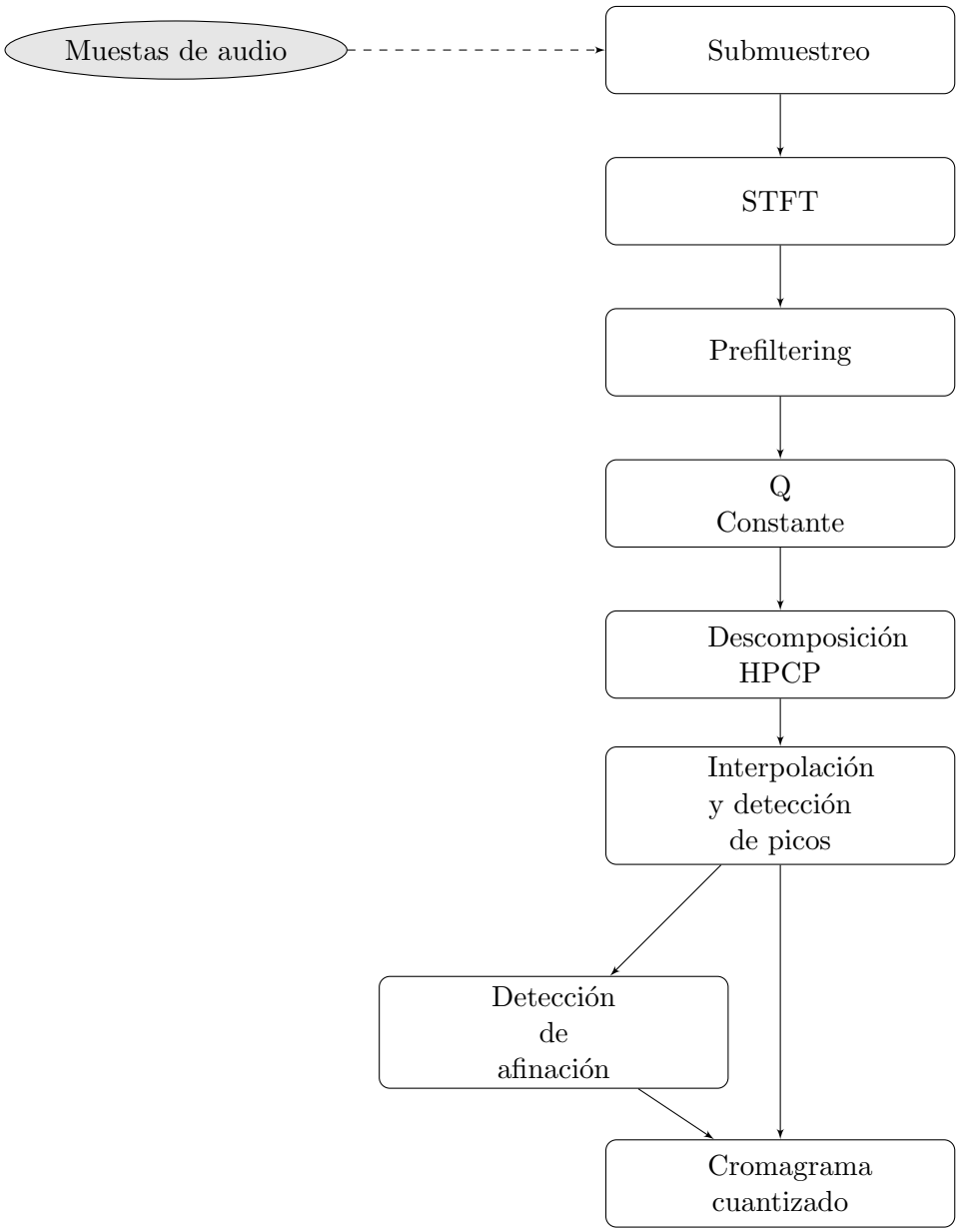


Figura 4.4: Sistema de obtención de cromagramas cuantizados, independientes de la afinación

Una posible solución sería aumentar el ancho de la ventana, pero perderíamos resolución temporal en los pasajes de la obra donde hay cambios más rápidos de acorde. Existe otra solución que consiste en filtrar el audio para suavizarlo y así ser capaces de tener toda la información del arpegio dentro de la ventana de análisis. Para realizar esta tarea, se suelen aplicar filtros de media o de mediana. El tamaño de la máscara usada en el filtro deberá ser también objeto de análisis empírico, hasta obtener el tamaño que mejores resultados proporciona en la estimación.

En la figura 4.5 vemos el efecto del suavizado en un espectrograma.

4.3.2. Sincronización con pulso

En relación a lo visto en el apartado anterior, en la música occidental es poco común que se produzca un cambio de acorde a mitad de pulso. Es por ello, por lo que se suele realizar una tarea de sincronización con el pulso. Esta sincronización puede realizarse en la fase de prefiltering, sobre los cromagramas, o puede realizarse tras la estimación de los acordes, sobre los acordes obtenidos.

Para realizar la sincronización con el pulso en la fase de prefiltering es común tomar todos los vectores de croma que existan entre dos pulsos y obtener la **media** o la **mediana** de todos ellos. De esta forma obtendremos **un único vector de croma** entre cada par de pulsos, asegurándonos de que, tras la estimación, no existirán cambios de acorde a mitad de pulso. Existen ocasiones en las que esta tarea se realiza antes de pasar al dominio de la frecuencia, es decir, se aplica el filtro en el dominio del tiempo, sobre las propias muestras de audio.

Además, de esta forma también conseguiremos disminuir la carga del sistema, teniendo que analizar únicamente un patrón para cada pulso. Como es obvio, esta tarea supondrá añadir a nuestro sistema la capacidad de detectar el pulso de las canciones, aunque esto es una tarea bastante avanzada dentro de la extracción de información musical, que no debe suponer ningún problema.

Encontramos ejemplos de sincronización con el pulso en [Yoshioka et al., 2004] o [Bello and Pickens, 2005] entre muchos otros. En la figura 4.6 vemos el efecto de la sincronización del cromagrama con el beat mediante el uso de filtros de mediana.

4.4. Comparación con modelo

Tras obtener los vectores de características llegamos al paso clave, estimar a qué acorde corresponde cada uno. Para este paso, tendremos que comparar cada uno de estos vectores con cada uno de los patrones de acordes que habremos definido previamente. El nombre del acorde correspondiente al patrón con mayor similitud al vector analizado, será el asignado para ese vector. En la figura 4.7 vemos un diagrama del proceso.

Existen ciertas **ambigüedades** que dificultan la tarea de la detección de los acordes. Podemos encontrar tríadas que a su vez están contenidas en otras tetradas, o acordes que comparten 2 notas con otro distinto. Por ejemplo, el acorde de séptima mayor de Do está formado por las notas **Do, Mi, Sol Si**. Fijándonos, podemos ver que las tres últimas notas corresponden con el acorde de Mi menor, encontrándonos por lo tanto ante

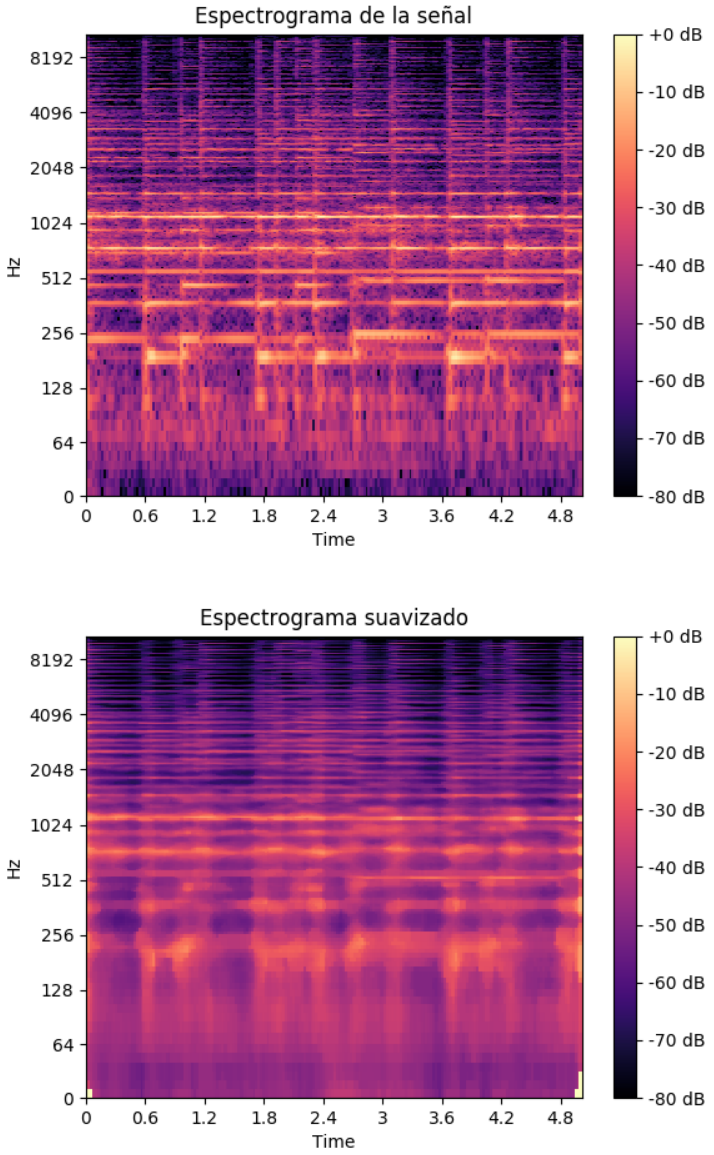


Figura 4.5: Diferencia entre espectrograma antes del suavizado mediante un filtro de mediana (arriba) y después del mismo (abajo)

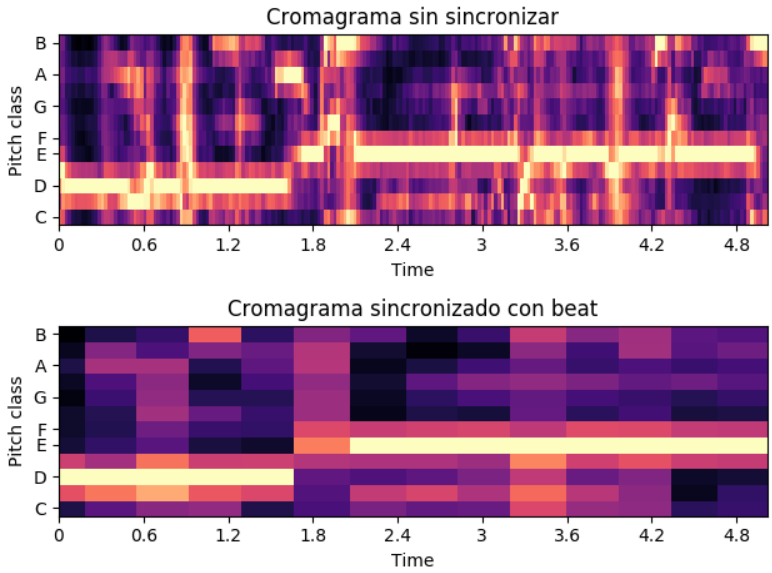


Figura 4.6: Cromagrama antes y después de sincronizar con el beat

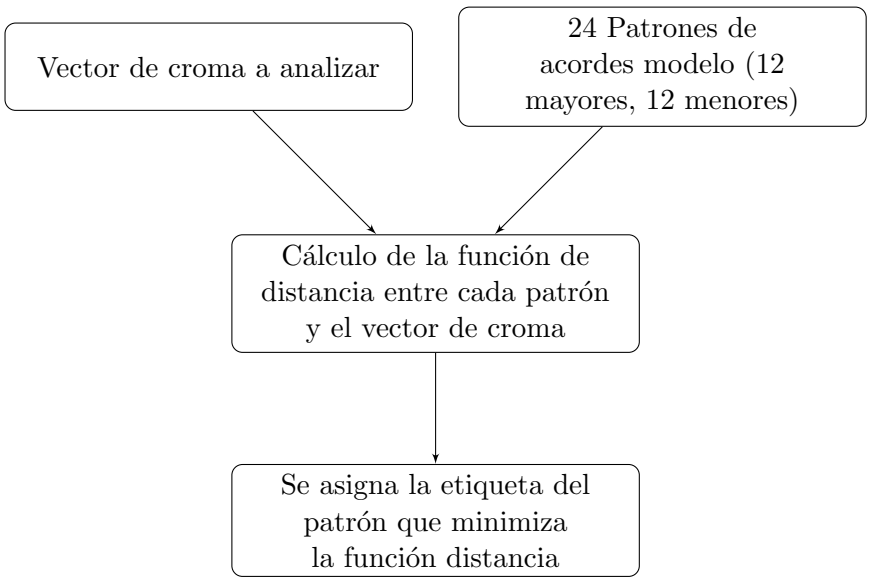


Figura 4.7: Diagrama de sistema de comparación con patrones de acordes

Notas \ Acordes mayores	C	$C\#$	D	$D\#$	E	F	$F\#$	G	$G\#$	A	$A\#$	B
C	1	0	0	0	0	1	0	0	1	0	0	0
C#	0	1	0	0	0	0	1	0	0	1	0	0
D	0	0	1	0	0	0	0	0	0	0	1	0
D#	0	0	0	1	0	0	0	0	0	0	0	1
E	1	0	0	0	1	0	0	0	0	0	0	0
F	0	1	0	0	0	1	0	0	0	0	0	0
F#	0	0	1	0	0	0	1	0	0	0	0	0
G	1	0	0	1	0	0	0	1	0	0	0	0
G#	0	1	0	0	1	0	0	0	1	0	0	0
A	0	0	1	0	0	1	0	0	0	1	0	0
A#	0	0	0	1	0	0	1	0	0	0	1	0
B	0	0	0	0	1	0	0	1	0	0	0	1

Figura 4.8: Matriz de patrones de acorde mayores ideales

una posible ambigüedad. Si nuestro conjunto Λ de posibles acordes únicamente tuviera las tríadas mayores y menores, no podría decidir si el acorde que suena es Do mayor o Mi menor. Por lo tanto, en muchas ocasiones, este problema se solucionaría ampliando el rango de posibles acordes en la detección. Sin embargo, esto podría dar lugar a una mayor probabilidad de error en los momentos en los que no existía ambigüedad.

4.4.1. Patrones ideales

Los patrones ideales son los más básicos. Son patrones binarios de 12 elementos donde tenemos ceros en todas las posiciones, menos en las posiciones correspondientes a las notas del acorde, en las cuáles tendremos unos. Estos patrones corresponderían al vector de croma de un acorde ideal, sin ningún tipo de armónico o ruido externo, y con exactamente la misma intensidad en cada nota. Por ejemplo, el acorde de Re mayor, cuyas notas son Re, $Fa\#$ y La daría lugar al siguiente vector de croma ideal x:

$$x = (0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0)^T \tag{4.2}$$

En la figura 4.8 vemos la matriz con los patrones ideales de los acordes mayores.

4.4.2. Patrones ideales con armónicos

Como se ha comentado previamente, el sonido real de los instrumentos dista mucho de ser como el de los patrones ideales. Cuando escuchamos un acorde procedente de cualquier instrumento, no suenan únicamente las frecuencias correspondientes a las notas del acorde sino que suenan también sus armónicos. Tomaremos como ejemplo la nota Do. Suponiendo que tenemos un modelo con 6 armónicos, y que es un instrumento donde suenan todos los armónicos, las notas que sonarán al interpretar la nota Do serán:

$$Do, Do, Sol, Do, Mi, Sol \tag{4.3}$$

Nuestro modelo asumirá que la energía en cada armónico disminuye de forma exponencial. Por lo tanto la energía del parcial k^{th} será α^{k-1} donde $\alpha \in [0, 1]$. De esta forma, podemos definir que el vector de croma obtenido al interpretar únicamente la nota Do, sería:

$$c = (1 + \alpha^1 + \alpha^3, 0, 0, 0, \alpha^4, 0, 0, \alpha^2 + \alpha^5, 0, 0, 0, 0) \tag{4.4}$$

Para obtener el vector de croma que resultaría de interpretar un acorde, habría que obtener y sumar el vector de croma de todas sus notas. Este es el resultado del vector de croma para el acorde de Do Mayor:

$$x = (1{,}625, 0, 0{,}28125, 0, 1{,}6875, 0, 0, 1{,}90625, 0{,}0625, 0, 0, 0, 0{,}34375)^T \tag{4.5}$$

Para obtener el resto de acordes mayores simplemente habrá que rotar hacia la derecha este patrón. En la figura 4.9 vemos representados los pesos de cada nota para el acorde de Do Mayor. Se puede observar que tienen más peso la tercera y la quinta, que la propia nota fundamental del acorde. Este método nos resultará especialmente útil si tenemos que realizar la estimación de acordes para una canción con un solo instrumento, del que conocemos los pesos de los parciales que genera.

4.4.3. Patrones reales

Para la obtención de patrones más adecuados a la realidad es común plantear el problema como un problema de aprendizaje supervisado. Puesto que tenemos datasets con una gran cantidad de canciones, junto con anotaciones de los acordes que aparecen en estas, podemos obtener una extensa base de datos de vectores de croma etiquetados con el nombre del acorde al que corresponden. En muchas ocasiones, usar patrones reales únicamente requiere una tarea intermedia de realizar la media de todos los posibles patrones correspondientes a un acorde en concreto. Sin embargo, estos patrones suelen utilizarse en bruto junto con los algoritmos de aprendizaje supervisado analizados en la sección 4.4.5. En dicha sección se profundizará más en las ventajas y desventajas que supone el uso de patrones de acordes reales.

4.4.4. Función de distancia

Para poder comparar cuantitativamente los patrones de acordes con los vectores de croma obtenidos de la pista analizada, tenemos dos opciones. Podemos definir una función que mida el grado de parecido entre cada patrón y el vector analizado y tomar el

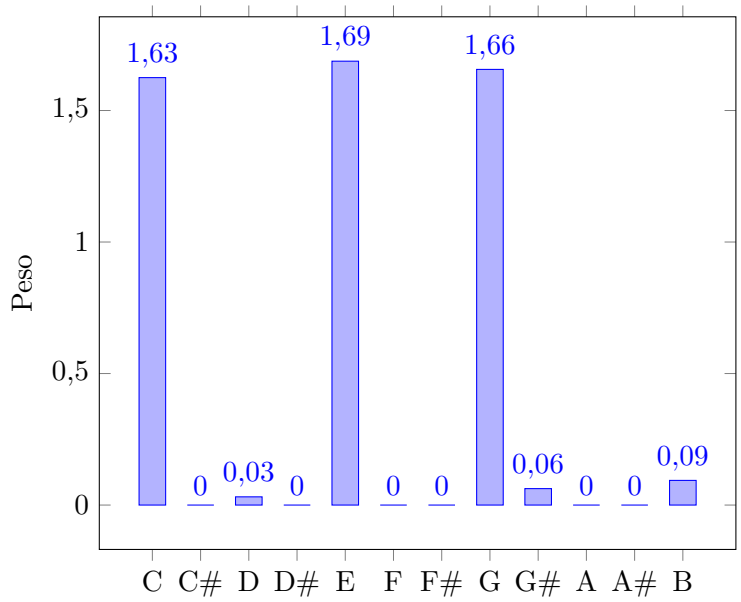


Figura 4.9: Pesos de cada nota en el patrón calculado con 6 armónicos del acorde de Do Mayor.

patrón que nos proporcione la máxima semejanza o, por el contrario, podemos definir una función que calcule la distancia entre cada patrón y el vector de croma y tomar el patrón que minimice esta distancia. Las siguientes, son algunas de las medidas de distancia o similitud entre dos vectores más utilizadas comúnmente para este tipo de propósitos:

Distancia euclídea

La distancia euclídea se define según la siguiente ecuación:

$$D(\overrightarrow{X_1}, \overrightarrow{X_2}) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2} \tag{4.6}$$

Distancia Manhattan

Esta es la ecuación que define la distancia Manhattan:

$$D(\overrightarrow{X_1}, \overrightarrow{X_2}) = ||X_1 - X_2|| = \sum_{i=1}^{n} |x_{1i} - x_{2i}| \tag{4.7}$$

Medida del coseno

Esta es una medida de similitud entre dos vectores en un espacio que tiene definido un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos.

$$similitud = cos(\theta) = \frac{\overrightarrow{X_1} * \overrightarrow{X_2}}{||X_1||||X_2||} \tag{4.8}$$

4.4.5. Aproximaciones de Aprendizaje Supervisado

Aunque hemos estudiado la estimación del acorde mediante el uso de una función de similitud o de distancia, actualmente es muy común encontrar aproximaciones a esta tarea basadas en el aprendizaje automático. Esto se da principalmente cuando se usan patrones de acordes reales, obtenidos a partir de los datasets de audios etiquetados. El uso de técnicas de aprendizaje supervisado en estos casos se debe a la existencia de ruido, el cual no existía al usar patrones ideales. Tendremos varios vectores de croma distintos que definen un posible acorde, y estos pueden ser muy diferentes los unos de los otros. Tendremos que utilizar algoritmos capaces de generalizar las observaciones que ha realizado, a nuevos vectores de croma.

Al igual que a partir del patrón ideal con armónicos de Do Mayor, realizábamos desplazamientos cíclicos del vector para obtener el resto de acordes mayores, es común aplicar esto mismo cuando se hace uso de patrones reales para ahorrar procesamiento y memoria. De esta forma, en lugar de tener que aprender los patrones de 24 posibles clases, habrá que aprender los de 2, acordes mayores y acordes menores.

El dataset de audios etiquetados se suele descomponer, de forma aleatoria en 2 grupos: **entrenamiento** y **validación**. Los vectores del grupo de entrenamiento, como su propio nombre indica, son usados para entrenar al algoritmo y que aprenda los posibles patrones para cada acorde. Una vez que el algoritmo ha sido entrenado, se le pasan los vectores del grupo de validación para comprobar si la estimación realizada por el algoritmo coincide con la etiqueta del acorde. Podemos ver este proceso en la figura 4.10

De esta forma podremos realizar el proceso de validación cruzada para asegurarnos de que el algoritmo es capaz de generalizar cuando se encuentre nuevos vectores de croma desconocidos para él. La técnica de la validación cruzada consiste en realizar diversas evaluaciones del algoritmo, tomando cada vez grupos distintos de entrenamiento y de validación, para después hacer una media de todas las puntuaciones obtenidas. De esta forma nos aseguramos que los resultados son independientes de la partición realizada entre entrenamiento y validación.

A continuación, se realizará un análisis de algunos de los algoritmos de aprendizaje supervisado más usados para esta fase.

KNN

El **KNN** es uno de los algoritmos más sencillos de aprendizaje supervisado. Este algoritmo clasifica nuevas instancias como la clase mayoritaria entre los k vecinos más cercanos de entre todos los datos de entrenamiento.

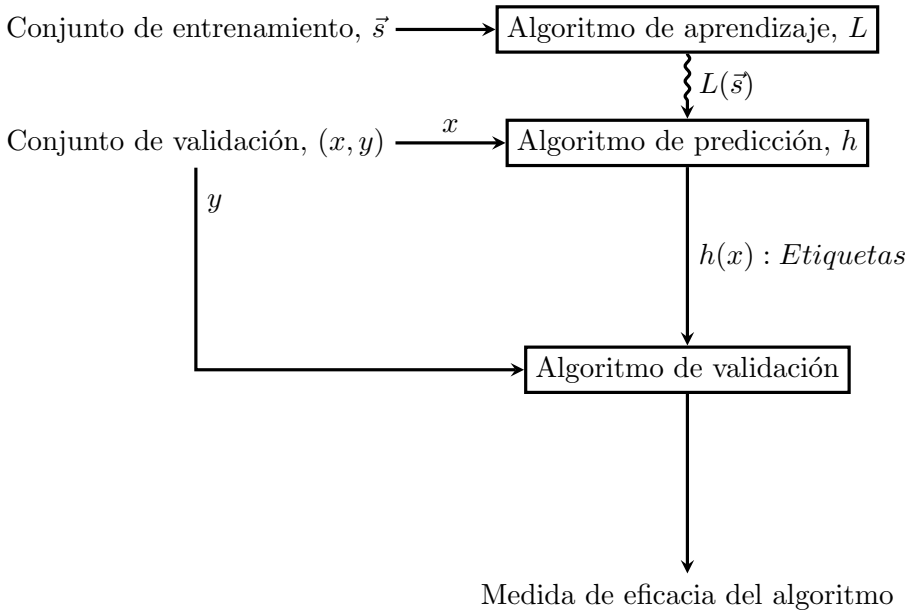


Figura 4.10: Diagrama básico de uso de conjunto de datos de entrenamiento y conjunto de datos de validación

Se suele decir que es un *lazy algorithm*(algoritmo perezoso) pues durante el entrenamiento no se construye ningún tipo de modelo como en otros tipos de algoritmos de aprendizaje supervisado. El entrenamiento del KNN consiste únicamente en almacenar las instancias. Durante la clasificación se realiza lo siguiente:

- Mediante una función de distancia se calcula la distancia desde la muestra a clasificar a todas las muestras del conjunto de entrenamiento.
- Se toman las k muestras más cercanas al elemento a clasificar.
- De todas ellas, se elije la que más se repita. A la salida se devolverá la clase de esta muestra elegida.

Aunque puedan usarse otras, la función de distancia más utilizada para este algoritmo es la distancia euclídea, que hemos visto previamente. El KNN es, además, un clasificador **no paramétrico**, pues no hace ningún tipo de suposición sobre la distribución de los datos. Además, es un clasificador **local**, pues únicamente necesita los k vecinos más cercanos para inferir la clase de la muestra a clasificar.

Es muy importante escoger un valor adecuado para el parámetro k. En la figura 4.11 vemos claramente su efecto. En la gráfica de la izquierda vemos que se ha usado un $k = 1$, y por lo tanto el sistema clasificará la muestra según la clase de la muestra más cercana, en este caso la clase 2. Sin embargo, en la gráfica de la derecha se ha usado un $k = 3$ por lo que se han obtenido las 3 muestras más cercana a la muestra analizada. En este caso, la clase mayoritaria era la clase 1, por lo que esa será la clase que se asigne a la muestra analizada. Por lo tanto, el sistema ha supuesto que, aunque existiera una muestra más cercana, con una clase distinta, al estar aislada y pertenecer todas las demás muestras a la clase 1, ésta era simplemente **ruido**, y tenía que ser obviada.

Hasta el momento, la estimación de los acordes que hemos analizado en apartados anteriores se basaba en tomar la clase del acorde de nuestro conjunto de entrenamiento *más parecido* al vector de croma analizado (el que minimiza una función de distancia). Esto es equivalente a usar un KNN con una $k = 1$. El uso del KNN con valores mayores de k supone ir un paso más allá, proporcionando a nuestro sistema un mayor nivel de inmunidad al ruido.

Máquinas de Vectores de Soporte

Las máquinas de vectores de soporte (en inglés *Support Vector Machine*, SVM) son algoritmos que nacieron como clasificadores binarios, aunque han sido adaptados para ser usados en tareas como regresión, clustering o multiclasificación. Estos clasificadores pertencen a la categoría de clasificadores lineales. Son clasificadores muy potentes, de gran utilidad en la estimación automática de acordes. Esta aproximación al problema se analiza más en profundidad en [Weller et al., 2009]

Las SVM tratan de encontrar las superficies que separan las distintas clases. Cuando tenemos únicamente dos atributos, esta superficie sería simplemente una línea. Sin embargo, cuando tenemos tres atributos, la superficie separadora pasa a ser un plano, y con más de tres atributos, un hiperplano. La principal característica de las SVM, como

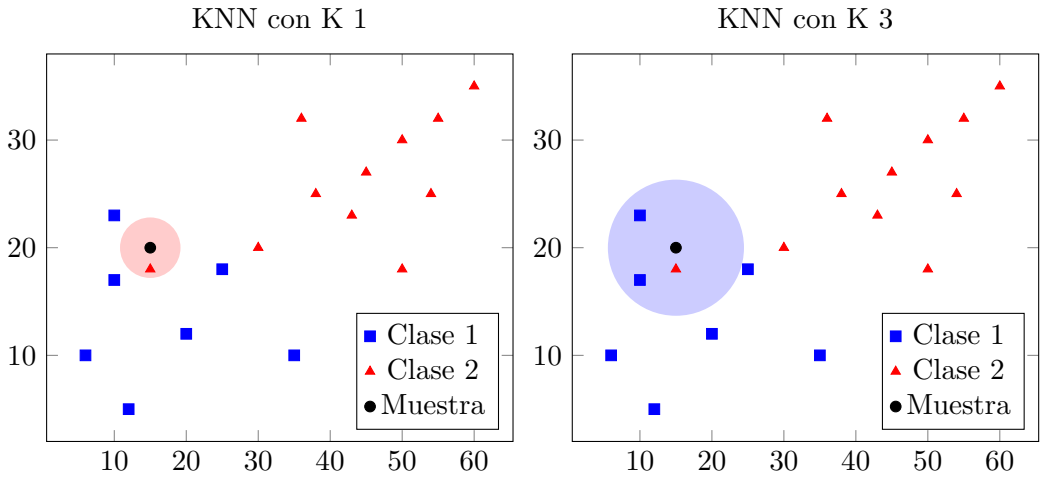


Figura 4.11: Efecto de cambiar el parámetro k de un KNN

vemos en la figura 4.12, es que buscan maximizar el **margen**, la distancia entre el hiperplano, y los puntos más cercanos. En muchas ocasiones, existen infinitos hiperplanos capaces de separar las distintas clases, por lo que se escoge el que proporciona el mayor margen. Por ello, a veces se conoce a las SVM como *clasificadores de margen máximo*. Al vector formado por los puntos más cercanos al hiperplano se le llama vector de soporte.

No siempre existe el hiperplano que separe perfectamente todas las clases, y aunque existiera, correríamos el riesgo de sobreajustar el modelo. Por ello, las SVM tienen un parámetro (C) que controla la compensación entre errores de entrenamiento y los márgenes, permitiendo algunos errores pero penalizándolos en mayor o menor medida según el valor de C.

El hecho de que se busque el hiperplano que separe las distintas clases, no quiere decir que las SVM no sean capaces de tratar datos que no sean linealmente separables. Aquí entran en juego los **kernel** que permiten transformar el espacio de características a uno donde las clases si sean separables linealmente.

Depp Learning

Como no podía ser de otra forma, los grande avances que ha habido en los últimos años en el campo de las redes neuronales y el deep learning, ha tenido su reflejo en la estimación de acordes. En el deep learning se usan estructuras lógicas que intentan imitar a las redes de neuronas de nuestro cerebro. Los algoritmos de este tipo usan una cascada de capas con unidades de procesamiento no lineal para extraer y transformar variables. La entrada de cada capa usa la salida de la capa anterior. Lo que hace que podamos pasar a considerar una red neuronal como aprendizaje profundo es que exista un gran número de transformaciones aplicadas a la señal mientras se propaga desde la

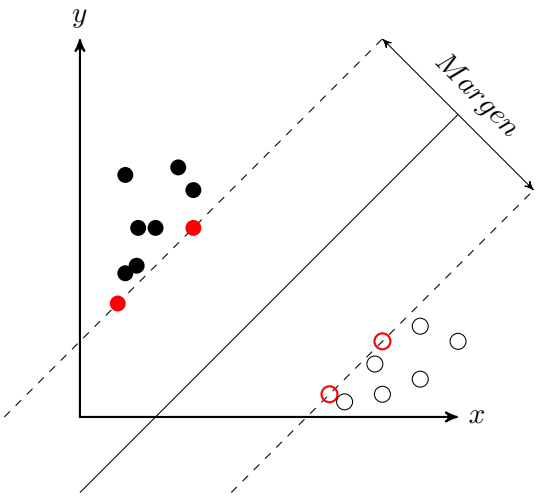


Figura 4.12: Ejemplo de SVM

capa de entrada a la capa de salida. En una red neuronal tenemos 3 capas de neuronas principales que podemos ver en la figura 4.13:

- **Capa de entrada**: Las neuronas de esta capa reciben los valores correspondientes a los distintos atributos.
- **Capas ocultas**: Es un conjunto de varias capas cuyas neuronas contienen cálculos intermedios de la red.
- **Capa de salida**: En los problemas de clasificación, las neuronas de esta capa contienen la clase estimada.

Los principales tipos de redes neuronales usados en este campo son las **redes neuronales convolucionales** [Humphrey and Bello, 2012] y las **redes neuronales recurrentes** [Boulanger-Lewandowski et al., 2013], [Sigtia et al., 2015].

Estas últimas son muy adecuadas para nuestra tarea, debido a sus características. Las redes neuronales recurrentes, como su propio nombre indican, permiten las conexiones recurrentes de una neurona con ellas misma, entre neuronas de una misma capa, o entre neuronas de una capa, con otra capa anterior. Esta arquitectura les permite tener cierta memoria y, por lo tanto, un sentido temporal. Esta es la característica que hace que tengan una gran utilidad en la estimación de acordes. El no basar la estimación únicamente en las muestras actuales, sino que se añada información del *contexto*, repercutirá muy positivamente en la eficacia en la estimación. En la figura 4.14 vemos como hay un traspaso información entre la capa oculta de la primera estimación, a la capa oculta de la red de la segunda estimación, y así sucesivamente en todas ellas. Por eso las redes neuronales recurrentes resultan de una gran utilidad cuando tratamos con **series temporales**.

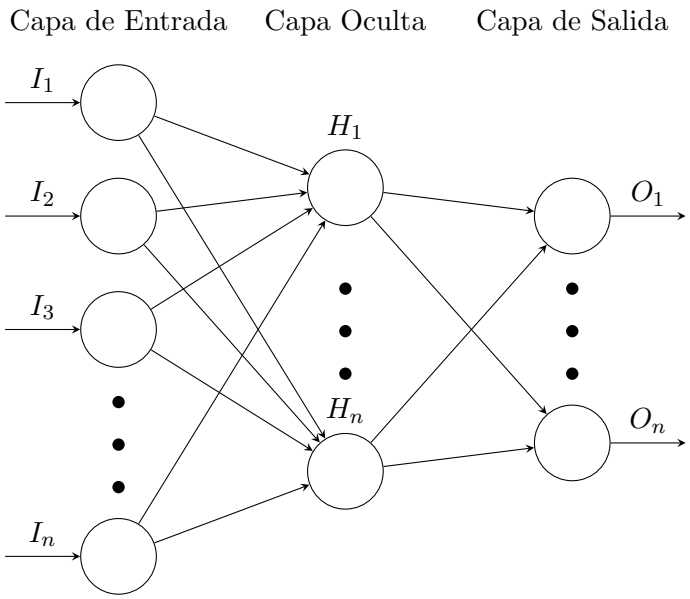


Figura 4.13: Red neuronal con 3 capas

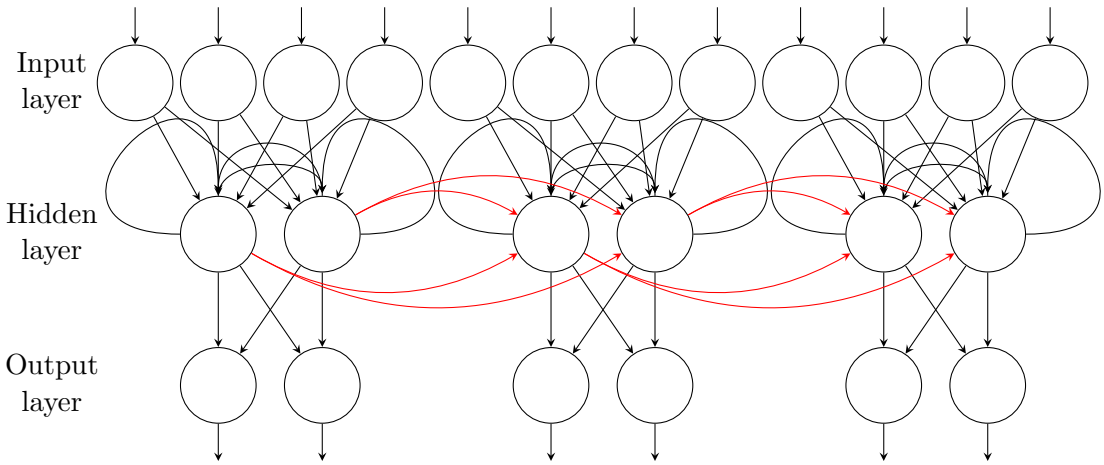


Figura 4.14: Diagrama de red neuronal recurrente

Redes bayesianas dinámicas

Las redes bayesianas dinámicas (RBN) son otros de los clasificadores más útiles para la estimación de acordes. Las redes bayesianas modelan un fenómeno mediante un conjunto de variables y las relaciones de dependencias entre ellas. A partir de este modelo, se puede hacer lo que se conoce como inferencia bayesiana, es decir, estimar la probabilidad de las variables no conocidas, en base a las conocidas. Inicialmente estos algoritmos no eran capaces de *aprender* a partir de los datos, sino que los modelos se construían de forma manual en base a un conocimiento experto. Sin embargo, actualmente existen varias técnicas que son capaces de obtener los parámetros del modelo a partir de los datos. Explicar el complejo funcionamiento de una red bayesiana ocuparía varias páginas, y no es el propósito de este trabajo, por lo que para entender su funcionamiento se remite al lector a [Murphy, 2002].

Destaca el trabajo de [Mauch and Dixon, 2010b] como principal implementación de estas redes a la tarea que nos ocupa. En este trabajo, los parámetros de las RBN se configuran de forma manual en base al conocimiento de expertos musicales. Las entradas del sistema (observaciones de la RBN) son dos cromagramas sincronizados con el pulso (uno para frecuencias altas y otro para frecuencias bajas). Además, las RBN le permiten añadir al modelo información como la métrica, tonalidad o la línea de bajo, consiguiendo así un sistema muy robusto. Por último, hay que destacar un tipo de red bayesiana, los **Modelos Ocultos de Markov**, que debido a su gran importancia en los sistemas ACE, tendrán un capítulo propio más adelante en este trabajo.

4.5. Postfiltering

En **postfiltering** se agruparían todas las técnicas que se aplican posteriormente a la estimación del acorde, con la intención de mejorar la eficacia del sistema solucionando posibles casos dudosos, detectando y eliminando posibles errores en la estimación o suavizando el resultado para evitar demasiados cambios de acorde.

4.5.1. Información de tonalidad

Como hemos visto al hablar de la **armonía funcional**, cada acorde tiene una función definida en la obra. Esta función puede ser de una gran utilidad, puesto que proporciona información sobre qué acorde es más probable tener antes y después del acorde analizado. Aprovechándonos de esta característica podemos detectar posibles errores en la estimación o resolver casos en los que no hayamos sido capaces de elegir la adecuada entre dos o más opciones. Como se ha visto anteriormente, esta información en muchas ocasiones se aplica en la propia fase de estimación, aunque en otros casos nos la encontramos en la fase de **postfiltering**, modificando la salida del clasificador, para hacerla más *coherente*, en base a las reglas de la armonía funcional.

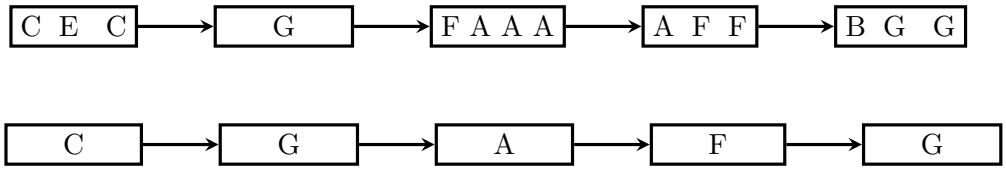


Figura 4.15: Ejemplo de sincronización de la estimación con el pulso en postfiltering. Cada cuadro representa un pulso. En el esquema de abajo vemos el resultado de sincronizar con el pulso el conjunto de acordes mostrados en el diagrama superior.

4.5.2. Sincronización con el pulso

A diferencia de la sincronización con el pulso comentada en el apartado de prefiltering, en este caso la sincronización se realiza posteriormente a la estimación de los acordes. Este proceso permite limpiar el resultado final. La forma de realizarla suele ser más simple que en prefiltering pues, en la mayoría de ocasiones, consiste en obtener simplemente la **moda** entre todas las detecciones realizadas entre dos pulsos. Es decir, asignamos a todo el pulso la etiqueta del acorde que más se ha repetido dentro del mismo. En la figura 4.15 vemos un ejemplo de sincronización con el pulso en fase de **postfiltering**. En ella, cada recuadro simboliza un pulso.

4.6. Estimación basada en modelos ocultos de Markov

Sabemos, y lo hemos analizado en capítulos previos, que hay progresiones armónicas que se repiten en multitud de canciones distintas. Además, hemos visto que en armonía existen ciertas reglas que nos dicen cómo deben resolverse las tensiones provocadas por ciertos acordes. La estimación basada en **modelos ocultos de Markov (HMM)** busca aprovechar esta información contextual para que complemente a la información proporcionada por el vector de croma. Un HMM se puede considerar como la red bayesiana dinámica más simple. Aunque, debido a la gran importancia que tienen en los sistemas ACE, se ha dedicado un capítulo entero a la explicación de estos algoritmos, los HMM estarían entrarían dentro del conjunto de aproximaciones de aprendizaje supervisado a la estimación automática de acordes.

Estos sistemas están basados en un modelo que contiene las probabilidades de pasar de un acorde a otro. Además, los sistemas convencionales son muy inestables, obteniendo a su salida incluso varios cambios de acorde por segundo debido, la mayoría de veces, a la presencia de notas en las líneas melódicas que no forman parte del acorde. Para solucionarlo, hemos visto que se suelen aplicar técnicas de filtrado en prefiltering o incluso en postfiltering. Sin embargo, el uso de los modelos ocultos de Markov, también supone una solución a este problema, que no necesita alterar los cromagramas detectados. Para explicar el funcionamiento de estos sistemas nos basaremos en la explicación del siguiente

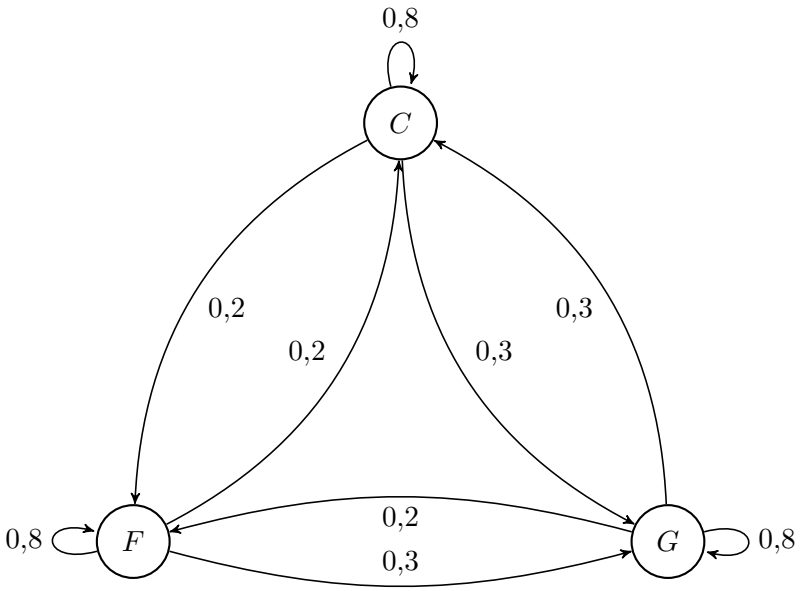


Figura 4.16: Ejemplo de Cadena de Markov para 3 acordes

trabajo de [Heittola et al., 2009].

Cadenas de Markov

Para comprender el funcionamiento de los HMM, es necesario comprender antes el concepto de **Cadena de Markov**. Una cadena de Markov es un proceso estocástico discreto que cumple la propiedad de que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. Una cadena de Markov se define por un conjunto de estados y un conjunto de probabilidades de transiciones entre esos estados. En nuestro caso, los estados serán todos los posibles acordes que el sistema puede detectar. En la figura 4.16 vemos un ejemplo de una serie de acordes con las probabilidades de transiciones entre ellos.

Las cadenas de Markov son **sistemas sin memoria**, pues las probabilidades de que se dé un acorde concreto únicamente dependen del acorde anterior a este.

Existe un elemento más en las cadenas de Markov que aún no hemos comentados, los estados iniciales. Para calcular la probabilidad de una serie de estados necesitamos saber la probabilidad del primer estado, que no depende de ninguna anterior. Por lo tanto, tendremos un conjunto de probabilidades de estados iniciales.

Para calcular la probabilidad de una progresión concreta de estados habrá que multiplicar las probabilidades de todas las transiciones que se producen y la probabilidad del primer estado.

Modelos ocultos de Markov

En los sistemas de detección de acordes, el elemento observado no serán los propios nombres de los acordes, sino que serán un conjunto de vectores de croma. En este caso, por lo tanto, el objetivo no es calcular la probabilidad de un conjunto de estados (una progresión de acordes), pues estos son desconocidos. El objetivo es precisamente descubrir cuáles son esos acordes y es en este punto donde pasamos de las cadenas de Markov a los modelos ocultos de Markov. Estos modelos se formarán con dos capas, una capa oculta, desconocida, de estados, y una capa visible de observaciones, los vectores de croma.

Además de las probabilidades de transiciones entre acordes, ahora añadiremos al sistema las posibilidades de que ciertos acordes **emitan** un patrón de croma concreto.

Por lo tanto, ahora, para calcular la probabilidad de un conjunto concreto de estados a partir de una serie de observaciones, necesitaremos multiplicar también por la probabilidad de que dichos estados emitieran los vectores de croma observados.

De la misma forma, para calcular la probabilidad de una sucesión de observaciones habrá que calcular y sumar las probabilidades de todos los posibles estados que puedan dar lugar a esa serie de observaciones. En la figura 4.17 vemos un ejemplo genérico de HMM.

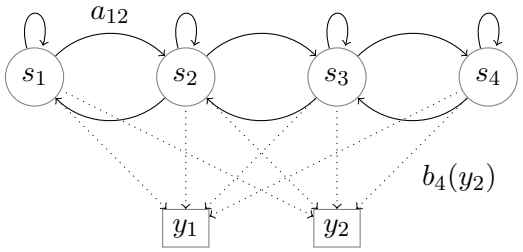


Figura 4.17: HMM con 4 estados que pueden emitir 2 símbolos discretos y_1 or y_2. a_{ij} es la probabilidad de transición desde el estado s_i al estado s_j. $b_j(y_k)$ es la probabilidad de emitir el símbolo y_k en el estado s_j.

Discretización de las observaciones

Puesto que nos hemos basado en la versión discreta de los HMM, para poder aplicar todo lo analizado en esta sección el conjunto de observaciones debe ser un conjunto finito. Sin embargo, podemos tener infinitos vectores de croma. Para solucionar este problema, una posible solución utilizada habitualmente, es crear un conjunto de vectores tipo, y mediante un proceso llamado cuantización, mapear cada posible vector de croma a uno de estos vectores tipo. Muchos de los algoritmos que realizan este proceso están basado en algoritmos de clustering, siendo el centroide de cada cluster, el vector elegido como prototipo.

Definición de parámetros del modelo

Además de los estados y las observaciones, los modelo ocultos de Markov usados en los sistemas de ACE cuentan con tres conjuntos de probabilidades: las probabilidades de los acordes iniciales, las probabilidades de las transiciones entre acordes y las probabilidades de que los cromagramas observados correspondan a cada uno de los posibles acordes. Los métodos utilizados, basados en el aprendizaje supervisado a partir de un conjunto de datos etiquetados, van más allá del alcance de este trabajo y se remite al lector a los trabajos [Durrieu et al., 2009] [Heittola et al., 2009].

Descubrimiento de estados

Volviendo al objetivo final de descubrimiento de los estados (acordes) correspondientes a unas observaciones dadas (cromagramas),lo que tratamos de conseguir es la secuencia que tiene mayor probabilidad. Sin embargo, este proceso requeriría calcular la probabilidad de todas las posibles secuencias, lo que sería extremadamente costoso computacionalmente. Por suerte, existen algoritmos cuyo propósito es realizar el descubrimiento de estados de forma mucho más eficiente. Una de las principales técnicas para ello el el **algoritmo de Viterbi** que utiliza una aproximación recursiva, dividiendo la secuencia de observaciones en varias subsecuencias.

4.7. Evaluación de los resultados

Los investigadores del campo de la obtención de información musical han desarrollado técnicas que les permiten medir, de forma objetiva, cómo están funcionando sus algoritmos. Sin embargo, pequeñas diferencias en las implementaciones de los estándares de evaluación, pueden suponer grandes cambios en los resultados finales. Es por ello, por lo que en 2014 se presentó la librería **mir eval**[1] [Raffel et al., 2014]. Esta librería implementa, en Python, los principales algoritmos de evaluación de las principales tareas de MIR, entre las que se encuentra, obviamente, la estimación automática de acordes.

Uno de los principales problemas que se encontraron los creadores fue la falta de estandarización en la forma de nombrar los acordes. Por lo tanto, la primera decisión fue utilizar una convención de nombres como la explicada en 2.5.4, basada en [Harte et al., 2005].

Con respecto a la forma de comparar 2 etiquetas de acordes (la estimada y el *ground truth*) para ver su acierto, se decidió separar el acorde en tres partes. Para ver claramente cuáles son estas partes, lo veremos sobre un ejemplo: el acorde **D:maj(6)/5**

- ***D***: La raíz del acorde
- ***maj(6)***: La palabra que define el tipo de acorde, junto con los intervalos adicionales añadidos al acorde (en este caso: ***6***).

[1] `https://github.com/craffel/mir_eval`

- ***/5***: El intervalo desde la raíz, a la nota que estará en la posición más grave en el acorde. Con esta anotación en realidad estamos definiendo la inversión del acorde.

A partir de esta representación podemos comparar las etiquetas en función de las siguientes medidas [2] creadas a partir del vocabulario de acordes propuesto en MIREX 2013: [3]

- **root**: Sólo compara la raiz de los acordes
- **majmin**: Sólo compara la raiz y las etiquetas de mayor, menor, y N (sin acorde)
- **majmin_inv**: Compara acordes mayores y menores con inversiones. Por lo tanto, es importante que la tercera parte de la etiqueta, correspondiente a la inversión, aparezca en las 2 etiquetas que comparamos
- **mirex**: Esta medida considera correcta una estimación si comparte al menos 3 notas con el *ground truth*
- **thirds**: Los acordes se comparan a nivel de terceras mayores y menores. Es decir, para que la estimación sea correcta, la raíz y la tercera de la etiqueta estimada tiene que ser igual a la del *ground truth*
- **thirds_inv**: Es igual que la anterior, pero añade la información de la inversión.
- **triads**: Para que dos acordes sean considerados iguales tienen que tener la misma raiz y el mismo intervalo de quinta.
- **triads_inv**: Es igual que la métrica anterior,pero añade la información de la inversión.
- **tetrads**: Dos acordes son iguales si tienen la misma raíz y los mismos intervalos de tercera quinta y séptima.
- **tetrads_inv**: Es igual que la métrica anterior,pero añade la información de la inversión.
- **sevenths**: Compara los acordes según las reglas de MIREX para los acordes de séptima. Es decir, sólo se comparan las etiquetas de acordes mayores, de séptima mayor, menores o de séptima menor.
- **sevenths_inv**: Es igual que la métrica anterior,pero añade la información de la inversión.

  La puntuación de la canción completa se calcula mediante la acumulación del resultado de cada estimación ponderado según la duración del acorde, para todas las estimaciones realizadas.

[2] `https://craffel.github.io/mir_eval/#id2`
[3] `http://www.music-ir.org/mirex/wiki/2013:Audio_Chord_Estimation`

5 Diseño del sistema final

Tras conocer el estado del arte, se ha pasado a experimentar todo lo estudiado sobre un sistema propio de estimación automática de acordes. En todo momento se ha puesto una gran cantidad de esfuerzo en obtener un **sistema altamente modularizado**, de forma que pudiéramos combinar distintos algoritmos en las diferentes partes del proceso, evaluando así la efectividad de cada uno de ellos. Además, queremos que cualquier investigador pueda añadir fácilmente nuevos módulos al sistema que implementen nuevas funcionalidades.

Como veremos más adelante, mediante un archivo de configuración podemos controlar todos los parámetros del sistema, además de activar o desactivar ciertas funciones. Es por ello, por lo que el valor del presente trabajo ya no es tanto la obtención de nuevas técnicas aplicables a la estimación de acordes, sino el estudio de la efectividad de cada una de ellas, así como de los beneficios e inconvenientes de las combinaciones de todas ellas.

5.1. Tecnologías usadas

La implementación del sistema ha requerido del uso de varias tecnologías que serán descritas durante esta sección.

5.1.1. Python

El lenguaje de programación elegido para el sistema final ha sido **Python**. En la figura 5.1 podemos ver su logo. Python es un popular lenguaje de programación interpretado de alto nivel cuya característica principal es su sencillez, legibilidad y precisión de sintaxis. Te permite programar algoritmos de gran complejidad en muy pocas líneas, pero siempre sin perder su sencillez característica. Además, es un lenguaje multiplataforma, y está desarrollado bajo una licencia de código abierto *Python Software Foundation License*, la cual es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1.

Python es un lenguaje muy versátil, que soporta tanto programación orientada a objetos como programación imperativa y funcional. Es por ello por lo que Python no tiene un ámbito específico y puede ser utilizado para desarrollos de todo tipo como aplicaciones científicas, de escritorio, web, de monitorización, de análisis de datos, etc

El ámbito científico y de análisis de datos, cada vez están más cerca de ser liderados por Python. Su principal competidor en estos campos, el lenguaje R, aún estando totalmente enfocado a este sector, no es capaz de evolucionar a la misma velocidad, y está perdiendo progresivamente su situación de liderazgo. Algo parecido pasa con Matlab, otro de los mayores competidores de Python. Matlab es mucho más que un lenguaje de

Figura 5.1: Logo del lenguaje Python

programación. Matlab es un ecosistema de desarrollo que contiene desde el intérprete del lenguaje hasta el propio IDE o las diferentes extensiones que permiten añadir funcionalidad extra a la de la propia librería estándar del lenguaje. Sin embargo, el ser software propietario con un fin comercial provoca que, el de Matlab, sea un ecosistema mucho más cerrado que el de Python. Las contribuciones de la comunidad enriquecen enormemente a Python, y son el elemento diferencial que influye en su crecimiento, mucho mayor que el de sus competidores.

El principal inconveniente que se le puede achacar a Python es el tener una velocidad de ejecución menor a la de los lenguajes compilados. Es por ello, por lo que muchas librerías están solucionando este inconveniente llevándose, de forma invisible para el programador, los cálculos pesados al lenguaje C. De este forma, se combina la velocidad de un lenguaje compilado y de más bajo nivel como es C, con la facilidad y legibilidad de Python.

La versión de Python utilizada en este proyecto ha sido la **3.6.0**. Esta era la última versión estable de Python al inicio de este trabajo. Sin embargo, en Marzo de 2017 se lanzó la versión 3.6.1, pero no existían cambios importantes como para plantearse la migración del sistema a la nueva versión. De cara a futuras migraciones a versiones superiores de Python es importante que tengamos presente que no dependemos únicamente de nosotros mismos, pues se hace uso de librerías de desarrolladores externos que también deben soportar esas nuevas versiones. Este último, es un punto muy importante que debe ser comentado. Para la implementación del sistema propuesto en este trabajo, se ha hecho uso de varias librerías de código abierto, realizadas por la comunidad de desarrolladores de Python. Estas librerías nos han permitido realizar sin esfuerzo tareas relacionadas con el tratamiento de la señal, la realización de cálculos numéricos complejos o el uso de algoritmos de aprendizaje supervisado. A continuación se describirán estas librerías.

Librosa

Librosa es una librería para Python centrada en el análisis de música y audio. Proporciona los principales componentes para crear sistemas de extracción de información

musical. Fue presentada en 2015 [McFee et al., 2015] por investigadores del grupo de investigación **LabROSA: Laboratory for the Recognition and Organization of Speech and Audio** de la Universidad de Columbia.

Librosa implementa una gran cantidad de funcionalidades para los sistemas MIR, separadas en varios submódulos:

- **librosa.beat:** Incluye funciones para la estimación del tempo y detección de pulsos.
- **librosa.core:** Incluye funciones para cargar audios, calcular varios tipos de representaciones frecuenciales, y una gran variedad de herramientas usadas en el análisis musical
- **librosa.decompose:** Este módulo contiene funciones para la descomposición del espectrograma y la separación harmónica-percusiva de los audios.
- **librosa.display:** Contiene métodos para la visualización de los resultados
- **librosa.effects:** Permite realizar distintos procesamientos en el dominio del tiempo como la modificación del tono o el time stretching.
- **librosa.feature:** Este módulo contiene varios tipos de funciones para la extracción de características a partir de las señales de audio. Ejemplos de estas características extraídas son los cromagramas, los MFCC o los espectrogramas MEL.
- **librosa.filters:** Permite la generación de bancos de filtros, que son usados por otros módulos de librosa.
- **librosa.onset:** Posee métodos para la detección del inicio de las notas.
- **librosa.output:** Contiene distintos métodos para la salida de ficheros de texto y de audio.
- **librosa.segment**: Incluye funciones para la segmentación estructural de las señales.
- **librosa.util**: Está formado por un conjunto de utilidades que no encajarían en ninguno de los demás módulos.

Scipy stack

SciPy es un ecosistema, basado en Python, de software de código abierto para aplicaciones relacionadas con las matemáticas, la ciencia y la ingeniería. Posee una gran cantidad de paquetes de los que destacaremos a continuación algunos, por ser de gran utilidad para nuestro proyecto.

- **NumPy**: Es el paquete fundamental de Python para la computación numérica. Tiene tipos de datos propios para los arrays multidimensionales y define operaciones entre ellos. Estas operaciones están muy optimizadas, realizando parte de

los cálculos en C. Los arrays que usaremos en nuestro sistema serán del tipo definido por Numpy permitiéndonos así realizar operaciones entre ellos a una gran velocidad.

- **Librería SciPy:** Contiene una colección de algoritmos numéricos y varios conjuntos de herramientas centradas en ámbitos diferentes como el procesamiento de señal, la optimización o la estadística.
- **Matplotlib:** Es la librería de visualización para la generación de gráficos más importante de Python. Presenta una sintaxis que recuerda a la libería de generación de gráficos de Matlab. Aunque puede realizar algunas representaciones en 3D, se suele usar principalmente para gráficos 2D
- **IPython:** Es un shell interactivo que añade nuevas funcionalidades al incluido por defecto con Python. Algunas de estas funcionalidades son el resaltado de la sintaxis o los errores mediante colores o el autocompletado de variables y atributos. Además es el núcleo de los Jupyter Notebooks de Python, aplicaciones web que permiten crear y compartir documentos que contiene código ejecutable, ecuaciones o visualizaciones.
- **Scikits**: Son un conjunto de paquetes que constituyen la base para otras librerías con funcionalidades más específicas entre las que destaca Scikit-Learn, que veremos a continuación.

Scikit-Learn

Scikit-Learn [Pedregosa et al., 2011] es una librería que nace en 2007 como parte del evento **Google Summer of Code**. Contiene un conjunto de herramientas simples y efectivas para minería y análisis de datos. Está basada en Numpy, Scipy y Matplotlib. Es de código abierto, y además puede ser usada para fines comerciales. Los tipos principales de problemas que resuelve se pueden separar en los siguientes grupos:

- **Clasificación:** Permite identificar a qué categoría pertenece un objeto.
- **Regresión:** Permite predecir un atributo continuo asociado con un objeto.
- **Clustering:** Permite la agrupación automática de objetos similares.
- **Reducción de la dimensionalidad:** Permite reducir el número de variables a considerar en un problema concreto.
- **Selección de modelos:** Permite comparar, validad y elegir parámetros y modelos.
- **Preprocesado:** Permite llevar a cabo tareas de extracción de características y normalización.

5.1.2. Latex

De cara a la visualización de los resultados de las distintas ejecuciones del programa, se ha utilizado el sistema de composición de textos LaTeX, el standard más utilizado para la publicación de textos científicos. LaTeX no es más que un paquete de macros del lenguaje TeX creado por Leslie Lamport. En TeX, los documentos se crean a partir de la compilación de ficheros que contienen instrucciones sobre cómo queremos que se vean nuestros documentos, y qué contenido deben tener. Esta forma de trabajar nos permite centrar la atención en el contenido, y no en la forma.

LaTeX permite escribir documentos TeX con una sintaxis de alto nivel. Gracias a LaTeX hemos podido generar dinámicamente ficheros, que son automáticamente exportados a PDF, con los resultados de la ejecución del programa. Hemos podido generar y compilar este ficheros desde nuestro sistema, gracias a la libería **pylatex**[1]

5.2. Esquema general del sistema

En la figura 5.2 podemos ver el esquema general del sistema. En este esquema aparecen todos los posibles componentes, aunque más adelante veremos que no siempre se utilizan todos.

5.2.1. Configuración

Mediante el archivo **conf.json** podemos controlar los parámetros de todas las partes de nuestro sistema. De esta forma, todo estará centralizado en un mismo archivo, y elegir una configuración u otra para el sistema será una tarea muy simple. JSON (JavaScript Object Notation) es un formato para el intercambios de datos. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído y escrito por cualquier lenguaje de programación. A continuación, podemos ver un fragmento del contenido de este archivo.

```
{
  "adaptation": {
    "hpss": {
      "using": true,
      "kernel_size":31,
      "margin":1
    },
    "filter":{
      "using":false,
      "freq":6000
    }
  }
}
```

[1] https://jeltef.github.io/PyLaTeX/current/

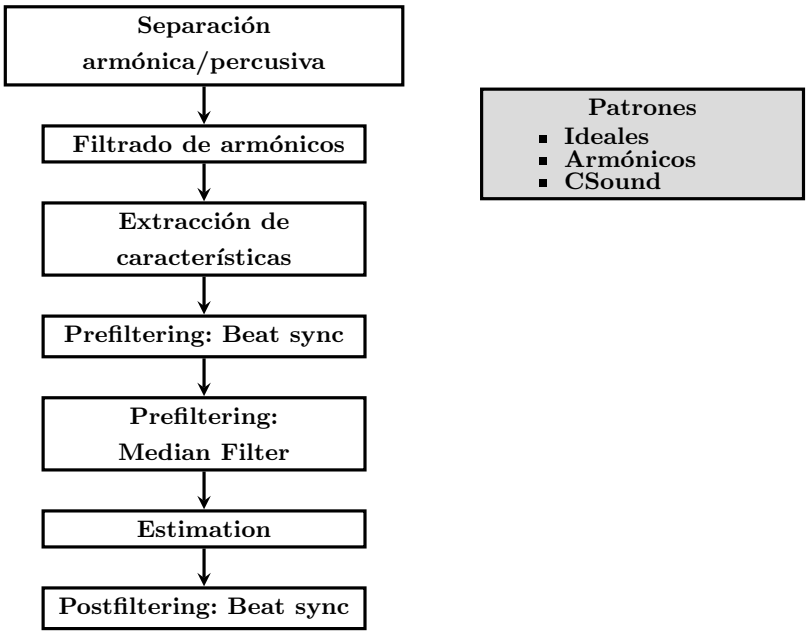


Figura 5.2: Componentes del sistema ACE implementado

5.2.2. Estructura del proyecto

En la figura 5.3 vemos la estructura del proyecto, dividido en 4 carpetas.

- **data**: Contiene 4 subcarpetas: artists, csound, patterns y tests.
  - artists: Contiene todos los datasets.
  - csound: Contiene todos los acordes generados con csound.
  - patterns: Contiene todos los templates de cromagramas de acordes.
  - tests: Contiene los archivos *.lab* generados a partir de la estimación.
- **docs**: Contiene la documentación del proyecto.
- **experiments**: Contiene los archivos de código utilizados para generar todos los ejemplos mostrados a lo largo de esta memoria.
- **pychordestimation**: Contiene el código del propio sistema de estimación de acordes.
  - **chordestimation.py**: Contiene el código principal que realiza la estimación.
  - **core.py**: Incluye algunas de las funciones más importantes usadas por los distintos componentes del sistema.

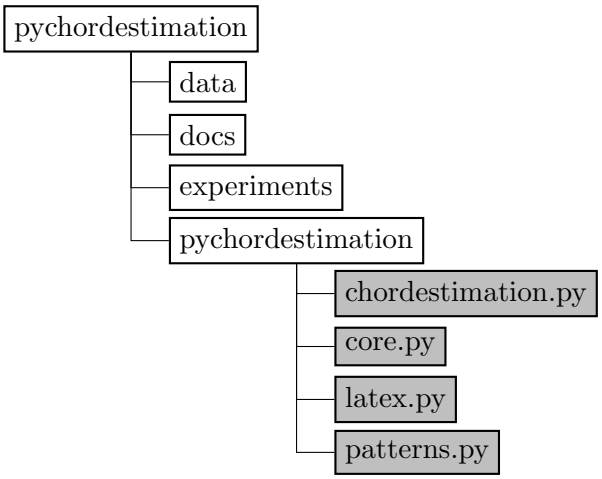


Figura 5.3: Estructura del proyecto

- **patterns.py**: En este archivo se encuentras las funciones relacionadas con la generación de patrones, su carga a partir de un fichero o su almacenamiento en disco.
- **latex**: Contiene todas las funciones para la generación del documento en latex con los resultados de la ejecución.

5.3. Adaptación de la señal

Como hemos visto, esta es una fase previa a la extracción de características cuyo propósito es eliminar de la pista de audio toda la información que no nos ayude o incluso que pueda ser perjudicial para la capacidad de estimación del sistema.

5.3.1. Separación armónica percusiva

Lo primero que realiza nuestro sistema es separar la parte armónica de la parte percusiva. Para realizar esta separación se ha utilizado la función **librosa.effects.harmonic()** de la librería **librosa**. En la figura 5.4 vemos las tres tareas principales que realiza esta función.

El principal elemento de este diagrama es el llamado **HPSS**. Este es el bloque que realmente realiza la separación armónica y percusiva. La implementación que realiza librosa de este algoritmo está basada en los dos siguientes trabajos: [Fitzgerald, 2010] y [Driedger et al., 2014]. Esta implementación se basa en la aplicación de 2 filtros de mediana al espectro para obtener las componentes armónicas y percusivas. Se usa un filtro de mediana horizontal para obtener las componentes armónicas, y un filtro de mediana vertical para obtener las componentes percusivas.

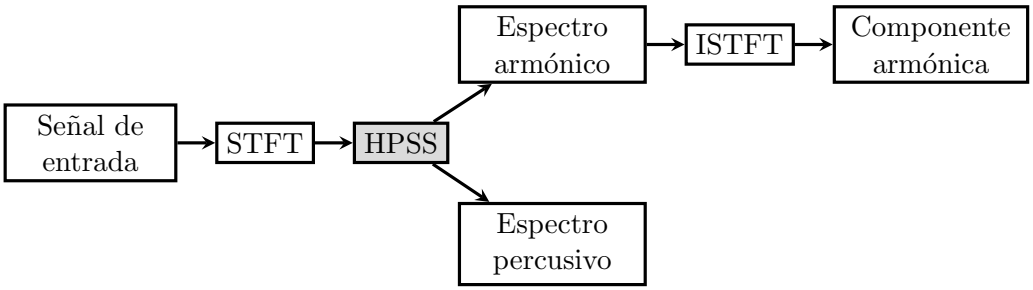


Figura 5.4: Diagrama de flujo de la función *harmonic()*

Existen dos parámetros de la separación armónica/percusiva que podemos controlar por medio del archivo de configuración del sistema:

- **kernel_size**: Especifica el tamaño de la máscara usada en los filtros de mediana. Por defecto es 31 para ambos. Se puede asignar un tamaño distinto para la máscara del filtro que obtiene la parte percusiva y para la máscara del filtro que obtiene la parte armónica.
- **margin**: Este parámetro, definido en [Driedger et al., 2014], nos permite obtener, además de las componentes armónicas y percusivas, una componente residual para los sonidos que no pueden ser claramente clasificados como armónicos o como percusivos. Cuando el margen sea igual a 1, no existirá componente residual. Según aumente este margen, el componente residual será mayor, quedando en los demás componentes sólo los sonidos más puramente armónicos o percusivos.

Para estar en consonancia con la filosofía modular del proyecto, el usuario puede seleccionar que no se realice esta tarea y que se pase directamente al filtrado de armónicos o a la extracción de características.

5.3.2. Filtrado de armónicos

Otro componente que puede ser utilizado en la fase de adaptación de la señal es el filtrado de armónicos. El filtrado de armónicos realiza un filtro paso bajo a la señal para quitar los armónicos de la parte alta del espectro que puedan entorpecer la estimación de los acordes. El sistema usa un **filtro Butterworth de orden 1**. La frecuencia a la que se realiza el filtrado es elegida por el usuario en el archivo de configuración. En la figura 5.5 vemos un ejemplo del espectro de un filtro Butterworth.

Igual que la separación armónica/percusiva, el filtrado de armónicos es opcional y el usuario puede elegir si quiere realizarlo o si quiere pasar directamente a la extracción de características.

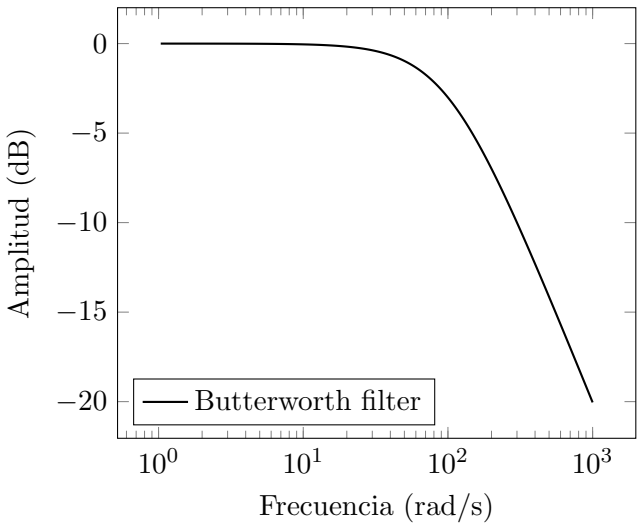


Figura 5.5: Espectro de filtro Butterworth de orden 1

5.4. Extracción de características

En la fase de extracción de características se obtendrán los **cromagramas** que posteriormente se compararán con los patrones de acordes para realizar la estimación. La librería **librosa**, proporciona 3 implementaciones distintas de cromagramas: **stft**, **cqt** y **cens**. Se ha dado la opción al usuario de elegir cuál de las 3 quiere usar. Existen algunos parámetros adicionales que el usuario puede elegir en el archivo de configuración del sistema, para el cálculo del cromagrama:

- **hop_length**: Especifica el número de muestras entre cromagramas sucesivos.
- **tuning**: Esta opción permite indicar, en los casos en los que se sepa de antemano, la desviación de la afinación de la pieza musical, con respecto a la afinación estándar A440. En el caso del cromagrama calculado con la STFT, esta desviación es calculada automáticamente.

A continuación, se detallan las 3 implementaciones de cálculo de cromagramas que contiene **librosa**. En la figura 5.6 podemos ver un ejemplo de cada tipo.

5.4.1. stft

El cromagrama se obtiene a partir de la Transformada de Fourier de Tiempo Reducido. La implementación está basada en el artículo [Ellis, 2007a].

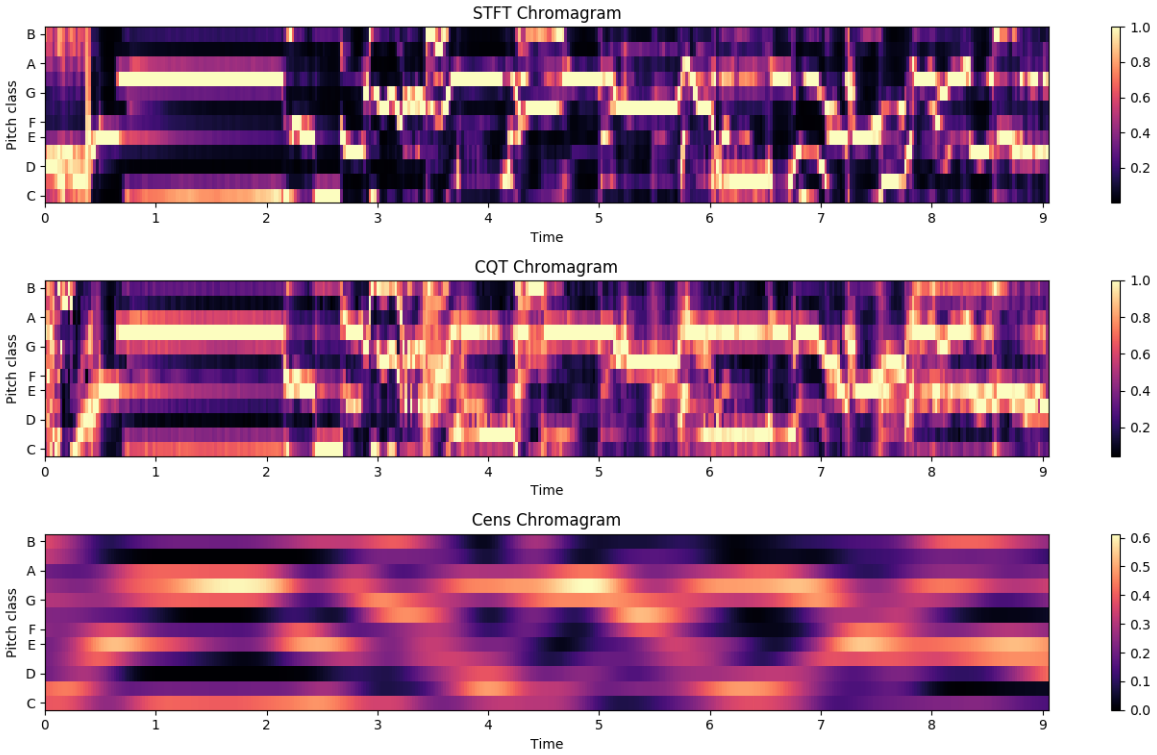


Figura 5.6: Diferentes tipos de cromagramas extraídos de un mismo fragmento de audio

5.4.2. cqt

En este caso, en vez de la STFT, se usa la Transformada de Q Constante. La CQT realiza una transformación más parecida a la que realiza el oído humano. En bajas frecuencias, somos capaces de distinguir frecuencias muy cercanas, mientras que en altas frecuencias, la mínima distancia entre frecuencias que podemos captar es mucho mayor. Esto se traduce, en la CQT, en una resolución espectral muy alta en bajas frecuencias, que va disminuyendo según aumentamos la frecuencia.

5.4.3. cens

Esta implementación calcula la variante del cromagrama *CENS: Chroma Energy Normalized* [Müller and Ewert, 2011]. Entre otras funciones, incorpora un suavizado del mismo.

5.5. Prefiltering

En esta apartado, el sistema puede realizar 2 posibles tareas: **filtrado del cromagrama** y **sincronización del cromagrama con el pulso de la canción (beats)**. El usuario deberá escoger una u otra. Si realizamos una sincronización del cromagrama con el pulso, sería perjudicial realizar además un filtro de mediana con cromagramas de varios pulsos distintos. Por otra parte, si realizamos el filtrado del cromagrama primero, no tendría sentido volver a aplicar un filtro para obtener un solo vector de croma por beat.

5.5.1. Filtrado del cromagrama

En primer lugar, explicaremos el filtrado del cromagrama para el cuál se hace uso de un **filtro de mediana**. El usuario podrá, desde el archivo de configuración, elegir un tamaño para la máscara del filtro. Esta tarea supone, simplemente, un suavizado del espectrograma, pero no soluciona el problema de tener varias detecciones de acordes distintas en cada pulso, por lo que habrá que combinarla con una función en postfiltering que nos deje una única detección por beat. Para realizar los filtros de mediana se ha usado la función **median()** que proporciona **numpy**.

5.5.2. Detección de beats

Antes de pasar a explicar cómo se realiza la sincronización del cromagrama con los beats, hay que aclarar cómo se obtienen estos. La implementación que se ha utilizado es la de la función **beat_track()** de la librería **librosa**. Esta implementación está basada en el siguiente artículo: [Ellis, 2007b]. El método seguido en dicho artículo tiene tres partes claramente diferenciadas como vemos en la figura 5.7

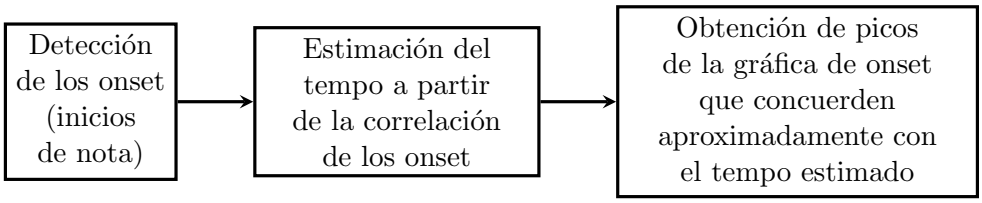


Figura 5.7: Diagrama de flujo de la obtención de los beats

5.5.3. Sincronización con beats

Librosa también nos permite realizar la sincronización con los beats sin apenas esfuerzo mediante la función **librosa.util.sync()**. La detección de beats se ha realizado de la forma descrita en la sección 5.5.2. El objetivo final es tener 1 sola detección entre cada par de beats.

Se ha detectado que, aunque no es común, existen ocasiones en las que se producen cambios de acorde a mitad de un beat. Es por ello por lo que se ha dado al usuario la posibilidad de subdividir los beats, de forma que tendremos 2 detecciones, y no una, entre cada par de beats.

5.6. Patrones

Se han implementado varios tipos de patrones de cromagramas acordes que han sido utilizados como conjunto de entrenamiento para el algoritmo.

5.6.1. Patrones ideales

El tipo de patrón más básico que se ha implementado en el sistema es el ideal. Como se ha visto anteriormente, este tipo de patrón es un patrón binario de 12 elementos donde tenemos ceros en todas las posiciones, menos en las posiciones correspondientes a las notas del acorde, en las cuáles tendremos unos. Se han generado patrones ideales para los siguientes tipos de acordes:

- **Tríadas mayores**
- **Tríadas menores**
- **Acordes de séptima dominante**
- **Acordes de séptima mayor**
- **Acordes de séptima menor**

Hemos generado 12 patrones ideales para cada tipo de acorde (uno por cada nota). Tras su generación, estos han sido almacenados en ficheros **csv**, para evitar tener que volver a

generarlos en cada ejecución del programa. En la figura 5.8 vemos los pesos de cada nota en los 5 tipos de patrones implementados, utilizando la nota **Do** como nota fundamental del acorde.

5.6.2. Patrones ideales con armónicos

De la misma forma, también se han generado patrones ideales con armónicos. El modelo de armónicos elegido supone un decrecimiento exponencial de la energía de los armónicos con $\alpha = 0{,}5$. Se han supuesto 6 armónicos. El proceso para obtener los patrones con armónicos se ha explicado en el apartado 4.4.2

El vector de croma generado para cada acorde se ha normalizado, y por lo tanto solo tiene valores entre 0 y 1. Igual que antes, en la figura 5.9 vemos los pesos de cada nota en los 5 tipos de patrones implementados, utilizando la nota **Do** como nota fundamental del acorde.

5.6.3. Patrones a partir de soundfonts

Basándonos en [Gil Ortiz, 2016], también se ha añadido al sistema la posibilidad de usar patrones de acordes sintetizados mediante **SoundFonts**. Los SoundFonts son archivos que contienen muestras de audio de diversos instrumentos musicales. Estas muestras suelen proceder de grabaciones reales del instrumento. Los sintetizadores de muestreo son capaces de generar ficheros de audio a partir de las notas existentes en ficheros MIDI. Estos ficheros contienen un conjunto de instrucciones que indican (a otros dispositivos capaces de comprender el protocolo MIDI) cómo se debe interpretar una obra. Generalmente, los paquetes de SoundFonts contienen muestras grabaas de todos los instrumento de **General MIDI**. General MIDI es una especificación para sintetizadores que exige una serie de requisitos adicionales a los del estándar MIDI. Entre ellos, se incluye una numeración definida para cada instrumento. En el estándar General MIDI se definen 128 instrumentos.

Los archivos de audio de los acordes generados a partir de soundfonts, utilizados en este trabajo, han sido proporcionados por el tutor del mismo, José Manuel Iñesta. Estos archivos han sido sintetizados mediante el software **CSound**. En la figura 5.10 podemos ver un esquema de la generación de estos. Como vemos, el programa, además de devolver los ficheros de audio, devuelve un fichero **csv** con información de los tiempos de inicio y final de cada acorde. Por lo tanto, podremos calcular los cromagramas de todos los acordes y almacenarlos para usarlos posteriormente como patrones.

En este trabajo se han utilizado 3 soundfonts distintos: **fluid**, **muse** y **tim**, así podremos comprobar cómo afecta al porcentaje de acierto de nuestro sistema el utilizar unos u otros soundfonts.

5.7. Estimación

La ventaja que supone en esta fase el uso de la librería **sklearn** es que, cambiando apenas una línea de código, podemos usar distintos clasificadores. Por lo tanto, hemos

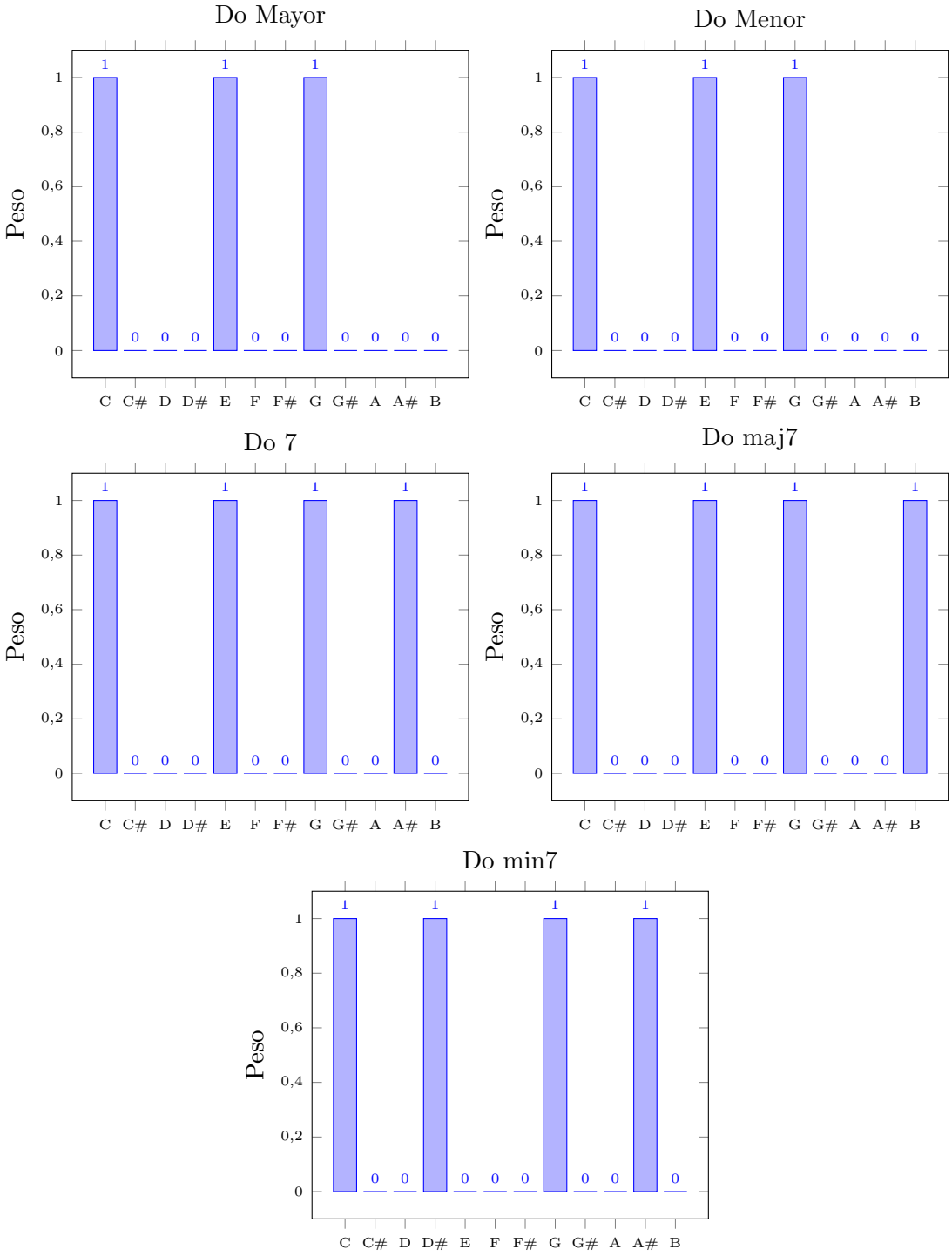


Figura 5.8: Pesos de cada nota en los 5 patrones ideales implementados

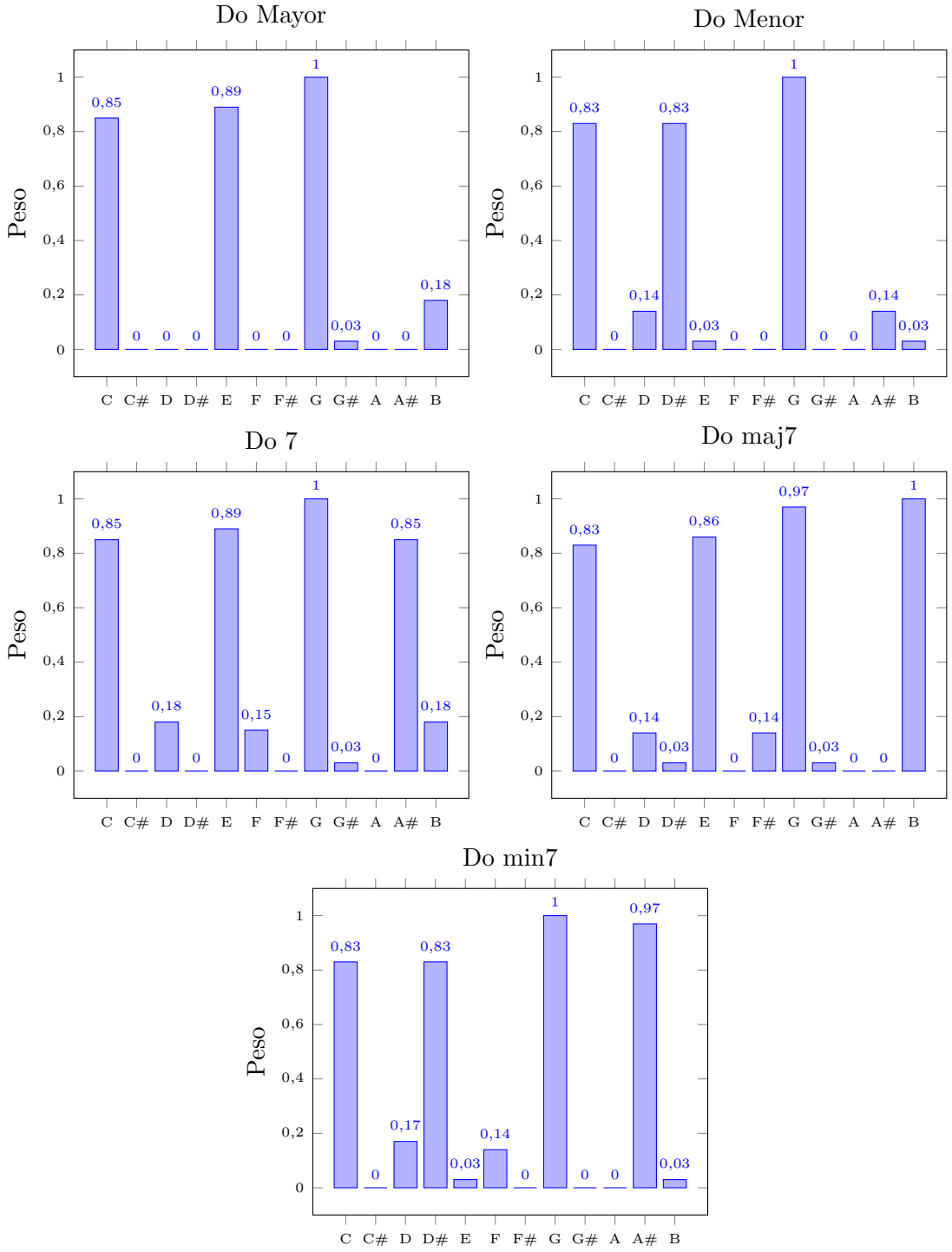


Figura 5.9: Pesos (normalizados) de cada nota en los 5 patrones con armónicos implementados

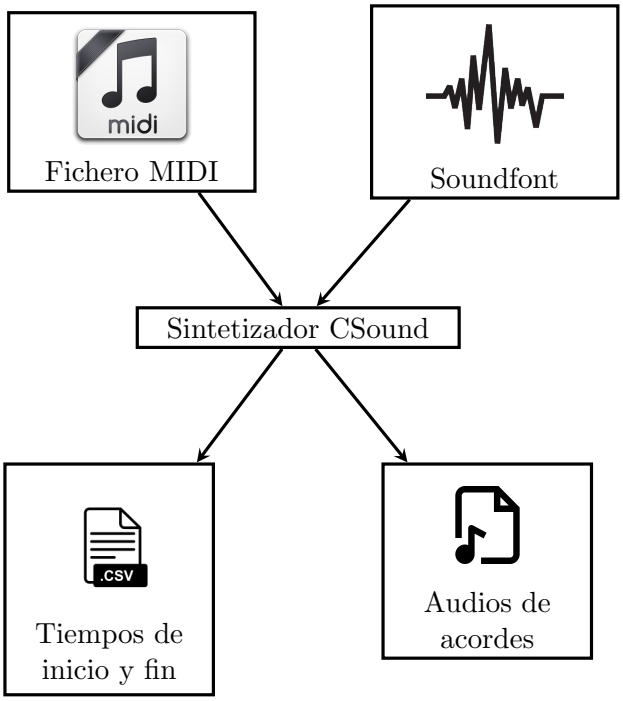


Figura 5.10: Diagrama de generación de patrones a partir de SoundFonts

sido capaces de implementar 3 clasificadores para el sistema.

5.7.1. KNN

Usaremos un clasificador de tipo KNN con $k = 1$ para los patrones ideales, ya sean con o sin armónicos. Usar un k mayor, en este caso, no tendría ninguna utilidad, pues solo tenemos un patrón de entrenamiento por clase (un único vector de croma por nota y acorde). Sin embargo, en el caso de los patrones de CSound, tenemos varios posibles patrones para cada acorde, por lo que realizaremos un estudio para obtener el k que maximiza la puntuación de la estimación. Con el fin de eliminar posibles muestras ruidosas de nuestro dataset de patrones obtenidos con CSound, se implementará un algoritmo de **selección de prototipos**, **Editing Nearest Neighbour (ENN)** [Kuncheva and Jain, 1999]. Para aplicar este algoritmo, se extrae cada vez una muestra del conjunto de entrenamiento. Si esa muestra no se clasifica correctamente, mediante un knn con el conjunto de datos modificado (no contiene a la muestra analizada), esa muestra será extraída del conjunto de datos de entrenamiento final.

5.7.2. Support Vector Machine

Otro posible clasificador que podrá ser utilizado por el sistema, es una **SVM**. Esta SVM usará un kernel de tipo **RBF**, que le permitirá trabajar con datos que no son linealmente separables. Contará con un parámetro configurable por parte del usuario:

- **C**: Este parámetro permite controlar si queremos ajustarnos más a los datos o si, por el contrario buscamos obtener una superficie de decisión más suave. Cuanto más alto sea el nivel de C, más tratará la SVM de clasificar correctamente absolutamente todas las muestras, dando al modelo libertad para seleccionar más muestras como vectores de soporte.

5.7.3. Red neuronal

Por último, también es posible seleccionar una red neuronal como clasificador. Se ha utilizado la clase **MLPClassifier** de **sklearn** para crear un perceptrón multicapa. Para entrenar el perceptrón, sklearn usa un algoritmo de descenso de gradiente. Los gradientes se calculan mediante **backpropagation**. Existen dos posibles parámetros relacionados con la red neuronal, que el usuario podrá configurar en el sistema:

- **hidden_layers**: Especifica el número de capas ocultas de la red, y el tamaño de cada una.
- **alpha**: Es el parámetro de regularización. El sobreajuste de la red se podrá evitar mediante valores altos de alfa. De la misma forma, disminuir el valor de alpha, permitirá que la red se adapte mejor a los datos de entrenamiento.

5.8. Postfiltering: Sincronización con beat

Tras la estimación, aún es posible retocar los resultados proporcionados por el clasificador con el fin de incorporar nueva información al sistema que mejore el porcentaje de acierto final. En el sistema, hemos implementado la función **one_label_per_beat()**. Esta función es prácticamente obligatoria si no hemos realizado la sincronización con el beat en la fase de **prefiltering**. De hecho, el sistema la ejecutará automáticamente (aunque el usuario no la haya activado para la ejecución) cuando dicha función no haya sido ejecutada. Esto es así porque, si el programa a la salida proporciona una gran cantidad de cambios de acorde por beat, no tendrá utilidad ninguna para un posible usuario.

La forma de realizar la sincronización con el beat en postfiltering es más sencilla que la de la fase de prefiltering. En este punto, la información que tenemos ya no son muestras de audio ni cromagramas, son simplemente cadenas de caracteres con el nombre del acorde detectado. Además, como es obvio, también tenemos los tiempos en los que se producen esos acordes. Con esa información, tendremos que obtener todas las detecciones realizadas entre cada par de beats y escoger la que más se repite, es decir, la moda. De la misma forma que pasaba en la fase de **prefiltering**, en **postfiltering** el usuario también podrá elegir, mediante el campo **subdivide**, si quiere 1 o 2 detecciones entre cada par de beats.

5.9. Evaluación

Para la evaluación del sistema se ha usado la librería **mir_eval**[2], de la cual hablamos en la sección 4.7. De todas las métricas que se nombran en dicha sección, usaremos las siguientes: **root**,**majmin** y **mirex**, por ser las que más se adecúan al nivel de profundidad conseguido en nuestro desarrollo. No nos aportaría nada de valor obtener una métrica relacionada con las séptimas o las inversiones, cuando nuestro sistema no tiene la capacidad de obtener estas características. En la figura 5.11 vemos el esquema de la evaluación. La librería **mir_eval** se comporta como una *caja negra* a la que únicamente tendremos que pasarle el archivo *.lab* que hemos generado y el de referencia, y nos devolverá las tres puntuaciones citadas anteriormente.

5.9.1. Datasets

Para evaluar el sistema se han usado 3 datasets distintos, que serán detallados a continuación.

The Beatles

Este dataset contiene gran parte de la discografía de The Beatles. Para muchos, este es el dataset más importante para la estimación de acordes. Fue presentado en 2010, en la tesis doctoral del investigador Chris Harte [Harte, 2010], perteneciente al grupo

[2] `https://github.com/craffel/mir_eval`

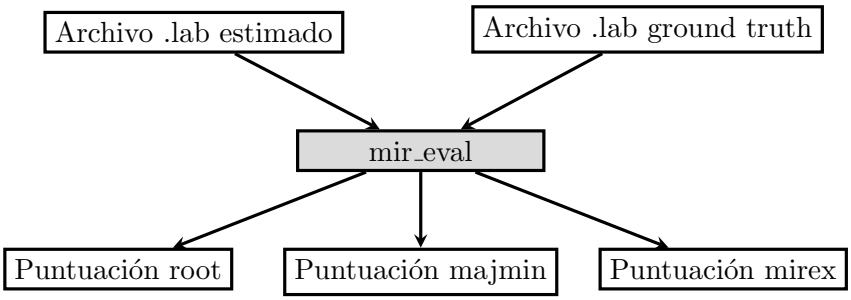


Figura 5.11: Sistema de evaluación de los resultados

Tabla 5.1: Discos del dataset de The Beatles junto con su duración

Disco	Album	Duración (mm:ss)
01	Please Please Me	32:45
02	With the Beatles	33:24
03	A Hard Day's Night	30:30
04	Beatles for Sale	34:13
05	Help!	34:21
06	Rubber Soul	35:48
07	Revolver	34:59
08	Sgt. Pepper's Lonely Hearts Club Band	39:50
09	Magical Mystery tour	36:49
10CD1	The Beatles	46:21
10CD2	The Beatles	47:14
11	Abbey Road	47:24
12	Let It Be	35:10

Isophonics[3], de la Universidad Queen Mary de Londres. Aunque posteriormente se han presentado diversos datasets, algunos de ellos con una cantidad mucho mayor de canciones, este tiene una gran ventaja, y es que ha sido validado por una gran cantidad de músicos y miembros de la comunidad MIR. Hasta hace unos años, era el dataset de referencia para la estimación de acordes, sobre el que se basaban todos los artículos que se publicaban con sistemas ACE, y el único que se utilizaba en **MIREX**. Esto provocó que se produjera un cierto sobreajuste de los algoritmos, y que estos vieran disminuida su eficacia al generalizar con nuevos datasets. En la tabla 5.1 vemos todos los discos que se incluyen en este dataset.

[3] `http://isophonics.net`

Queen

Este dataset, mucho más reducido que el anterior, pertenece también a **Isophonics**, y contiene los 3 discos de Gratest Hits del grupo Queen. Podemos encontrarlo en la página web de Isophonics[4]. En la propia página de Isophonics advierten que, aunque se puede usar para medir la eficacia de nuestros algoritmos de estimación de acordes, este dataset no tiene el nivel de precisión del dataset de The Beatles, por lo que debe usarse con cierta precaución.

Robbie Williams

Este dataset fue incorporado a **MIREX** en el año 2016, y consta de 5 discos del artista Robbie Williams. Se puede encontrar en la web del **Image and Sound Processing Group** de la Universidad Politécnica de Milán[5] y fue presentado en el siguiente artículo [Di Giorgi et al., 2013].

5.10. Visualización de resultados

Para poder visualizar de forma más clara los resultados de una estimación de una o varias canciones, se ha implementado la posibilidad de generar un documento resumen con todos los resultados. El documento se ha generado en LaTeX, mediante la librería **pylatex**[6]. En la figura 5.12 vemos un fragmento del documento generado tras el análisis de un dataset por parte del programa. La información que nos proporciona el fichero (no toda se ve en la imagen) es:

- **Parámetros y algoritmos usados en todas las fases de la estimación**
- **Tiempo total de ejecución**
- **Puntuaciones para cada canción**
- **Puntuación media para cada disco**
- **Puntuación media para cada artista**
- **Puntuación media total de todos los artistas analizados**

El análisis del contenido de este fichero ha sido de gran utilidad para detectar problemas que hacían que la puntuación de algunas canciones fuera muy inferior a la media. Además, también ha sido utilizado para comparar las puntuaciones obtenidas en función de las características y parámetros utilizados.

En la siguiente URL: `https://github.com/unmonoqueteclea/pychordestimation/blob/master/data/tests/Reports/Example%20report.pdf` podemos ver un ejemplo de un fichero generado tras una ejecución del sistema.

[4] `http://isophonics.net/content/reference-annotations-queen`
[5] `http://ispg.deib.polimi.it/mir-software.html`
[6] `https://jeltef.github.io/PyLaTeX/latest/`

Resultados de la ejecución

Pablo González Carrizo

February 20, 2017

1 Info

Test realizado con la siguiente configuración

- Se ha separado la parte armónica de la percusiva antes de obtener el croamgrama.
- El tipo de cromagrama utilizado ha sido: cqt
- Se ha promediado el cromagrama en función de los beats detectados.
- Se ha utilizado la técnica de semejanza a patrones ideales.
- Los tipos de acorde usados han sido :maj, min
- Los patrones ideales usados NO contienen armónicos

2 Ejecución

El programa ha tardado: 57 minutos y 39 segundos en ejecutarse

3 Artistas analizados

3.1 Beatles

3.1.1 01_-_Please_Please_Me

Estas son las canciones analizadas

- 07 - Please Please Me.wav - root: 60.1095583123% majmin: 39.3076197915% mirex: 39.3076197915%
- 05 - Boys.wav - root: 54.5515429165% majmin: 30.5179656885% mirex: 30.5179656885%
- 02 - Misery.wav - root: 73.6533026196% majmin: 60.5349010761% mirex: 60.5349010761%

Figura 5.12: Fragmento del archivo generado por el sistema implementado

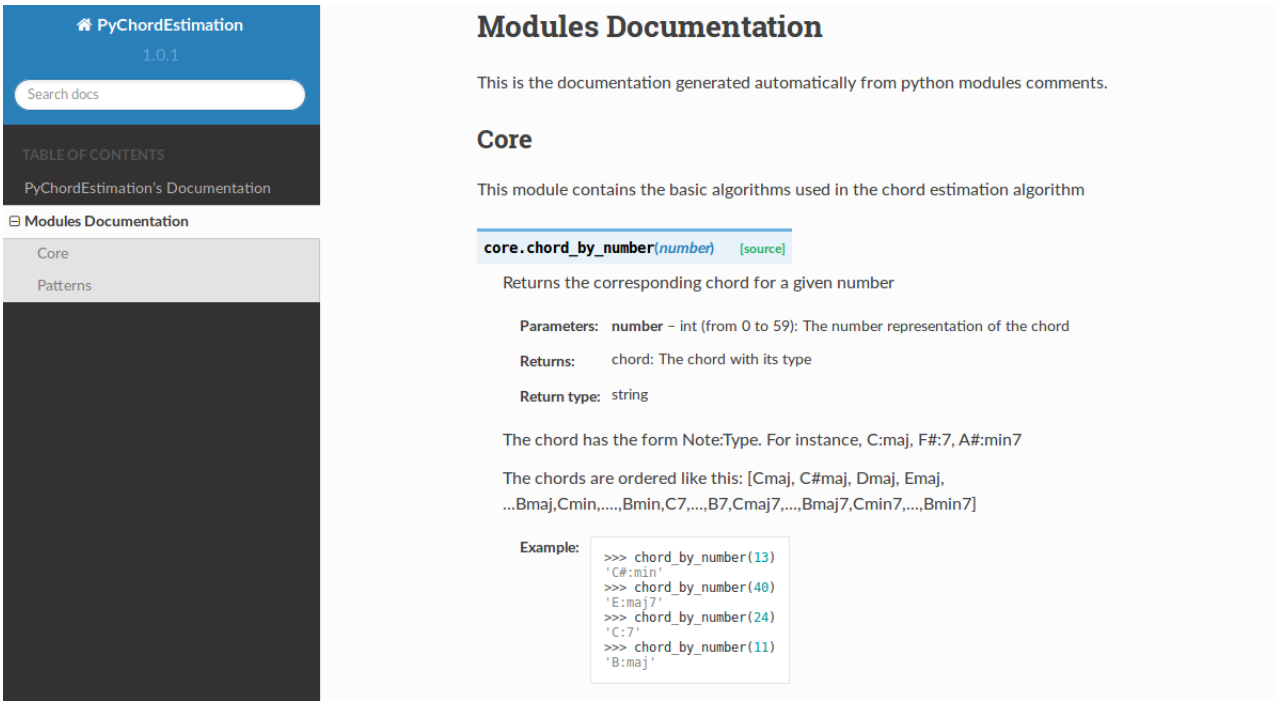


Figura 5.13: Documentación auto-generada por Sphinx

5.11. Documentación

Con el fin de facilitar el trabajo a otros investigadores que decidieran usar este proyecto como base para sus implementaciones, se ha generado una documentación para el proyecto. Para ello se ha hecho uso de la herrramienta **Sphinx**[7] que nos ha permitido convertir los comentarios del propio código en una documentación estructurada, como se muestra en la figura 5.13 Es posible ver esta documentación descargando la siguiente carpeta: `https://github.com/unmonoqueteclea/pychordestimation/tree/master/docs/build/html` y abriendo el archivo **main.html**

5.12. Repositorio de código

Todo el código del proyecto se ha subido a un repositorio público en la siguiente URL: `https://github.com/unmonoqueteclea/pychordestimation`. Desde allí, podrá ser descargado y ejecutado por cualquiera, una vez haya instalado las librerías comentadas anteriormente. Por motivos de propiedad intelectual, los audios de los datasets de entrenamiento no han sido subidos al repositorio.

[7] `http://www.sphinx-doc.org/en/stable/`

6 Evaluación del sistema

A continuación, se muestran los datos procedentes de la evaluación del sistema para los 3 posibles datasets. La forma de evaluación se ha explicado en el capítulo anterior. El sistema final se evaluará con todos los datasets, pero para la evaluación de los subsistemas, utilizaremos algunos discos concretos, pues de lo contraría llevaría demasiado tiempo. Se aplicarán todas las conclusiones extraídas de la evaluación de los subsistemas en el diseño del sistema final.

6.1. Evaluación de los subsistemas

En el capítulo anterior hemos visto que se han implementado varias técnicas distintas para cada una de las partes del sistema. Además, algunas de ellas tenían parámetros, y desconocemos cuales son los valores ideales para estos. Es por ello, por lo que, de cara a crear el sistema final de estimación de acordes, en esta sección se evaluará la eficacia de las diferentes técnicas propuestas y cómo afectan al resultado de la estimación los parámetros escogidos para cada una de ellas. Cuando se haya realizado la evaluación de todos los subsistemas ya se estará en condiciones de elegir los parámetros adecuados para el sistema final.

6.1.1. Configuración básica

Para la evaluación de los subsistemas, se ha decido partir de un sistema con los elementos *básicos*, mostrados en la figura 6.1.

El sistema incluye separación de parte armónica y parte percusiva, filtro de mediana en prefiltering, sincronización con beat en postfiltering y usa un clasificador knn con $k = 1$. Los patrones para este sistema básico son patrones binarios ideales de acordes mayores y menores. Antes de medir la eficacia de los distintos subsistemas, se ha obtenido la puntuación de esta configuración para varios discos. En la tabla 6.1, vemos las puntuaciones obtenidas.

6.1.2. Separación armónica percusiva

Como hemos visto previamente, existían dos parámetros que nos permiten controlar el funcionamiento de la separación de las partes armónica y percusiva de la señal. Estos eran el **tamaño del kernel** y el **margen**.

Comenzaremos modificando el margen para ver cómo afecta al resultado. Para ello, nos basaremos en la **configuración básica** descrita en el apartado anterior e iremos mo-

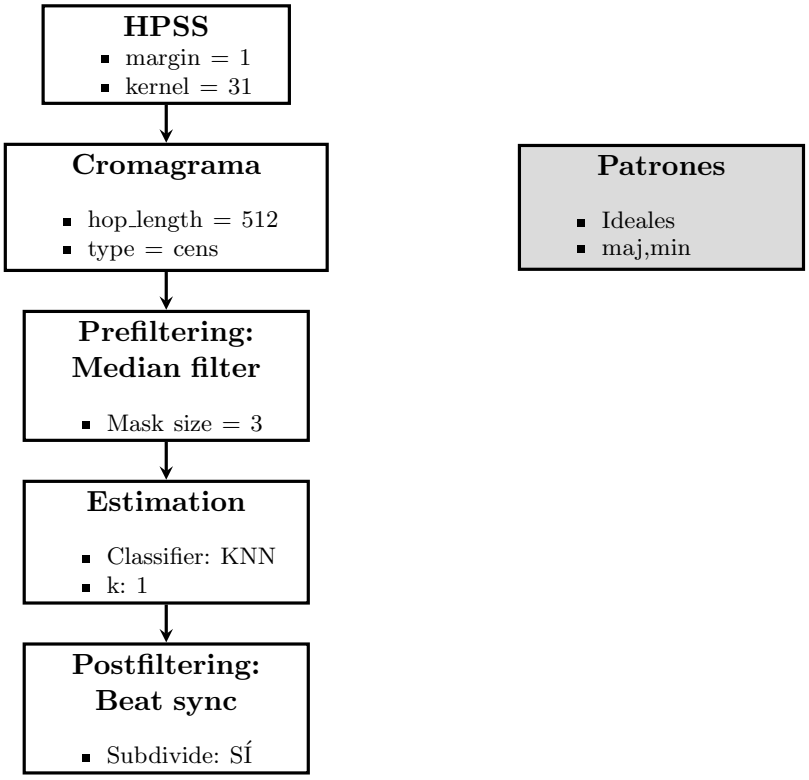


Figura 6.1: Configuración básica del sistema

Tabla 6.1: Puntuaciones de la configuración básica

Artista	Disco	root	majmin	mirex
Beatles	01	59.79	49.70	50.15
Beatles	02	60.45	46.12	45.13
Beatles	03	62.60	50.05	49.95
Beatles	04	68.99	59.75	58.84
Queen	01	48.55	38.54	38.11
Robbie Williams	01	67.65	49.65	49.63

Tabla 6.2: Puntuaciones tras modificación margen HPSS

Artista	Disco	margen	root	majmin	mirex
Beatles	01	1	59.79	49.70	50.15
Beatles	01	2	57.64	47.27	47.68
Beatles	01	3	55.37	45.26	45.67
Queen	01	1	48.55	38.54	38.11
Queen	01	2	48.27	37.00	36.68
Queen	01	3	47.01	36.03	35.75

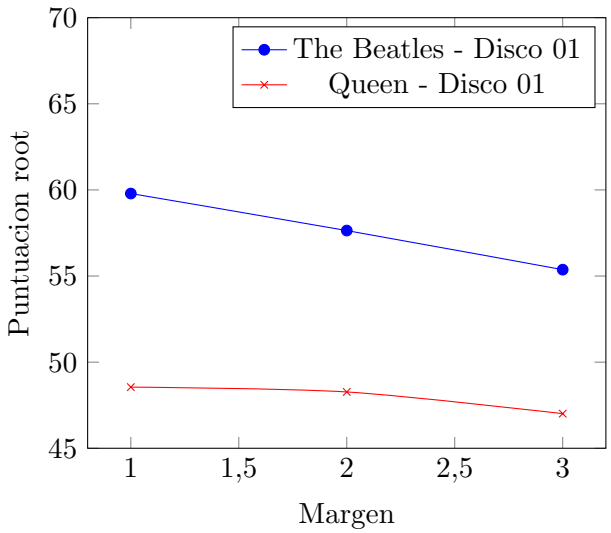


Figura 6.2: Puntuación *root* en función del margen en el HPSS

dificando progresivamente dicho parámetro. En la tabla 6.2, podemos ver los resultados obtenidos y en la figura 6.2 una representación de estos.

Podemos ver claramente que la puntuación final desciende según aumenta el margen. Este descenso es un poco menor en el disco de Queen, seguramente debido a que tiene más presencia de componentes percusivas. La conclusión que extraemos, por lo tanto, es que **el margen ideal para nuestro sistema es 1**. Sin embargo, en ciertas piezas musicales con gran cantidad de componentes percusivas podría ser beneficioso usar un margen de 2 en vez de de 1.

Ahora, pasamos a analizar como afecta a la puntuación variaciones del tamaño del **kernel** utilizado en los filtros de mediana en la función **hpss**. Para ello, a partir de la **configuración básica**, hemos fijado el valor del **margen** en 1, y hemos ido modificando progresivamente el valor del kernel. En la tabla 6.3 podemos ver los resultados obtenidos, y en la figura 6.3 vemos la representación gráfica de estos. De la gráfica podemos

Tabla 6.3: Puntuaciones tras modificación del tamaño del kernel del HPSS

Artista	Disco	margen	kernel	root	majmin	mirex
Beatles	01	1	31	59.79	49.70	50.15
Beatles	01	1	41	60.08	49.83	50.29
Beatles	01	1	61	61.44	51.01	51.47
Beatles	01	1	101	61.79	51.95	52.41
Beatles	01	1	131	61.65	52.20	52.59
Beatles	01	1	251	62.63	53.17	53.43
Beatles	01	1	351	62.71	53.37	53.63
Beatles	01	1	421	62.69	53.19	53.51
Beatles	02	1	31	60.45	46.12	45.13
Beatles	02	1	41	61.33	46.79	45.79
Beatles	02	1	61	62.50	49.17	48.11
Beatles	02	1	101	63.29	49.76	48.65
Beatles	02	1	131	63.80	50.32	49.23
Beatles	02	1	240	63.83	49.59	48.49
Beatles	02	1	351	63.51	49.36	48.25

Tabla 6.4: Puntuaciones *root* tras eliminación del bloque HPSS

Artista	Disco	root	majmin	mirex
Beatles	01	61.18	51.63	51.97
Beatles	02	61.48	46.48	45.45

deducir que **un valor alrededor de 240 será el óptimo para este parámetro**. Sin embargo, por razones de eficiencia, y puesto que el aumentar este parámetro tiene un coste computacional muy alto, **escogeremos un valor de aproximadamente 150**. Usar este valor en vez de 240, supone que la ejecución del sistema para 1 solo disco tarde aproximadamente 3 minutos menos.

Para terminar este apartado, nos faltaría ver qué pasa si eliminamos este bloque del sistema. Por lo tanto, usaremos la configuración de la figura 6.1 pero sin el bloque de **HPSS**. En la tabla 6.4 vemos los resultados obtenidos. De ella podemos extraer que **el uso de la separación armónica percusiva es beneficioso para el sistema**, pues ha mejorado la estimación alrededor de un 1.5 % en el primer disco, y sobre de un 2 % en el segundo disco (si usamos como referencia la mejor puntuación obtenida en con el bloque de separación de parte armónica y percusiva). Sin embargo: vemos algo que nos llama la atención, la puntuación final, en el caso de usar el bloque HPSS con los parámetros por defecto, es menor que la de simplemente no usarlo.

6.1.3. Filtrado de armónicos

A continuación, se va a analizar cómo afecta el filtrado de armónicos a la estimación. Nos basaremos en la **configuración básica** descrita anteriormente pero añadiremos

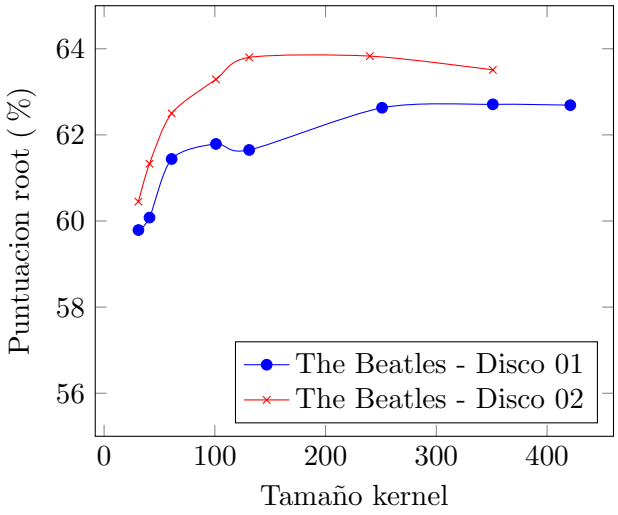


Figura 6.3: Puntuación *root* en función del tamaño del kernel en el HPSS

entre el bloque **HPSS** y el bloque de **estimación** un bloque de **filtrado de armónicos**. En la figura 6.4 vemos el diagrama de esta nueva configuración. Modificaremos de forma progresiva la frecuencia de corte del filtro paso bajo utilizado, para observar cómo afecta a la estimación. En la tabla 6.5 vemos los resultados obtenidos, y en la figura 6.5, una representación gráfica de estos. Las conclusiones que podemos interpretar de estos resultados es que **el efecto del filtrado de armónicos es bastante bajo**. Podemos ver que, en el caso del disco de Robbie Williams, prácticamente no supone ningún tipo de mejora en el resultado final. Sin embargo, en el disco de The Beatles, si que hay algo más de un 1 % de mejora, que desaparece al filtrar por encima de los 5000Hz. Esto puede ser debido a que, al ser una grabación antigua, no tiene suficiente energía en la parte alta del espectro, y por lo tanto filtrar por encima de 5000 Hz es prácticamente igual que no filtrar.

6.1.4. Extracción de características: Parámetros y tipos de cromagrama

Como hemos comentado, dentro de la fase de extracción de características había que elegir el tipo de cromagrama, y algunos de los parámetros usados para su obtención. En la tabla 6.6 vemos que, con respecto al tipo de cromagrama que debemos utilizar no cabe ninguna duda. **El tipo de cromagrama que mejor resultados nos dará es el cens**, que supone un incremento de casi un 20 % con respecto al uso del cromagrama basado en la stft. Con respecto al cromagrama cqt la diferencia no es tan grande, pero, aún así es claramente apreciable. La configuración utilizada para este apartado ha sido la **configuración básica** de la figura 6.1.

Usando la misma configuración, y fijando el tipo de cromagrama a cens, pasaremos

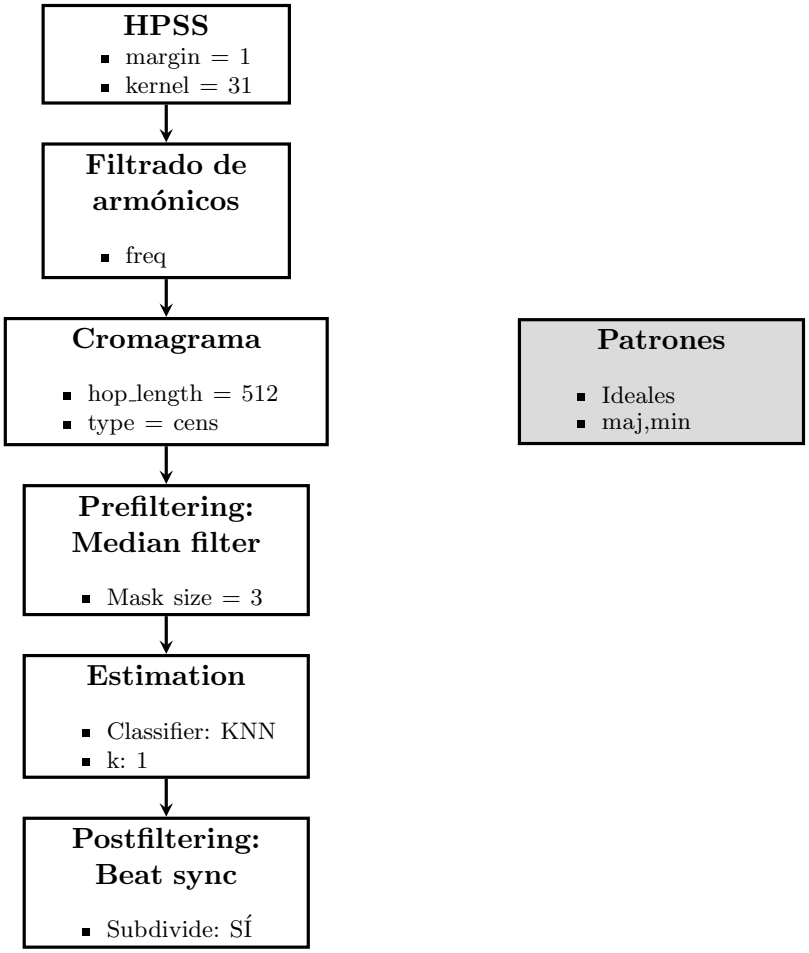


Figura 6.4: Configuración del sistema usando filtrado de armónicos

Tabla 6.5: Puntuaciones tras filtrado de armónicos

Artista	Disco	freq	root	majmin	mirex
Beatles	01	2400	60.95	50.45	50.90
Beatles	01	2600	60.76	50.23	50.70
Beatles	01	2800	60.66	50.01	50.54
Beatles	01	3000	60.44	49.95	50.38
Beatles	01	3200	60.66	50.18	50.60
Beatles	01	3500	60.41	49.99	50.44
Beatles	01	3800	60.55	50.15	50.61
Beatles	01	4000	60.42	50.16	50.62
Beatles	01	5000	60.04	49.83	50.30
Beatles	01	6000	59.94	49.79	50.25
Robbie Williams	01	2400	67.65	49.13	49.11
Robbie Williams	01	3000	67.78	49.36	49.35
Robbie Williams	01	3500	67.72	49.37	49.36
Robbie Williams	01	4000	67.68	49.46	49.46
Robbie Williams	01	6000	67.72	49.56	49.54

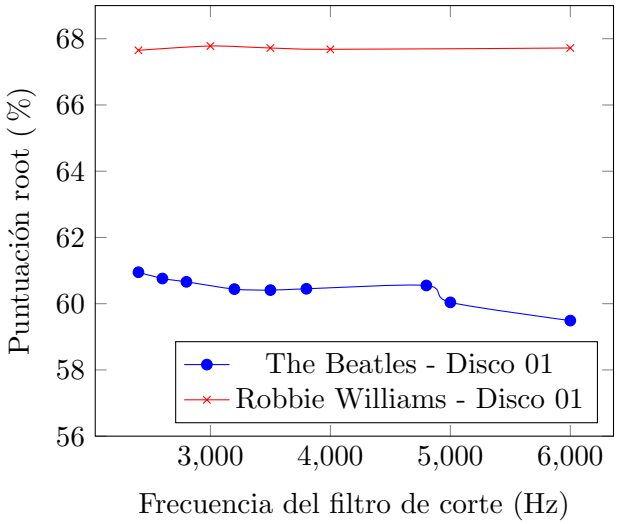


Figura 6.5: Puntuación *root* en función de la frecuencia de corte del filtrado de armónicos

Tabla 6.6: Puntuaciones en función del tipo de cromagrama

Artista	Disco	tipo	root	majmin	mirex
Beatles	01	stft	40.60	34.83	36.10
Beatles	01	cens	59.79	49.70	50.15
Beatles	01	cqt	59.31	47.86	48.33
Beatles	02	stft	43.83	36.74	35.96
Beatles	02	cens	60.45	46.12	45.13
Beatles	02	cqt	57.12	43.99	42.99

Tabla 6.7: Puntuaciones en función del hop_length utilizado para el cálculo del cromagrama

Artista	Disco	hop	root	majmin	mirex
Beatles	01	128	54.52	44.66	45.12
Beatles	01	256	56.55	46.45	46.94
Beatles	01	512	59.79	49.70	50.15
Beatles	01	1024	55.42	36.23	36.17

ahora a probar distintos **hop_length**, el número de muestras entre dos vectores de croma consecutivos. En la tabla 6.7 vemos el efecto de este parámetro. En ella podemos ver que **el hop_length ideal es 512.**

6.1.5. Prefiltering: Filtro de mediana

Nos interesa ahora conocer cómo afecta el tamaño de la máscara usada en el filtro de mediana en prefiltering, al resultado final de la estimación. Para ello, volvemos a utilizar la configuración básica, e iremos modificando el tamaño de la máscara progresivamente. En la figura 6.8 vemos los resultados obtenidos.

Además, en la tabla 6.9 vemos cómo afectaría al sistema eliminar este filtro. Tras

Tabla 6.8: Puntuaciones en función del tamaño del kernel usado en el filtro de mediana de prefiltering

Artista	Disco	mask_size	root	majmin	mirex
Beatles	01	3	59.79	49.70	50.15
Beatles	01	5	59.79	49.70	50.15
Beatles	01	7	59.79	49.68	50.14
Beatles	01	11	59.80	49.82	50.27
Beatles	01	15	59.83	49.82	50.28
Beatles	01	21	60.11	50.21	50.66
Beatles	01	29	60.31	50.21	50.65
Beatles	01	61	59.52	49.48	49.89

Tabla 6.9: Puntuaciones sin el filtro de mediana de prefiltering

Artista	Disco	root	majmin	mirex
Beatles	01	59.77	49.68	50.13
Beatles	02	60.47	46.12	45.13
Beatles	03	62.61	50.05	49.95

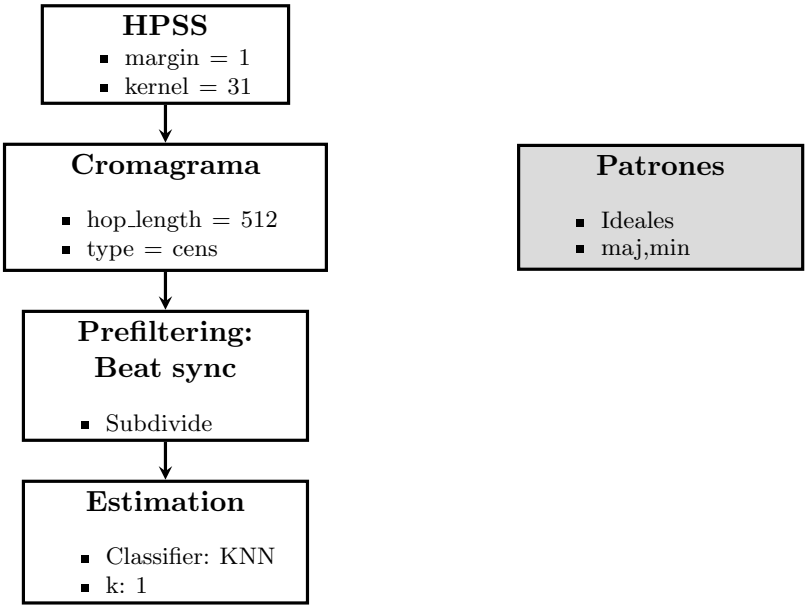


Figura 6.6: Configuración del sistema, usando sincronización con beat en prefiltering

ver las dos tablas anteriores podemos concluir que **el uso del filtro de mediana en prefiltering prácticamente no supone ninguna mejora en el resultado final de la estimación**. En el mejor caso, con un tamaño de máscara de 29, la estimación ha mejorado en un 0.5 % aproximadamente.

6.1.6. Prefiltering: Sincronización con beat

Una alternativa a usar la combinación de **Filtro de mediana en prefiltering + Sincronización con beat en postfiltering** es realizar la sincronización con el pulso en la fase de prefiltering, es decir, antes de la estimación. En la figura 6.6 vemos el esquema de la configuración usada para este apartado.

En la tabla 6.10 podemos ver los resultados obtenidos usando la sincronización con el pulso en prefiltering, tanto subdividiendo el beat, como sin subdividir.

Podemos ver que, al menos para los datasets analizados, **los resultados son prácticamente iguales que con la sincronización con el beat en postfiltering**. Además

Tabla 6.10: Puntuaciones obtenidas usando sincronización con el beat en prefiltering

Artista	Disco	subdivide	root	majmin	mirex
Beatles	01	NO	59.72	49.61	50.13
Beatles	01	SÍ	59.60	49.61	49.68
Beatles	02	NO	60.61	46.25	45.23
Beatles	02	SÍ	60.52	45.94	44.94
Beatles	03	NO	61.55	49.65	49.65
Beatles	03	SI	62.40	49.67	49.60

Tabla 6.11: Puntuaciones obtenidas usando sincronización con el beat en postfiltering sin subdividr el beat

Artista	Disco	root	majmin	mirex
Beatles	01	60.15	50.54	51.05
Beatles	02	60.94	46.73	45.73
Beatles	03	62.31	50.61	50.51

vemos, que no existe apenas diferencia entre subdividir el beat y no hacerlo.

6.1.7. Postfiltering: Sincronización con beat

Volvemos de nuevo a la configuración básica, pero ahora comprobaremos como afecta al sistema si, en la sincronización con el beat de la fase de postfiltering, no subdividiéramos los beats. Podemos ver los resultados en la tabla 6.11. Los resultados muestran que **no realizar la subdivisión con el beat proporciona en este caso un ligero aumento de alguna décima en la puntuación final**.

6.1.8. Patrones y clasificadores

Tenemos tres tipos de patrones de cromagramas en nuestro sistema que habrá que evaluar: ideales, ideales con armónicos y patrones obtenidos a partir de csound. Además dentro de los dos primeros grupos, podemos incluir únicamente acordes mayores y menores o incluir también los acordes de séptima. En la tabla 6.12 vemos el resultado del análisis, realizado a partir de la configuración básica. De aquí, podemos extraer conclusiones bastante importantes para el desarrollo del sistema:

- El uso de patrones con armónicos mejora notablemente la estimación
- Con respecto al uso de acordes de séptima en el conjunto de entrenamiento, no llegamos a una conclusión clara. Existen ocasiones en las que esto supone una mejora en la estimación, y otras ocasiones en las que supone un ligero descenso de la puntuación final.
- De los 3 datasets, el que mejores resultados proporciona es **fluid**

Tabla 6.12: Puntuaciones en función de los tipos de patrones elegidos

Artista	Disco	tipo patron	tipos de acordes	root	majmin	mirex
Beatles	01	ideal	maj,min	59.79	49.70	50.15
Beatles	01	ideal	maj,min,7,maj7,min7	60.93	53.88	56.74
Beatles	01	ideal + armónicos	maj,min	62.04	52.22	52.49
Beatles	01	ideal + armónicos	maj,min,7,maj7,min7	62.67	55.35	57.63
Beatles	01	csound (fluid)	maj,min	65.98	50.69	50.72
Beatles	01	csound (muse)	maj,min	65.26	48.65	48.68
Beatles	01	csound (tim)	maj,min	58.05	48.42	48.39
Beatles	02	ideal	maj,min	60.45	46.12	45.13
Beatles	02	ideal	maj,min,7,maj7,min7	57.39	47.02	50.95
Beatles	02	ideal + armónicos	maj,min	64.22	49.07	47.99
Beatles	02	ideal + armónicos	maj,min,7,maj7,min7	59.96	48.44	51.90
Beatles	02	csound (fluid)	maj,min	61.28	44.06	43.10
Beatles	02	csound (muse)	maj,min	60.38	41.94	41.01
Beatles	02	csound (tim)	maj,min	54.86	41.98	41.07

- El uso de los patrones basados en CSound no supone un incremento sustancial en la puntuación del sistema. Sin embargo, al ser un conjunto de entrenamiento mucho más extenso, este tipo de patrones tienen un margen de mejora mucho mayor, en función de los parámetros de los clasificadores. Por lo tanto, estos serán los patrones usados en nuestro sistema final.

6.1.9. Clasificadores

Para probar los clasificadores usaremos usaremos la **configuración básica**. Sin embargo, modificaremos el tipo de patrón, usando los obtenidos a partir de CSound. A partir de ahí, modificaremos los parámetros de los clasificadores y veremos cómo afectan a la estimación.

En la tabla **??** podemos ver el resultado de modificar el parámetro **k** para el clasificador knn. La conclusión que extraemos es que **el k ideal estaría entre 7 o 9**

Por otra parte, en la tabla 6.14 podemos ver el resultado de utilizar una **máquina de vector de soporte (SVM)**. Podemos ver, que la puntuación obtenida dependerá en gran medida del valor **C** de la SVM. **Esta puntuación llega a ser superior a la obtenida mediante el clasificador KNN**. Un valor de **C igual a 2** es el que proporciona mejores resultados.

Por último, el tercer clasificador que usaremos es una **red neuronal**. Podemos ver en la tabla 6.15 los resultados obtenidos. Como vemos, estos resultados son ligeramente inferiores a los conseguidos por los otros dos clasificadores.

Tabla 6.13: Puntuaciones en función de la k del clasificador KNN

Artista	Disco	k	root	majmin	mirex
Beatles	01	1	65.98	50.69	50.72
Beatles	01	3	67.19	55.62	55.59
Beatles	01	5	66.91	54.99	55.01
Beatles	01	7	66.88	54.76	54.73
Beatles	02	1	61.28	44.06	43.10
Beatles	02	3	63.42	47.19	46.14
Beatles	02	5	63.76	47.25	46.19
Beatles	02	7	63.85	47.42	46.35
Beatles	02	9	63.79	47.29	46.22
Robbie Williams	01	1	68.12	50.01	49.84
Robbie Williams	01	3	68.48	50.35	50.16
Robbie Williams	01	5	70.99	52.43	52.20
Robbie Williams	01	7	71.48	52.27	52.07
Robbie Williams	01	9	71.43	52.40	52.21

Tabla 6.14: Puntuaciones en función del parámetro C de la SVM

Artista	Disco	C	root	majmin	mirex
Beatles	01	0.001	65.93	51.59	51.62
Beatles	01	0.01	66.88	55.64	55.65
Beatles	01	0.05	63.80	50.13	50.15
Beatles	01	0.2	66.70	53.50	53.49
Beatles	01	0.5	67.79	55.34	55.32
Beatles	01	1	68.27	56.78	56.76
Beatles	01	2	68.91	58.32	58.30
Beatles	01	5	68.57	58.27	58.23
Beatles	01	10	67.74	57.54	57.72
Beatles	02	0.001	54.26	37.18	36.27
Beatles	02	0.01	55.36	37.49	36.56
Beatles	02	0.05	60.15	42.82	41.80
Beatles	02	0.1	62.26	49.50	44.41
Beatles	02	0.2	63.71	46.70	45.61
Beatles	02	0.3	64.74	47.72	46.61
Beatles	02	0.5	65.41	48.16	47.04
Beatles	02	0.7	65.63	48.74	47.61
Beatles	02	0.9	65.51	49.83	48.69
Beatles	02	5	64.81	49.79	48.70
Beatles	02	10	64.23	49.78	48.69
Beatles	02	20	64.35	49.81	48.75

Tabla 6.15: Puntuaciones en función del parámetro α y el número de capas ocultas de la red neuronal

Artista	Disco	α	hidden layers	root	majmin	mirex
Beatles	01	0.1	(30,30,30)	61.69	51.63	51.61
Beatles	01	1	(30,30,30)	64.45	54.01	54.01
Beatles	01	10	(30,30,30)	63.48	49.37	49.56
Beatles	01	0.1	(30,30,30,30,30)	61.80	49.77	49.82
Beatles	01	1	(30,30,30,30,30)	61.56	51.58	51.57
Beatles	01	10	(30,30,30,30,30)	16.45	11.82	12.24

Tabla 6.16: Puntuaciones del sistema final

Artista	root	majmin	mirex
Beatles	65.3	54.98	53.75
Queen	54.77	43.28	42.73
Robbie Williams	66.70	54.90	53.33

6.2. Sistema final

Analizados los demás subsistemas, ya estamos en condiciones de elegir todos los parámetros del sistema final. Podemos ver los subsistemas y parámetros utilizados en la figura 6.7. En la tabla 6.16 se muestran los resultados obtenidos por este sistema para los 3 datasets.

6.2.1. Velocidad de ejecución

Analizando los documentos generados a partir de los análisis a los 3 posibles datasets podemos observar un hecho muy interesante. El tiempo empleado por el programa para analizar todas estas canciones es mucho menor que la duración de todas ellas. El ordenador en el que se ha ejecutado tiene un procesador Intel Core i5-3230M con 4 núcleos a una frecuencia de 2.60GHz. Esto tiene una implicación muy importante, nuestro sistema podría trabajar **a tiempo real**, en cualquier sistema con prestaciones que no tienen por qué ser demasiado elevadas.

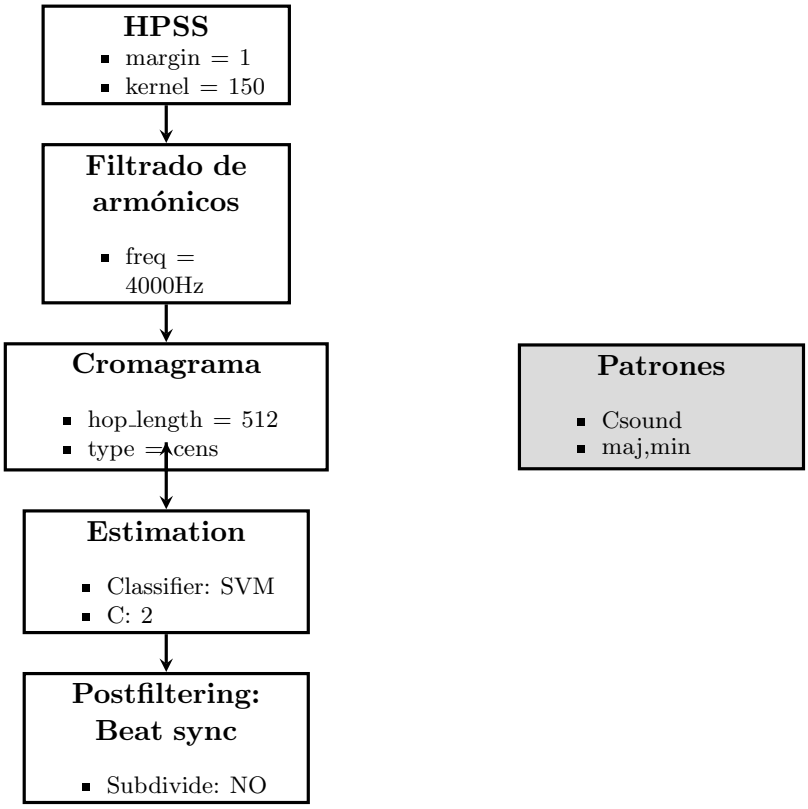


Figura 6.7: Configuración del sistema final

7 Conclusiones

Como el lector habrá podido comprobar, en este trabajo se ha puesto gran esfuerzo en comprobar, de forma exhaustiva, absolutamente todos los parámetros y algoritmos del proceso de estimación de acordes. Gracias a esto, hemos comprobado que el peso real del sistema de estimación se encuentra en los **patrones** y en el **clasificador** usados. Estos son los elementos verdaderamente diferenciales gracias a los que se producen los grandes avances. Hemos visto que el clasificador que más útil nos será es la máquina de vector de soporte. Además, los patrones obtenidos a partir de muestras de instrumentos reales son los que nos han dado mejores resultados. Todos los demás elementos del sistema, de adaptación de la señal, prefiltering o postfiltering, suponen mejoras muy pequeñas (aunque no podemos obviarlos, pues la suma de muchas pequeñas mejoras, puede llegar a ser muy importante). Además, para que estas mejoras tengan efecto hay que realizar un ajuste muy fino de los parámetros de dichos elementos.

Los algoritmos usados en el sistema son los utilizados en este campo desde hace muchos años. No son especialmente novedosos, sin embargo, hasta ahora no se había realizado ningún otro trabajo que detallara en qué medida beneficiarían al sistema cada uno de ellos. Como era de esperar, obtener puntuaciones cercanas a las obtenidas en las últimas ediciones de **MIREX** no ha sido posible. En unos meses de trabajo es muy difícil llegar a asemejarse a sistemas que tienen detrás a varios investigadores trabajando durante años. Sin embargo, hemos descubierto que ajustando bien los parámetros de cada parte del sistema, con los componentes básicos de un sistema ACE, podemos llegar a obtener puntuaciones sin nada que envidiar a las de otros trabajos que usan componentes mucho más sofisticados. El problema del deep learning, que está detrás de muchos de los sistemas ACE actuales, es que el sistema puede funcionar o no, pero un conjunto de valores de pesos de conexiones neuronas no darán al investigador la información necesaria para saber qué está pasando. Son cajas negras a las que simplemente les pasamos una serie de inputs, y recibimos unos output determinados. Puesto que es inevitable que la estimación de acordes avance, como muchos otros campos, hacia el deep learning, trabajos como el presente permitirán arrojar un poco de claridad al proceso. Ya que no podemos evitar que el clasificador sea una caja negra, al menos será de utilidad que conozcamos cómo afectan el resto de elementos, y por qué el uso de ciertos parámetros beneficia al sistema.

Por desgracia, el tiempo es limitado, y hay que parar el desarrollo en un punto. Sin embargo, son muchas las dudas que aún nos quedan en el aire. El lector deberá comprender que en el presente trabajo, los diferentes subsistemas han sido evaluados sobre una configuración básica. Sin embargo, aunque son sistemas parcialmente aislados y no debería suponer un gran cambio, no podemos asegurar que el uso de estos subsistemas sobre otra configuración distinta tenga exactamente el mismo efecto. Esta es una de las lineas de trabajo futuro, habría que *rizar el rizo* un poco más, y evaluar también los

subsistemas de forma combinada, y sobre otras configuraciones base. Sólo así seríamos capaces de evaluar si esta suposición de partida era realmente correcta.

Como hemos visto, el elemento que ha supuesto una gran mejora en nuestro sistema, ha sido el uso de patrones de cromagramas obtenidos a partir de instrumentos sintetizados a partir de soundfonts de instrumentos reales. Sin embargo, a raíz de los resultados obtenidos en los patrones ideales, en los que el uso de patrones de acordes de séptima no suponía prácticamente ninguna mejora en el sistema, se decidió no generar también patrones de acordes de séptima a partir de soundfonts. Sin embargo, teniendo más tiempo, el uso de estos patrones, junto con patrones de inversiones de los acordes, podría resultar beneficioso al sistema.

Otro apartado clave que hubiera podido suponer una importante mejora, es el análisis de los posibles patrones de los clasificadores usados. Aunque hayamos usado uno o dos parámetros para controlar cada clasificador, en realidad estos algoritmos dependen de muchos más parámetros que siendo ajustados correctamente, podrían mejorar en gran medida el resultado de la estimación. De hecho, podría usarse la función **GridSearchCV** de **sklearn** que además va muy en consonancia con la filosofía de este proyecto. Esta función permite definir una lista de posibles valores para los distintos parámetros del clasificador, que **sklearn** analizará con el fin de proporcionar los que maximicen la puntuación final.

Por último, es necesario añadir al sistema información del **contexto**. Como sabemos, hay acordes que tienen una probabilidad mucho mayor de aparecer en un determinado momento de la obra que otros. Incorporar esta información al sistema puede ser de gran utilidad. Durante este trabajo, hemos visto varias formas de realizar esta tarea. Posiblemente, la más adecuada para este proyecto sería el uso de un clasificador basado en redes neuronales recurrentes.

Bibliografía

[Bello and Pickens, 2005] Bello, J. P. and Pickens, J. (2005). A robust mid-level representation for harmonic content in music signals. In *ISMIR*, volume 5, pages 304–311.

[Benetos et al., 2012] Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., and Klapuri, A. (2012). Automatic music transcription: Breaking the glass ceiling.

[Boulanger-Lewandowski et al., 2013] Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2013). Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340.

[Brown, 1991] Brown, J. C. (1991). Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434.

[Burgoyne et al., 2011] Burgoyne, J. A., Wild, J., and Fujinaga, I. (2011). An expert ground truth set for audio chord recognition and music analysis. In *ISMIR*, volume 11, pages 633–638.

[Cazden, 1945] Cazden, N. (1945). Musical consonance and dissonance: A cultural criterion. *The Journal of Aesthetics and Art Criticism*, 4(1):3–11.

[Chafe and Jaffe, 1986] Chafe, C. and Jaffe, D. (1986). Source separation and note identification in polyphonic music. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 1289–1292. IEEE.

[Chafe et al., 1985] Chafe, C., Jaffe, D. A., Kashima, K., Mont-Reynaud, B., and Smith, J. O. (1985). *Techniques for note identification in polyphonic music.* CCRMA, Department of Music, Stanford University.

[Chen et al., 2012] Chen, R., Shen, W., Srinivasamurthy, A., and Chordia, P. (2012). Chord recognition using duration-explicit hidden markov models. In *ISMIR*, pages 445–450.

[Deutsch, 2013] Deutsch, D. (2013). *Psychology of music.* Elsevier.

[Di Giorgi et al., 2013] Di Giorgi, B., Zanoni, M., Sarti, A., and Tubaro, S. (2013). Automatic chord recognition based on the probabilistic modeling of diatonic modal harmony. In *Multidimensional Systems (nDS), 2013. Proceedings of the 8th International Workshop on*, pages 1–6.

[Downie, 2003] Downie, J. S. (2003). Music information retrieval. *Annual review of information science and technology*, 37(1):295–340.

[Driedger et al., 2014] Driedger, J., Müller, M., and Disch, S. (2014). Extending harmonic-percussive separation of audio signals. In *ISMIR*, pages 611–616.

[Durrieu et al., 2009] Durrieu, J.-L., Richard, G., and David, B. (2009). An iterative approach to monaural musical mixture de-soloing. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 105–108. IEEE.

[Ellis, 2007a] Ellis, D. (2007a). Chroma feature analysis and synthesis. *Resources of Laboratory for the Recognition and Organization of Speech and Audio-LabROSA*.

[Ellis, 2007b] Ellis, D. P. (2007b). Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60.

[Fitzgerald, 2010] Fitzgerald, D. (2010). Harmonic/percussive separation using median filtering.

[Fujishima, 1999] Fujishima, T. (1999). Realtime chord recognition of musical sound: a system using common lisp music. In *ICMC*, pages 464–467.

[Gil Ortiz, 2016] Gil Ortiz, D. (2016). Estimación automática de acordes en audio mediante clasificación instrumental. http://grfia.dlsi.ua.es/repositori/grfia/degreeProjects/23/Memoria.pdf.

[Harte, 2010] Harte, C. (2010). *Towards automatic extraction of harmony information from music signals*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London.

[Harte and Sandler, 2005] Harte, C. and Sandler, M. (2005). Automatic chord identifcation using a quantised chromagram. In *Audio Engineering Society Convention 118*. Audio Engineering Society.

[Harte et al., 2006] Harte, C., Sandler, M., and Gasser, M. (2006). Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26. ACM.

[Harte et al., 2005] Harte, C., Sandler, M. B., Abdallah, S. A., and Gómez, E. (2005). Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR*, volume 5, pages 66–71.

[Heittola et al., 2009] Heittola, T., Klapuri, A., and Virtanen, T. (2009). Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *ISMIR*, pages 327–332.

[Helmholtz and Ellis, 1886] Helmholtz, H. and Ellis, A. J. (1886). On the sensations of tone, as a physiological basis for the theory of music.

[Humphrey and Bello, 2012] Humphrey, E. J. and Bello, J. P. (2012). Rethinking automatic chord recognition with convolutional neural networks. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 357–362. IEEE.

[Kashino and Hagita, 1996] Kashino, K. and Hagita, N. (1996). A music scene analysis system with the mrf-based information integration scheme. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 2, pages 725–729. IEEE.

[Korzeniowski and Widmer, 2016] Korzeniowski, F. and Widmer, G. (2016). A fully convolutional deep auditory model for musical chord recognition. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*, pages 1–6. IEEE.

[Kuncheva and Jain, 1999] Kuncheva, L. I. and Jain, L. C. (1999). Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern recognition letters*, 20(11):1149–1156.

[Lee and Downie, 2004] Lee, J. H. and Downie, J. S. (2004). Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *ISMIR*, volume 2004, page 5th.

[Lee and Slaney, 2006] Lee, K. and Slaney, M. (2006). Automatic chord recognition from audio using a hmm with supervised learning. In *ISMIR*, pages 133–137.

[Magalhaes and de Haas, 2011] Magalhaes, J. P. and de Haas, W. B. (2011). Functional modelling of musical harmony: an experience report. In *ACM SIGPLAN Notices*, volume 46, pages 156–162. ACM.

[Martin, 1996] Martin, K. D. (1996). A blackboard system for automatic transcription of simple polyphonic music. *Massachusetts Institute of Technology Media Laboratory Perceptual Computing Section Technical Report*, 385.

[Mauch and Dixon, 2010a] Mauch, M. and Dixon, S. (2010a). Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140.

[Mauch and Dixon, 2010b] Mauch, M. and Dixon, S. (2010b). Mirex 2010: Chord detection using a dynamic bayesian network. *Music Information Retrieval Evaluation Exchange (MIREX)*.

[Mauch and Dixon, 2010c] Mauch, M. and Dixon, S. (2010c). Simultaneous estimation of chords and musical context from audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1280–1289.

[Mauch et al., 2009] Mauch, M., Noland, K., and Dixon, S. (2009). Using musical structure to enhance automatic chord transcription. In *ISMIR*, pages 231–236.

[McFee et al., 2015] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*.

[Müller and Ewert, 2011] Müller, M. and Ewert, S. (2011). Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR), 2011. hal-00727791, version 2-22 Oct 2012*. Citeseer.

[Murphy, 2002] Murphy, K. P. (2002). *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley.

[Ni et al., 2012] Ni, Y., McVicar, M., Santos-Rodriguez, R., and De Bie, T. (2012). An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783.

[Ono et al., 2008] Ono, N., Miyamoto, K., Le Roux, J., Kameoka, H., and Sagayama, S. (2008). Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram. In *Signal Processing Conference, 2008 16th European*, pages 1–4. IEEE.

[Papadopoulos and Peeters, 2007] Papadopoulos, H. and Peeters, G. (2007). Large-scale study of chord estimation algorithms based on chroma representation and hmm. In *Content-Based Multimedia Indexing, 2007. CBMI'07. International Workshop on*, pages 53–60. IEEE.

[Pauws, 2004] Pauws, S. (2004). Musical key extraction from audio. In *ISMIR*.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Raffel et al., 2014] Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O., Liang, D., Ellis, D. P., and Raffel, C. C. (2014). mir_eval: A transparent implementation of common mir metrics. In *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer.

[Rasch and Plomp, 1999] Rasch, R. and Plomp, R. (1999). The perception of musical tones. *The psychology of music*, 2:89–112.

[Reed et al., 2009] Reed, J., Ueda, Y., Siniscalchi, S. M., Uchiyama, Y., Sagayama, S., and Lee, C.-H. (2009). Minimum classification error training to improve isolated chord recognition. In *ISMIR*, pages 609–614.

[Segura Sogorb, 2015] Segura Sogorb, M. (2015). Estimación automática de acordes para uso en transcripción musical a partir de audio. http://grfia.dlsi.ua.es/repositori/grfia/degreeProjects/5/memoria.pdf.

[Shepard, 1964] Shepard, R. N. (1964). Circularity in judgments of relative pitch. *The Journal of the Acoustical Society of America*, 36(12):2346–2353.

[Sigtia et al., 2015] Sigtia, S., Boulanger-Lewandowski, N., and Dixon, S. (2015). Audio chord recognition with a hybrid recurrent neural network. In *ISMIR*, pages 127–133.

[Stark and Plumbley, 2009] Stark, A. M. and Plumbley, M. D. (2009). Real-time chord recognition for live performance. In *Proceedings of the 2009 International Computer Music Conference, ICMC 2009*, pages 85–88.

[Sumi et al., 2008] Sumi, K., Itoyama, K., Yoshii, K., Komatani, K., Ogata, T., and Okuno, H. G. (2008). Automatic chord recognition based on probabilistic integration of chord transition and bass pitch estimation. In *ISMIR*, pages 39–44.

[Terhardt, 1974] Terhardt, E. (1974). Pitch, consonance, and harmony. *The Journal of the Acoustical Society of America*, 55(5):1061–1069.

[Ueda et al., 2010] Ueda, Y., Uchiyama, Y., Nishimoto, T., Ono, N., and Sagayama, S. (2010). Hmm-based approach for automatic chord detection using refined acoustic features. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5518–5521. IEEE.

[Varewyck et al., 2008] Varewyck, M., Pauwels, J., and Martens, J.-P. (2008). A novel chroma representation of polyphonic music based on multiple pitch tracking techniques. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 667–670. ACM.

[Weller et al., 2009] Weller, A., Ellis, D., and Jebara, T. (2009). Structured prediction models for chord transcription of music audio. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 590–595. IEEE.

[Yoshioka et al., 2004] Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T., and Okuno, H. G. (2004). Automatic chord transcription with concurrent recognition of chord symbols and boundaries. In *ISMIR*, pages 100–105.