Escuela Politécnica Superior

Estimación automática de acordes en audio mediante clasificación instrumental

Grado en Ingeniería en Sonido e Imagen en Telecomunicación

Trabajo Fin de Grado

Autor:
Daniel Gil Ortiz

Tutor/es:
José Manuel Iñesta Quereda

José Javier Valero Más

Septiembre 2016

Universitat d'Alacant
Universidad de Alicante

Estimación automática de acordes en audio mediante clasificación instrumental

Daniel Gil Ortiz

Septiembre 2016

Resumen

La extracción de información musical, más concretamente la detección automática de acordes, es un área de investigación abierta desde hace muchos años debido a la dificultad que supone realizar dicha tarea de forma manual. De este modo, se propone el desarrollo e implementación de un sistema automático de estimación de acordes de señales de audio por medio de comparación de patrones armónicos (cromagrama), siendo el motivo principal de este trabajo los patrones obtenidos mediante clasificación instrumental.

El trabajo se realizará principalmente en entorno MATLAB, donde se implementarán todos los algoritmos de automatización de los procesos vinculados con el trabajo, diferentes técnicas destinadas al análisis armónico y espectral de señales de audio, así como técnicas de aprendizaje automático como pueda ser el algoritmo k-means para la clasificación instrumental. Al mismo tiempo, y con intención de disponer de una amplia biblioteca de sonidos con los que desarrollar las fases de entrenamiento y de test, será necesaria la implementación de scripts en Csound que faciliten la síntesis automática de audio musical. Para disponer de información específica sobre el análisis sonoro, también se hará uso de sistemas como Sonic Annotator y Sonic Visualizer.

Agradecimientos

Tantas son las personas que se cruzan por nuestra vida, simplemente caminando por la calle, tomando un café, compartiendo unas risas, o convirtiéndose en el mejor de los compañeros habidos o por haber, que estoy seguro de que, desmerecidamente, dejaré a alguien fuera de estos párrafos. Sea como sea, trataré al menos de nombrar a todos aquellos que hayan compartido conmigo (ya sea de un modo u otro) esta maravillosa etapa de mi vida que siempre recordaré, y que aún no puedo imaginar acabando.

Como cabe esperar (o al menos creo que siempre debería ser así), agradecer a mis padres, mi familia, su apoyo incondicional para que siguiese estudiando. Siempre habéis guardado una frase de ánimo, un empujón para que siguiese trabajando en lo que creo. Nunca sabré cómo agradeceros toda la confianza que depositáis en mí.

A Marta, mi compañera de aventuras y aprendizaje vital. Gracias por darme en cada momento las palabras que necesito, y no solo las que quiero. Has tenido que aguantar muchos de mis momentos de achaque, pero también los buenos, y este es ahora uno muy grande que quiero compartir contigo. Sin olvidar a Domingo, mi nuevo amigo peludo, que sentándose encima del teclado del ordenador me recuerda que además de trabajar también hay que saber divertirse, ya sea cazando moscas o jugando con una bola de papel.

A José Manuel Iñesta por avivar mi interés por la informática musical. Las clases de Síntesis Digital del Sonido me mostraron la vía en la que poner en común ingeniería con mi gran pasión por la música es posible, y fruto de ellas ha sido la elaboración de este trabajo, del cual no puedo sentirme más agradecido que hayas aceptado. También quiero dar las gracias a todos aquellos profesores de la Universidad de Alicante que me han influenciado, y me han enseñado lo maravilloso que puede ser estudiar una ingeniería. Gracias a Miguel Romá, Josep David Ballester, Tomás Martínez, Juan Manuel López, Sergio Bleda, Carolina Pascual, Eva María Calzado y Jenaro Vera solo por decir algunos nombres.

A José Javier por aguantarme prácticamente todos los días con millones de dudas. Ya fuesen en mis frecuentes visitas o con mis interminables correos electrónicos, siempre has tenido tiempo para darme un millón de valiosas respuestas sin las que no podría haber realizado este trabajo.

A todos mis amigos, especialmente a Antonio y David, juntos hemos compartido gran parte de nuestra vida, y me habéis enseñado que los amigos de verdad resisten el tiempo y la distancia. A mis fabulosos compañeros de clase: a Julián, Iñigo, Alberto, Samu, Lidia, Martín

y Juanpe por los maravillosos años juntos; a Miguel e Hirahí por nuestras interminables charlas de audio; y por supuesto también a Jaime (echo de menos tus chistes), Ferris, Patrick y Leti.

Para terminar, y no por ello menos importante, a todos los músicos con los que he tenido el placer de poder compartir unos acordes en algún momento de mi vida (al final la cosa ha ido de esto). Tanto las personas con las que he ensayado esporádicamente, como con las que llevo tocando desde que me compré mi primer bajo me han influido, y me han ayudado a convertirme en el músico que ahora soy. Gracias a todos ellos, especialmente a Lucas, Juanma y Manu. Con vosotros he vivido una década de viajes, furgoneta, risas, sesiones interminables de grabación, lecciones, y experiencias inolvidables que espero que sigan sucediendo.

Índice

Índice **7**

Índice de tablas **9**

1. Introducción **11**

1.1. Motivación 11

1.2. Objetivos 12

1.3. Campo de aplicación 12

2. Marco teórico **15**

2.1. Características espectrales de una señal de audio 15

2.2. Altura y duración de notas musicales 17

2.3. Escalas 19

2.4. Acordes 21

2.5. Teorema y transformada de Fourier 21

2.6. Cromagrama 23

2.7. Aprendizaje automático no supervisado 25

3. Descripción del sistema **29**

3.1. Corpus empleado 29

3.1.1. Vocabulario armónico 29

3.1.2. Ficheros de instrumentos sintetizados 30

3.1.3. Ficheros de instrumentos reales grabados 31

3.2. Necesidad de automatización 31

3.3. Sistema inicial 32

3.3.1. Algoritmo de separación armónico-percusivo 32

3.3.2. Sonic Annotator y extracción de cromagrama 34

3.3.3. Generación de plantillas 36

3.3.4. Comparación de patrones 38

3.3.5. Estimación de acordes 39

3.4. Sistema basado en clasificación instrumental 39

3.4.1. Plantillas de familias de instrumentos GM 40

3.4.2. Estudio de las agrupaciones 42
3.4.3. Plantillas obtenidas del clustering 45

4. Evaluación **47**
4.1. Métricas empleadas 47
4.2. Resultados 49

5. Conclusiones **53**
5.1. Discusión de resultados 53
5.2. Líneas futuras de trabajo 55

6. Bibliografía **57**

A. Anexo: Scripts desarrollados **59**

Índice de tablas

1. Etiquetas de acorde empleadas 30
2. Resultados para ficheros sintetizados con el motor TimGM6mb. 50
3. Resultados para ficheros sintetizados con el motor FluidR3GM2. 51
4. Resultados para ficheros de instrumentos reales grabados. 52
5. Comparativa de tasas de acierto general. 53

1. Introducción

El principal pilar sobre el que se sustenta una pieza musical occidental es su estructura armónica, siendo ésta el cifrado armónico que refleja la sucesión progresiva de los acordes que la componen formando así el esqueleto armónico de la misma. La posibilidad de disponer de dicha información no solo resulta de gran importancia para un grupo de personas que pretende entenderse en un entorno musical, ya que será necesaria para la ejecución contextualizada de cada instrumentista, del mismo modo resultará clave a la hora de realizar otras tareas complejas como pueda ser la identificación del estilo musical de la obra, o la detección de su tonalidad por poner algunos ejemplos.

Motivado por el interés que supondría disponer de dicha información, así como cualquier otro tipo de datos musicales relevantes, nace un campo de investigación llamado MIR (Music Information Retrieval), el cual engloba (entre otras cosas) diferentes tareas de extracción automática de información musical. Con un enfoque multi-disciplinar que incluye el tratamiento digital de la señal, el aprendizaje automático, o la psico-acústica, los desarrollos e investigaciones relacionadas con la extracción de información musical vienen realizándose desde hace unos 15 años, creciendo de forma progresiva, y produciéndose constantes avances que, debido a sus múltiples campos de aplicación, siguen causando gran interés en el ámbito científico-técnico.

1.1. Motivación

Movido por la pasión del autor del presente proyecto por la música, nace la idea de elaborar como Trabajo Final de Grado (TFG) un sistema relacionado con la extracción de información musical. Es así como surge la propuesta por parte conjunta con José Manuel Iñesta (profesor en la Universidad de Alicante, así como tutor del TFG) de realizar un sistema de estimación automática de acordes.

De este modo, y con la colaboración de José Javier Valero como co-tutor, se propone el desarrollo de un sistema modular que, partiendo de algunos de los fundamentos en los que se desarrolla el Trabajo Final de Grado 'Estimación automática de acordes para uso en transcripción musical a partir de audio' realizado por Miguel Segura (Septiembre de 2015), sea capaz de contextualizarse dentro del marco de investigación del Departamento de Lenguajes y Sistemas Informáticos (DLSI) de la Universidad de Alicante.

Partiendo del enfoque general respaldado por la comparación de patrones de cromagrama, se propone la implementación y desarrollo de un sistema que, mediante la utilización de plantillas generalizadas a familias de instrumentos musicales, obtenga información armónica precisa.

1.2. Objetivos

Desde un punto de vista general, los objetivos del trabajo descrito son los de comprender e implementar distintas técnicas relacionadas con la estimación automática de acordes, emplear técnicas de procesado de la señal, y la categorización de diferentes plantillas de cromagrama por familias de instrumentos (basadas en la lista de instrumentos General Midi) con el fin de particularizar el problema, y de este modo conseguir mejores resultados.

Al mismo tiempo, y como consecuencia de los objetivos antes mencionados, se plantean transversalmente objetivos como la automatización del sistema, la contextualización del mismo bajo el marco propuesto por MIREX (Music Information Retrieval Evaluation eXchange), y la comprensión e implementación de algoritmos de aprendizaje automático no supervisado como es el caso concreto de k-means para proponer clasificaciones instrumentales basadas en caracterización armónica.

1.3. Campo de aplicación

A menudo que las investigaciones relativas a MIR avanzan, son más visibles las aplicaciones directas de éstas. Es así como, concretamente en el campo de la estimación de acordes, se desarrollaron diferentes sistemas como el popular Chordino (2010), el cual fue desarrollado por la Universidad Queen Mary en Londres como un plugin V-Amp capaz de ser lanzado a través de Sonic Visualizer o Audacity.

Ya en 2013 surgen otros sistemas más enfocados al público que no dispone necesariamente de conocimientos técnicos, facilitando de este modo su ejecución y uso. Es así cómo, entre otras, surge la herramienta web Chordify, la cual obtiene los acordes de un fichero en MP3, incluso de una URL de YouTube facilitada por el usuario.

Más actualmente, y estando ya el campo de investigación bastante maduro, no resulta complicado encontrar aplicaciones móviles (iOS y Android) que utilicen algoritmos de estimación de acordes. Es así como aplicaciones modernas como Smule o Yousician permiten, a través de la detección de acordes, crear una vía de comunicación en tiempo real con el

intérprete, y así poder preparar ejercicios personalizados orientados principalmente al ensayo y al aprendizaje.

A su vez, también se pueden encontrar herramientas de gran utilidad para músicos y compositores, como es el caso de Bloc Musical (desarrollado por Apple). Una aplicación móvil que permite registrar bocetos musicales, analizando al mismo tiempo la estructura armónica de los mismos para hacer propuestas de patrones rítmicos y arreglos, y facilitar el proceso de composición.

De este modo, y teniendo en cuenta que actualmente ya existen diferentes herramientas que explotan las posibilidades que ofrece la detección automática de acordes, se espera del presente trabajo el incremento en la precisión de este tipo de sistemas, siendo una posible aplicación directa del mismo la propuesta de una aplicación que ofrezca una mejor funcionalidad.

2. Marco teórico

En la siguiente sección se procederá a dar el soporte teórico necesario para comprender el trabajo realizado, así como su contextualización dentro de las diferentes disciplinas involucradas. De este modo se tratará, dentro de lo posible, dar un enfoque secuencial en el que aquellas explicaciones teóricas más complejas se apoyen en las descritas previamente.

2.1. Características espectrales de una señal de audio

Partiendo de la base de la que una onda sonora es la propagación de una perturbación de la presión en el aire, y que de este modo una señal de audio se definirá como una onda sonora que transporta información de audio, los parámetros que definirán la periodicidad de dicha señal son el periodo (T) y la frecuencia (f), los cuales son la inversa uno del otro.

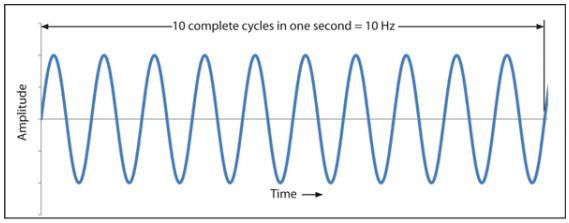

Figura 1: Frecuencia y periodo de una onda sinusoidal. Fuente: http://www.minelab.com

Cabe decir que las ondas sinusoidales como la de la Figura 1 (tono puro) no están presentes en la naturaleza, teniendo de este modo la mayoría de los sonidos reales una composición de diferentes frecuencias que conforman su espectro característico.

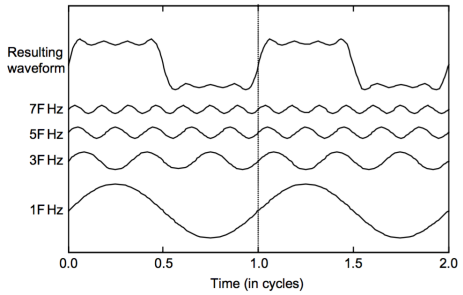

Figura 2: Resultado de la composición de diferentes sinusoides. Fuente: HOWARD & ANGUS, 1996

Teniendo en cuenta lo anteriormente explicado, cada una de las frecuencias que conforman un sonido se denominan frecuencias parciales, o sencillamente parciales. La cantidad de energía que tenga cada uno de los parciales del sonido representará la identidad frecuencial del mismo, o lo que es lo mismo, su espectro.

A su vez, un espectro se puede categorizar como armónico o inarmónico. La diferenciación resulta bastante sencilla, un espectro armónico será aquél en el que todos sus parciales sean múltiplos enteros de la frecuencia fundamental, siendo ésta la inversa del periodo más largo, aunque también se puede definir como el máximo común divisor de todos los parciales del espectro siempre que esté contenido dentro del rango audible de 20 Hz a 20 kHz. Así, aquél espectro que contenga armónicos que no sean múltiplos enteros de la frecuencia fundamental, será un espectro inarmónico.

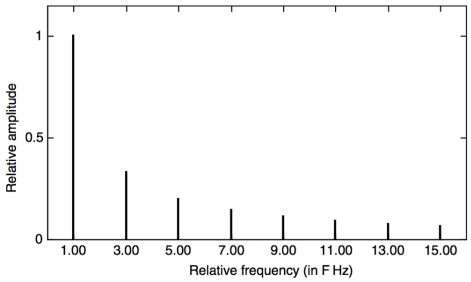

Figura 3: Espectro de una señal armónica. Fuente: HOWARD & ANGUS, 1996

Desde un enfoque mucho más realista, los sonidos no son necesariamente armónicos o inarmónicos. No se trata de una cuestión polarizada, si no que guardarán cierto grado de armonicidad en función de su espectro. Si analizamos el sonido producido por una nota de piano, podremos ver cómo, además de la frecuencia fundamental y los correspondientes armónicos característicos de una cuerda vibrante, se encuentra energía en otras frecuencias. Esto es debido, entre otros factores, al ruido generado por piezas vibrantes, o la misma resonancia de la caja del instrumento. De este modo, y teniendo en cuenta que un instrumento real no será completamente armónico, se define la armonicidad como la proporción unitaria de parciales armónicos entre sí (teniendo en cuenta la frecuencia fundamental).

Como norma general, los instrumentos determinados (que producen sonidos con una altura perceptible), como puedan ser los instrumentos de cuerda y vientos, tendrán espectros mucho más armónicos que los de instrumentos indeterminados (que producen sonidos sin una altura perceptible), como la mayoría de los instrumentos de percusión.

Además, el hecho de que el sonido que pueda producir un instrumento no es infinito en el tiempo, tiene como consecuencia directa que los armónicos producidos en el dominio frecuencial no se correspondan únicamente con una sola frecuencia localizada, si no con una banda de ellas generalmente con un máximo que representaría al parcial. A estas bandas de frecuencias que representan el espectro real se le denominan formantes, y determinan cuáles son las frecuencias predominantes de un sonido, definiendo así el timbre de un instrumento musical. Dicho de algún modo, el timbre sería el elemento que nos permitiría distinguir una misma nota producida por dos instrumentos distintos.

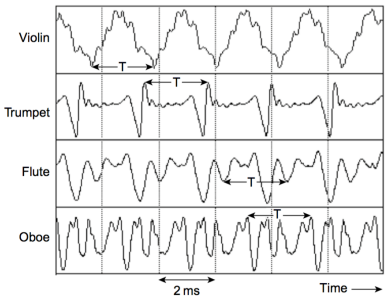

Figura 4: Timbre de diferentes instrumentos musicales. Fuente: HOWARD & ANGUS, 1996

2.2. Altura y duración de notas musicales

Con el fin de ofrecer una breve contextualización musical al trabajo realizado, se definirán algunos de los parámetros necesarios para la comprensión básica de la armonía implicada.

Por un lado está la altura, la cual se puede definir como el atributo de la sensación auditiva que hace que un sonido se ordene en una escala de bajo a alto (American National Standards Institute, 1960). En un pentagrama quedaría descrita por su posición en el eje vertical, siendo, al menos en lo referente al presente trabajo, el intervalo mínimo entre dos notas un semitono (1/2 tono).

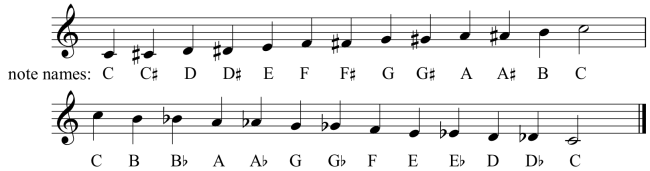

Figura 5: Escala cromática. Fuente: `http://www.musiccrashcourses.com`

Como se puede observar, existen 12 tipos de altura distanciadas cada una de ellas un semitono de la siguiente, creando así un vocabulario de notas en el que tenemos de C (Do) a B (Si), sucesión de notas que se repite de forma cíclica.

Empleando un enfoque mucho más físico, la altura expresa la frecuencia fundamental de la nota. Perceptualmente sentiremos cómo una nota con una frecuencia fundamental superior a otra tiene una altura mayor. Así por ejemplo, una nota con f_0=440 Hz (A4) será más alta que una nota con f_0=392 Hz (G4).

Sin entrar en detalles acerca de la afinación, ni la evolución de ésta a lo largo de la historia de la música occidental, en la actualidad se establecen las relaciones entre todas las notas musicales de forma discreta según la razón $2^{\frac{n}{12}}$, siendo n el número de semitonos. Por ejemplo, la frecuencia fundamental de un B4 se calcula como la frecuencia fundamental de un A4 por la relación descrita de un tono (dos semitonos) ($n = 2$):

$$440Hz \cdot 2^{\frac{2}{12}} = 493{,}9\ Hz \tag{1}$$

De este modo se definen cada una de las notas de la escala cromática tal y como se muestra en la siguiente figura:

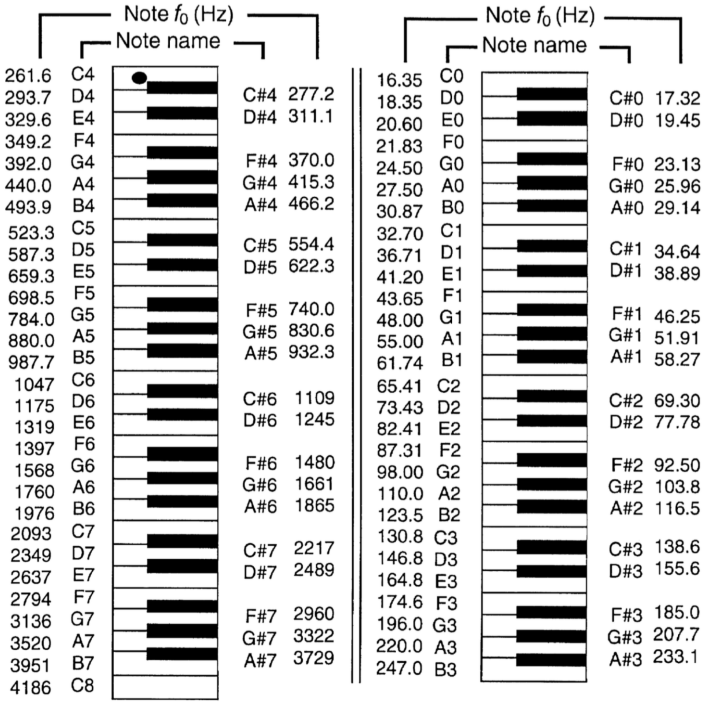

Figura 6: Frecuencia fundamental para las primeras 8 octavas. HOWARD & ANGUS, 1996

La música se define en muchas ocasiones como la habilidad de combinar diferentes sonidos en el tiempo, es por esto que cabe imaginar que la dimensión temporal debe quedar descrita en el lenguaje musical. Así, se define la simbología referente a la duración de las notas musicales en torno a una unidad denominada tiempo. El tiempo o pulso musical será por tanto los instantes de tiempo en los que se fragmentará una obra, siendo el tempo la velocidad del pulso medido en BPM (Pulsos por minuto).

Note	Name	Beats
	Whole note	4 beats
	Half note	2 beats
	Quarter note	1 beat
	Eighth note	½ beat
	Sixteenth note	¼ beat

Figura 7: Simbología referente a la duración de las notas musicales.

http://www.austincelloschool.com

2.3. Escalas

De forma general se emplea un conjunto ordenado de las 12 notas de la escala cromática para crear la armonía y melodía de una pieza musical al cual denominaremos escala. De este modo, y más concretamente en el campo de la música occidental, existen dos tipos de escala ampliamente utilizados, y que resultan perceptivamente más naturales a nuestro oído: las escalas diatónica mayor y menor.

Cualquier escala se verá definida por la sucesión de saltos de intervalo, formando así una secuencia caracterísitica. En función del patrón de la secuencia, diremos cómo es la escala. En lo que concierne al trabajo propuesto, se ha decidido, por la amplia información que aportan sobre la estructura armónica de prácticamente cualquier pieza musical, emplear un vocabulario de acordes reducido en los que solo emplearemos dos escalas, las antes mencionadas escala diatónica mayor, y diatónica menor.

Este patrón de saltos de tonos y semitonos se podrá aplicar a cada una de las 12 alturas, formando así 24 tipos de escalas. Éstas recibirán la nomenclatura de raíz + tipo, siendo la

raíz la nota de comienzo, y el tipo el patrón de saltos empleado.

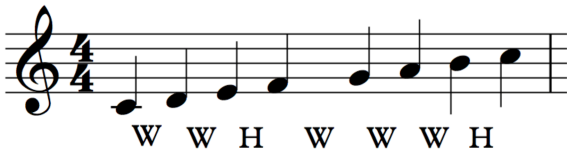

Figura 8: Patrón mayor de saltos de intervalo. *W = 1 tono, H = 1/2 tono*

http://www.graftonmusic.blogspot.com.es

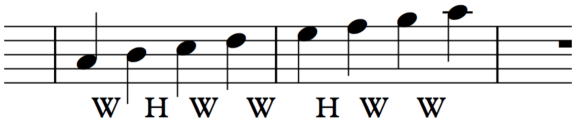

Figura 9: Patrón menor de saltos de intervalo. *W = 1 tono, H = 1/2 tono*

http://www.graftonmusic.blogspot.com.es

A sí mismo, cada una de las notas producidas por los saltos de la escala tendrán un grado con respecto a la tónica (raíz de la escala) representado como un número ordinal. De este modo, cuando hablemos de la segunda de C (Do), estaremos refiriéndonos a D (Re).

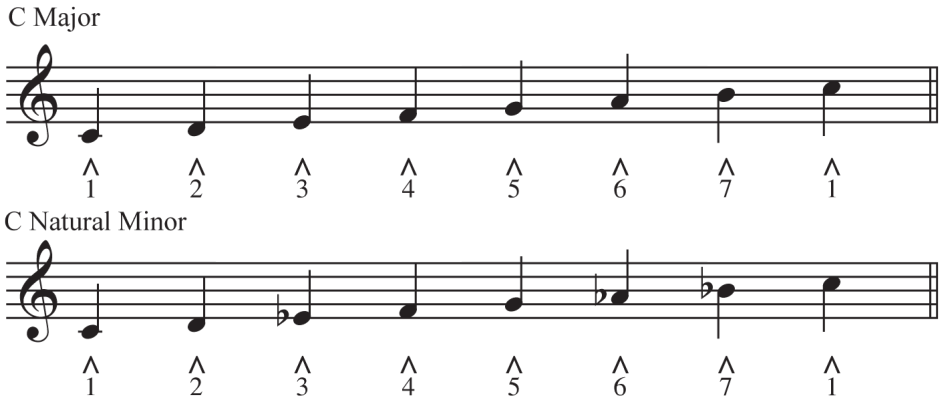

Figura 10: Comparación entre las escalas de C mayor y C menor. Fuente:

http://www.robertabatemusic.com

Como se puede observar, y a pesar de que muchas de las notas de ambas escalas coincidan, siempre diferirán en su 3º, 6º y 7º grado, siendo éstos medio tono inferiores cuando se compare la escala diatónica menor con la mayor.

2.4. Acordes

Un acorde se puede definir como un conjunto de dos o más notas que suenan simultáneamente. Con una definición tan amplia, cabe pensar que exista una infinidad de posibilidades de distintas formaciones de acordes, y así es. De cualquier modo, en lo que atañe a este trabajo, solo pondremos el foco en los acordes de triada diatónica mayor y menor.

La palabra triada hace referencia al uso de tres notas en la formación del acorde, y el hecho de que sea diatónica mayor o menor tendrá que ver con la escala musical en la que está basado. Así, un acorde de triada diatónica mayor o menor estará definido por tres de las notas pertenecientes a su correspondiente escala, siendo éstas la tónica (raíz del acorde), su tercera, y su quinta.

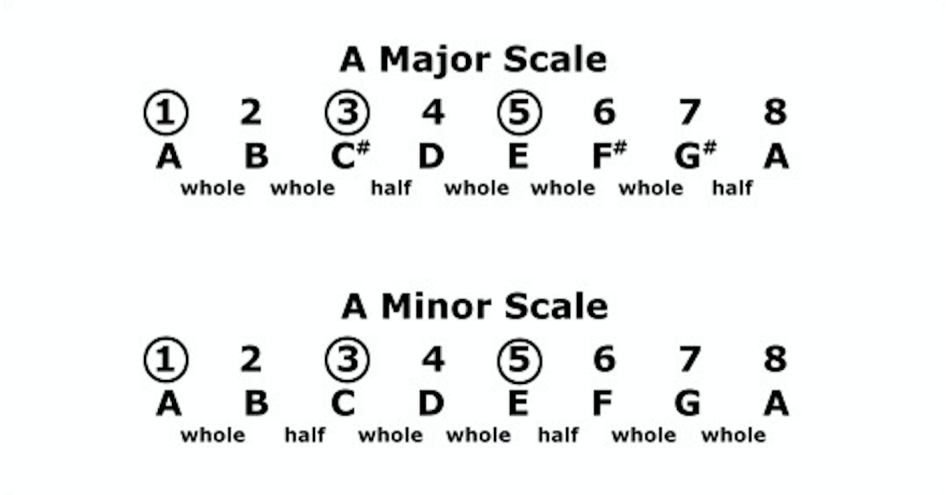

Figura 11: Comparación entre los acordes de triada de A (La) mayor y menor.
http://www.ibreathemusic.com

Como consecuencia, y teniendo en cuenta que las únicas notas que difieren entre la escala diatónica mayor y la diatónica menor son la 3ª, 5ª y 6ª, la única nota que variará entre un acorde de triada mayor y uno de triada menor, será su tercera. Por tanto, la tercera será uno de los elementos diferenciadores a la hora de determinar cuando un acorde es mayor o menor.

2.5. Teorema y transformada de Fourier

Tal y como se describió en la sección 2.1, el teorema de Fourier argumenta de forma matemática que una señal compleja puede descomponerse como suma de muchas señales simples sinusoidales, o lo que es lo mismo su espectro de frecuencias. Esto permite, me-

diante su transformada, transportar de forma reversible una señal del dominio temporal al frecuencial sin producirse pérdidas de información.

Esta demostración matemática es la que ha permitido que las técnicas de procesado de señales avancen como lo han hecho hasta el día de hoy, ya que la mayoría de ellas solo tienen sentido en el ámbito frecuencial. Pero, a la hora de hablar de la transformada de Fourier, tendremos que tener en cuenta que las señales con las que trataremos serán digitales, y por tanto discretas. De este modo, una señal digital no será continua en el tiempo, si no que estará muestreada temporalmente. Este muestreo determinará la resolución de la señal, y estará definido por la frecuencia de muestreo (f_s), la cual expresa el número de muestras que se harán por segundo y típicamente es de 44.1 kHz en audio (se tomarán 44100 muestras por segundo).

Para el caso concreto de señales digitales, se emplea la transformada discreta de Fourier (DFT), la cual queda definida por la siguiente expresión:

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

De este modo una señal con N muestras temporales se transformará en una señal con N muestras frecuenciales distribuidas uniformemente entre 0 Hz y f_s que definirá su espectro muestreado.

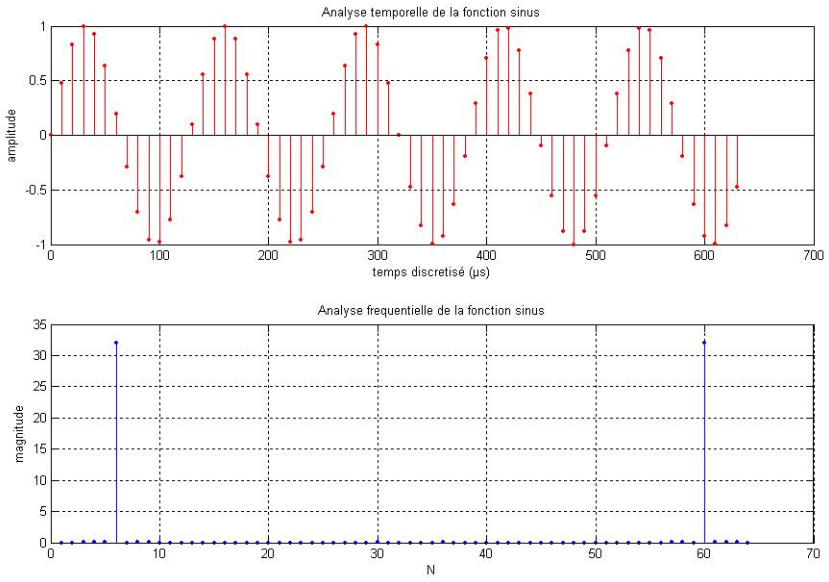

Figura 12: Transformada discreta de Fourier de un seno. `http://www.wikipedia.fr`

De forma análoga a cómo ocurre con la transformación de espacio temporal a frecuencial, se puede realizar la operación inversa (IDFT):

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n \mathrm{e}^{-\frac{2\pi i}{N}kn} \qquad k = 0, ..., N-1 \tag{3}$$

Un factor a tener muy en cuenta a la hora de realizar la DFT sobre una señal es que, debido a que tanto los cálculos capaces de ser procesados por un ordenador, como la duración de una señal digital son finitos, será necesario enventanar temporalmente la señal para así delimitar el número de muestras en los que se realiza la transformada cada vez.

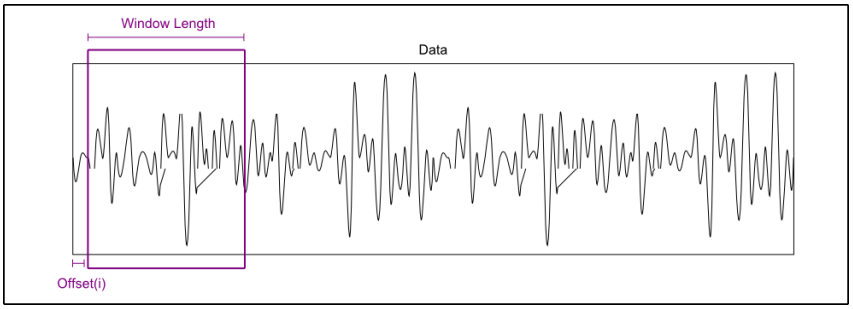

Figura 13: Enventanado de una señal.

http://www.kevinsprojects.wordpress.com

Instintivamente podemos pensar que una ventana temporal necesariamente tiene que ser rectangular, pero este tipo de ventanas producen un efecto llamado leakage que produce una serie de frecuencias indeseadas sobre el espectro analizado, siendo este ruido consecuencia de las discontinuidades en los bordes de la ventana. Es así cómo surgen entonces otros tipos de ventana (como puedan ser las ventanas tipo Hamming o Blackman) que precisamente pretenden suavizar el efecto producido por el leakage.

Escoger la ventana adecuada, será pues un compromiso entre resolución frecuencial y resolución temporal al que se tendrá que llegar con el fin de favorecer lo máximo posible el procesado y análisis de la señal.

2.6. Cromagrama

El núcleo de la representación del audio en la mayoría de los sistemas modernos de reconocimiento automático de acordes es el cromagrama (G.Wakefield, 1999). Éste se podría definir como un espectrograma en el que se analiza, dentro de una ventana temporal, la

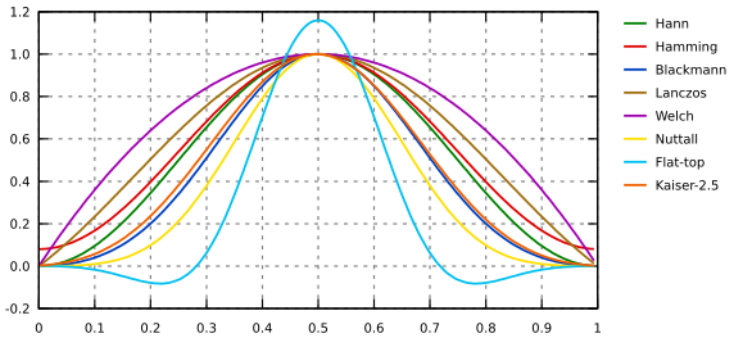

Figura 14: Diferentes tipos de ventana. `http://www.gwyddion.net`

energía en diferentes bandas de frecuencias centradas donde se encontrarían las alturas de las diferentes notas de la gama cromática armónica.

Esta herramienta es de vital importancia para la extracción de información musical ya que sirve como método de análisis, ofreciendo información específica acerca de la gama cromática armónica de la señal de audio que se pretende procesar. Más concretamente, en el ámbito del reconocimiento automático de acordes resulta absolutamente imprescindible, ya que precisamente lo que pretenderemos identificar serán las notas presentes en un acorde para poder categorizarlo.

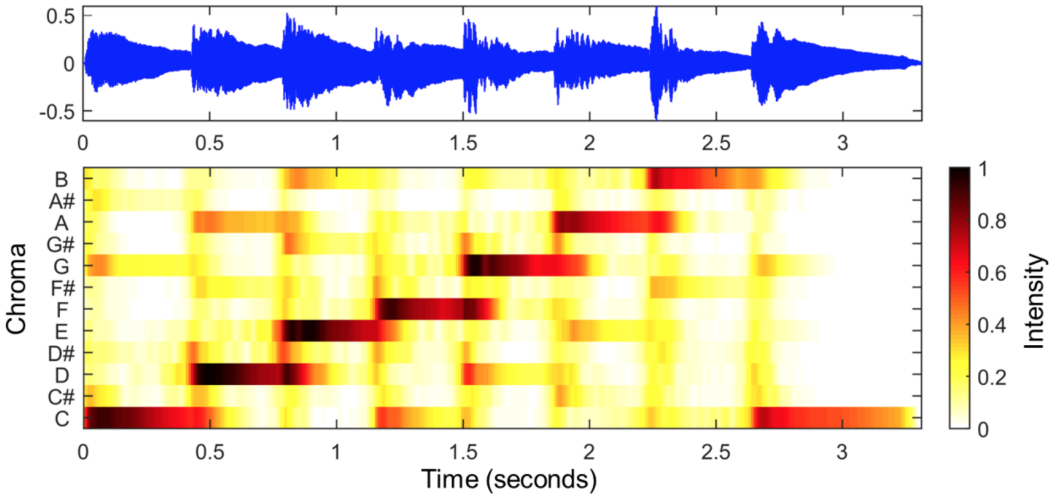

Figura 15: Cromagrama de una señal de audio. `http://www.wikipedia.org`

El cromagrama, se expresará como una matriz de 12 elementos, conteniendo cada uno de ellos el valor energético de cada una de las bandas pertenecientes a cada altura, por el número de muestras obtenidas del enventanado de la señal. Así pues, se podrá disponer de

un análisis discreto de la señal con la información armónica de cada uno de los instantes de tiempo definidos por la resolución.

2.7. Aprendizaje automático no supervisado

Debido a que la interpretación de un cromagrama se puede englobar al fin y al cabo dentro del reconocimiento de patrones, se ha considerado el uso de técnicas de aprendizaje automático.

El aprendizaje automático es una rama de la inteligencia artificial que pretende que un sistema informático sea capaz de generalizar comportamientos y aprender. Más concretamente, el aprendizaje automático no supervisado es un modelo en el que las variables de entrada se considerarán completamente aleatorias, construyendo así un modelo de densidad de datos.

Los métodos de clustering son, a su vez, procesos por los cuales se pretende agrupar las variables de entrada en conjuntos en función de algún criterio, como pueda ser la distancia entre los puntos. De este modo se define k-means, uno de los algoritmos de clustering más conocidos y empleados.

k-means, como algoritmo de clustering, pretende agrupar los valores de entrada sin tener, a priori, ninguna condición que le sirva de información para la clasificación más que el número de agrupaciones que se desean (representado con la letra k). Este algoritmo es ampliamente utilizado en procesos de identificación y selección de imágenes, aunque su uso se puede extrapolar a muchas otras situaciones, como la que concierne al presente trabajo.

Así se definen los pasos que sigue el algoritmo:

1. Se estipulan aleatoriamente k centroides.

2. Genera k grupos en función de la distancia de cada variable con el centroide.

3. Recalcula el centroide de los nuevos k grupos generados.

4. Se repiten los pasos 2 y 3 hasta que el resultado alcance la convergencia.

Lo realmente interesante de este algoritmo es que, recibiendo únicamente los valores de entrada y la condición impuesta del número de grupos, es capaz de generar agrupaciones bastante coherentes.

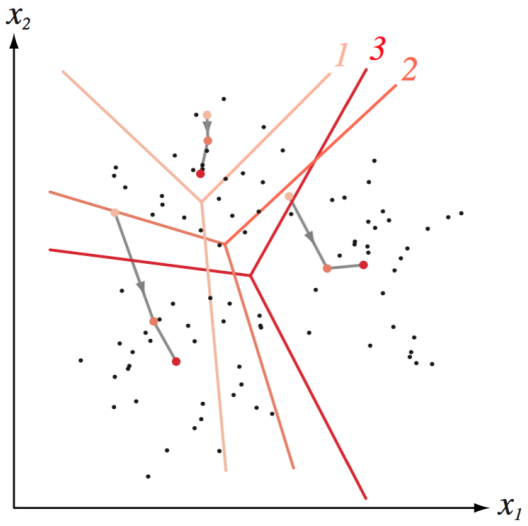

Figura 16: Evolución de agrupaciones realizadas por k-means. DLSI-UA, 2013
Los puntos negros representan las variables de entrada, los puntos a color y las líneas representan la evolución de los centroides y las fronteras de los clusters a través de las diferentes iteraciones.

Puesto que la finalidad de implementar el algoritmo k-means proviene de la necesidad de crear agrupaciones compactas, se define como elemento de decisión la función J. Ésta representa el sumatorio de distancias de cada una de las variables de entrada con respecto a su centroide correspondiente. Así pues, a menudo que se empleen valores superiores de k, menor será la función J.

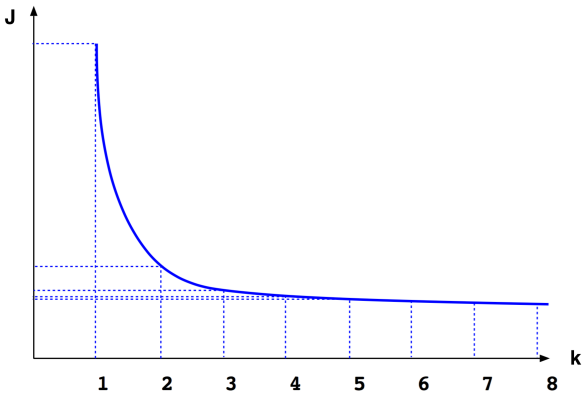

Figura 17: Representación de la función J para 8 valores de k. DLSI-UA, 2013

Consecuetemente, y pretendiendo llegar a un compromiso entre número de agrupaciones y compacidad en éstas, se define el punto en el que la función tiende al eje de abscisas (coloquialmente el codo de la función) como valor idóneo de k.

3. Descripción del sistema

Esta sección compondrá, con sus múltiples subsecciones, casi la totalidad del cuerpo del trabajo. En ella se abordarán todos los detalles relacionados con el diseño del sistema de reconocimiento de acordes, así como la toma de decisiones y diseños alternativos fruto de los obstáculos encontrados en su realización.

3.1. Corpus empleado

Puesto que se pretende diseñar un sistema preciso de reconocimiento de acordes en señales de audio, el formato de codificación de los ficheros de la base de datos deberá cumplir con unos requisitos de calidad mínimos para poder realizar los procesos de forma eficaz, y sin problemas derivados del ruido o una baja resolución espectral. Por este motivo se ha escogido el formato WAVE, desarrollado por IBM y Microsoft.

Los ficheros WAVE (con extensión .wav) se encuentran en formato PCM, lo que garantiza una codificación sin pérdidas, siendo éste el principal motivo de su elección para el trabajo desarrollado. Más concretamente, los ficheros pertenecientes al corpus estarán indistintamente en mono o estéreo puesto que el procesado implícito en el sistema hará una conversión a un solo canal. Además tendrán una frecuencia de muestreo de 44.1 kHz, y una resolución de 16 bits (estándar impuesto para el CD).

3.1.1. Vocabulario armónico

Simplemente recordando la definición de acorde (sección 2.4) podemos hacernos una idea de la cantidad de posibilidades de acorde fruto de la probabilidad podríamos encontrarnos. Puesto que el sistema diseñado pretende ofrecer una estimación que sirva de análisis general, nos encontramos con la necesidad de acotar el vocabulario a los objetivos del presente trabajo.

Con el fin de no añadir complejidades añadidas por el uso de un vocabulario armónico que intente definir demasiado en profundidad los acordes estimados, produciendo de este modo mayor probabilidad de error, en parte por el uso de acordes con cromagramas poco definidos como puedan ser cuatriadas como los acordes de séptima dominante (acordes con cuatro notas), se ha optado por la contemplación exclusivamente de acordes de triada diatónica mayor y menor (referidos a partir de este punto como acorde mayor y acorde menor).

Cabe resaltar que, a pesar de ser un vocabulario armónico no demasiado amplio, la mayoría de las piezas musicales pertenecientes a los géneros más populares como puedan ser el Pop o el Rock cuentan con un vocabulario de acordes que, en su mayoría, se encuentran dentro del contemplado en este trabajo.

Consecuentemente, dispondremos de un vocabulario de 24 acordes, en los que tendremos definido un tipo mayor y un tipo menor para cada una de las 12 alturas, siguiendo la sintaxis de etiquetado propuesta por C.Harte (2010).

Tabla 1: Etiquetas de acorde empleadas

Raíz	Etiqueta	Tipo	Etiqueta
Do	C	Mayor	maj
Do#	C#	Menor	min
Re	D	Silencio	N
Re#	Eb		
Mi	E		
Fa	F		
Fa#	F#		
Sol	G		
Sol#	Ab		
La	A		
La#	Bb		
Si	B		

Ejemplo: Do mayor = C:maj

3.1.2. Ficheros de instrumentos sintetizados

La decisión de emplear ficheros de audio procedente de la síntesis digital proviene de la motivación del trabajo. Puesto que se pretende diseñar un sistema que ofrezca resultados precisos de estimación de acordes mediante la clasificación de los patrones en familias de instrumentos, resulta casi inevitable pensar en lo increíblemente costoso que resultaría obtener ficheros de audio de una amplia variedad de instrumentos. Es por este motivo por el

que el groso del corpus empleado en este trabajo está compuesto por audio sintetizado desde Csound.

Con el fin de disponer de resultados representativos de un amplio espectro de sonidos, se han utilizado dos motores de síntesis General Midi (llamados soundfonts), siendo uno de ellos más sencillo y minimalista, produciendo una síntesis no muy compleja y menos realística (TimGM6mb), y un soundfont de mayor complejidad y realismo (FluidR3sGM2). Además se han sintetizado ficheros de audio de diversos instrumentos pertenecientes a varias de las familias de instrumentos musicales más representativas (como puedan ser guitarras, pianos, cuerdas), así como otros instrumentos musicales menos populares, pero con un espectro o envolvente muy característicos (como pueda ser el caso de la marimba, perteneciente a la familia de los ideófonos).

3.1.3. Ficheros de instrumentos reales grabados

Los ficheros de instrumentos sintetizados resultan de gran utilidad para diseñar y evaluar el sistema propuesto, pero no hay que olvidar que éstos, a pesar de ser fieles recreaciones en muchos de los casos, en determinado grado (en función del realismo de la síntesis) diferirá del comportamiento de un instrumento real. Por este motivo, además de evaluar nuestro sistema con ficheros de audio sintetizado, también se hará con ficheros de audio procedentes de grabaciones de instrumentos reales, probando así su eficacia en una situación más acorde con sus posibles aplicaciones directas.

De este modo, se ha seleccionado parte de la base de datos de grabaciones de guitarra, acordeón y violín perteneciente al repositorio del trabajo sobre reconocimiento de acordes mediante redes neuronales de Julien Osmalskyj (University of Liège, Belgium).

Así, y disponiendo de una base de datos con ficheros de audio de múltiples instrumentos sintetizados, así como ficheros de audio de grabaciones de algunos de los instrumentos más representativos de la música occidental actual, podremos considerar que el corpus del trabajo será lo suficientemente diverso como para poder extraer conclusiones significativas de los resultados.

3.2. Necesidad de automatización

Puesto que para poder extraer conclusiones significativas de la evaluación del sistema de reconocimiento de acordes se tendrá que probar con múltiples señales de audio de distinta

índole, será necesario desarrollar el programa desde un enfoque automatizado en el que éste se ejecute de forma automática para todos los ficheros que componen el corpus.

Así pues, no solo se ejecutará el programa de forma automática, si no que además se reorganizarán todos los ficheros generados en el proceso de estimación y evaluación (cromagramas, ficheros de estimación, hojas de resultados...), depositando cada uno de ellos en la carpeta correspondiente de un directorio jerarquizado y clasificado. De este modo, la consulta de cualquier fichero generado durante la ejecución del programa resultará muy sencillo, facilitando así el análisis de los resultados obtenidos.

3.3. Sistema inicial

El desarrollo del primer sistema funcional referente a este trabajo está desarrollado, en gran parte, como un esqueleto modular que permita el desarrollo del sistema definitivo añadiendo otros sub-sistemas paralelos. Por tanto, el sistema inicial establece los procesos básicos necesarios para la estimación automática de acordes mediante comparación de patrones de cromagrama, pero no mediante clasificación instrumental.

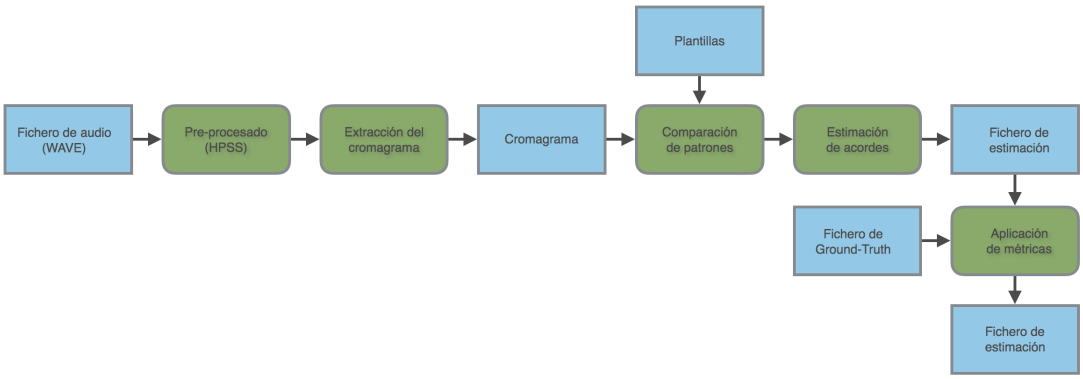

Figura 18: Diagrama de bloques del sistema inicial de reconocimiento de acordes. Fuente: Elaboración propia

3.3.1. Algoritmo de separación armónico-percusivo

Puesto que la extracción del cromagrama nos ofrecerá información armónica acerca de la señal analizada, el hecho de poder aislar su contenido armónico del percusivo, que como

norma general no será más que ruido representado en el cromagrama, resulta absolutamente interesante.

Incluso en señales de audio mono-tímbricas (un solo instrumento) como las que componen el corpus de este trabajo, existen componentes percusivas. Pongamos por ejemplo la pulsación de una tecla de piano. Podemos creer que el sonido producido por la pulsación será completamente arómico, pero no sería cierto. El hecho de que la cuerda vibre por el golpeo de una maza implica que en los primeros milisegundos de vibración de la cuerda, ésta se encuentre en un estado transitorio en el que el contenido espectral no es para nada armónico. Es en los siguientes milisegundos cuando la cuerda estabiliza su vibración en función de su longitud y tensión cuando su espectro será armónico, otorgándole así determinada altura musical. Como consecuencia de todo ello, se ha optado por la inclusión de un algoritmo de separación armónico-percusivo (HPSS).

El algoritmo ha sido desarrollado por Jonathan Driedger, perteneciente al toolbox distribuido por Audiolabs (`http://www.audiolabs-erlangen.de/resources/MIR/TSMtoolbox/`). Su funcionamiento no resulta demasiado complicado, y se puede resumir en pocos pasos.

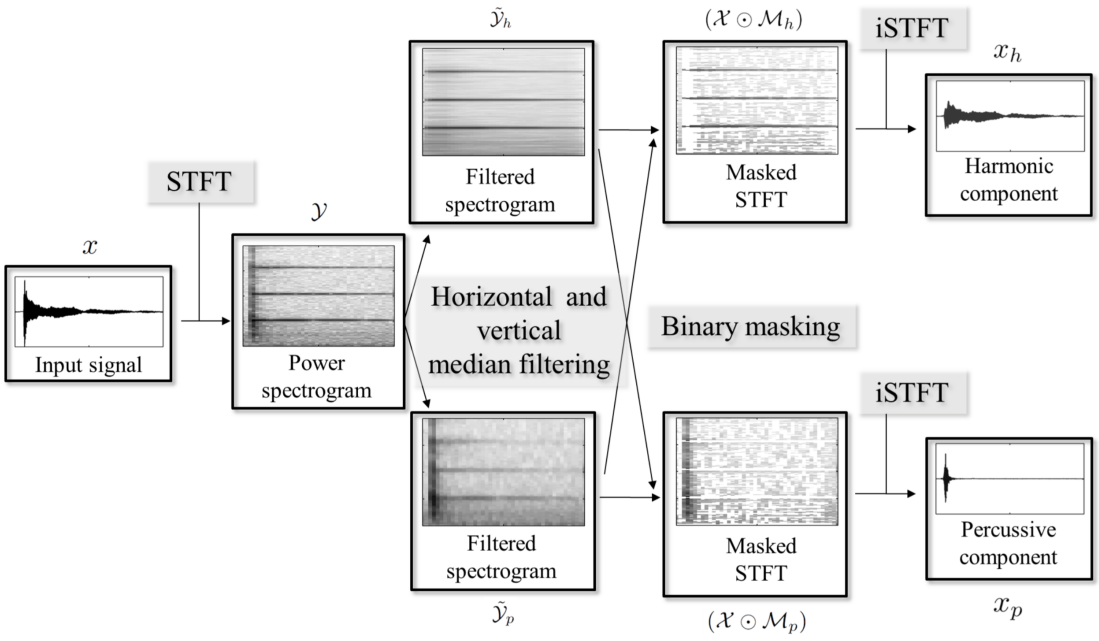

Figura 19: Diagrama de bloques del algoritmo HPSS. Fuente: DRIEDGER, 2014

Inicialmente se pasará la señal del dominio temporal, al dominio frecuencial, utilizando para ello la transformada corta de Fourier (STFT), que no es más que la implementación de la

transformada discreta de Fourier empleando para ello una pequeña ventana. Esto dará como resultado el espectrograma de la señal, el cual, y teniendo en cuenta lo antes mencionado, reflejará la parte percusiva (ruido producido en el momento transitorio de la vibración) como un impulso vertical debido a su amplio contenido espectral (casi se podría definir como un instante de tiempo en el que predomina ruido blanco aleatorio), y la parte armónica de la señal quedará reflejada como bandas de frecuencias que perduran a través de la duración de la nota (en el espectrograma quedará reflejado como varias líneas horizontales).

De este modo, y utilizando un filtro de mediana horizontal, se podrán aislar las componentes horizontales del espectro, y con ello el contenido frecuencial armónico de la señal. De forma análoga, pero aplicando un filtro de mediana vertical, ocurrirá lo mismo para las componentes frecuenciales más impulsivas (verticales), aislando así el contenido frecuencial correspondiente a la parte percusiva de la señal de audio.

Posteriormente se aplica una máscara binaria a cada uno de los dos espectros para realizar las asignaciones correctamente, y después se devolverían al dominio temporal mediante la transformada corta inversa de Fourier (iSTFT) para así disponer de las dos señales de audio aisladas.

3.3.2. Sonic Annotator y extracción de cromagrama

El elemento que nos permitirá obtener información armónica sobre la señal analizada, y por tanto podemos considerar como la pieza fundamental de este trabajo, es el cromagrama. Por tanto, su obtención será uno de los procesos relevantes del sistema.

Para realizar dicha tarea, se ha empleado la herramienta de distribución libre Sonic Annotator. Desarrollado por la Universidad Queen Mary de Londres, Sonic Annotator permite la extracción y anotación (de ahí su nombre) de características de señales de audio a través de su interfaz, mediante Vamp plugins.

Estos plugins funcionan de forma idéntica a los conocidos plugins VST, pero en lugar de aplicar efectos de audio, extraen información descriptiva de la señal como pueden ser instantes de onset, espectrogramas, o cromagramas.

Sonic Annotator se ejecuta a través de la consola de comandos de cualquier ordenador con sistema operativo Windows, Mac OS X o Linux (en este caso se ha ejecutado desde la consola de Mac OS X) mediante unas directrices específicas descritas en la línea de comando donde se ejecuta la aplicación.

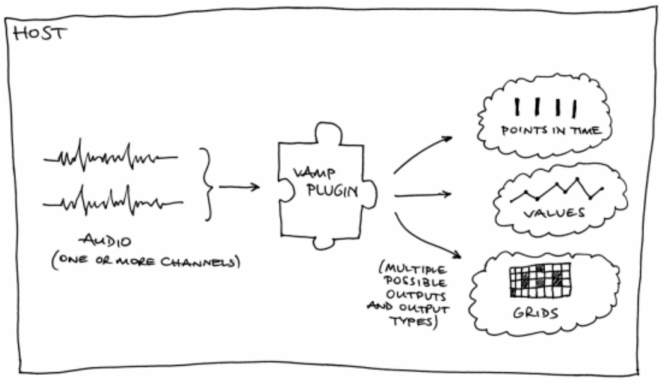

Figura 20: Funcionamiento de los plugins Vamp. Fuente:
`http://www.vamp-plugins.org`

./sonic-annotator -t config.txt fichero.wav -w csv

Uno de los requisitos indispensables para poder ejecutar Sonic Annotator es la definición de un documento de texto donde se configurará del plugin Vamp que se pretende utilizar. El programa extraerá los parámetros necesarios para definir los procesos implícitos en la ejecución del plugin como puede ser la longitud de la ventana de análisis, o el paquete que se desea utilizar.

De este modo, y para cumplir nuestro objetivo de obtener el cromagrama de una señal para la estimación posterior de sus acordes, se ha definido en el fichero de configuración (llamado config.txt en nuestro sistema) la utilización del plugin NNLS-chroma (MATTHIAS MAUCH, 2010). La decisión de usar el anteriormente citado plugin en lugar de cualquier otro destinado a la obtención del cromagrama se debió, en primer lugar a la popularidad del mismo, así como a la posibilidad de obtener al mismo tiempo un segundo cromagrama (llamado bass chroma) que ofrece información de cromagrama de notas más graves. A pesar de haber valorado la opción de utilizar complementariamente ambos cromagramas, aportando el cromagrama grave información significativa que podría solucionar determinados errores producidos en acordes invertidos, finalmente no se llevó a cabo.

El algoritmo NNLS Chroma analiza un único canal de audio, transformando el espectro en un espectro de frecuencias logarítmico dividido en tres bandas por semitono para poder centrar la frecuencia central a cada altura, aunque la afinación se desvíe un poco de la afinación estándar (440 Hz). La salida entonces será un espectro de semitonos, el cual se multiplicará por el perfil de cromagrama, quedando entonces mapeado a 12 elementos discretos.

(`http://www.isophonics.net/nnls-chroma`)

De este modo, otro de los parámetros que será necesario definir en el documento de configuración será la longitud de la ventana que defina el número de muestras cada cuanto se calculará el cromagrama, delimitando consecuentemente la resolución temporal del cromagrama obtenido. Así pues, y teniendo en cuenta que la frecuencia de muestreo de los ficheros de entrada será de 44.1 kHz, y que con una resolución temporal de 0,05 segundos se tendría un análisis lo suficientemente fino para la práctica a llevar a cabo (los cambios de acorde de cualquier pieza musical siempre se producirán en pasos temporales muy superiores), se calcula el siguiente tamaño de ventana:

$$N = f_s * \Delta t = 44100 \cdot 0{,}05 = 2205 \Longrightarrow 2048\ muestras \tag{4}$$

Una vez definidos todos los parámetros necesarios, podremos ejecutar Sonic Annotator. Puesto que el sistema de detección de acordes propuesto está basado en el entorno MATLAB, y teniendo en cuenta que gracias al comando system podremos ejecutar comandos de consola desde MATLAB, la ejecución de Sonic Annotator quedará completamente automatizada e integrada en nuestro sistema.

Como resultado de todo ello, obtendremos un fichero en formato CSV (Comma Separated Vector) que contendrá un vector de 13 elementos (siendo el primero el instante temporal, y los otros 12 los elementos pertenecientes a cada una de las alturas del croma) por cada uno de los frames de análisis (resolución temporal consecuencia del tamaño definido de la ventana), que representará el cromagrama de la señal de audio procesada.

3.3.3. Generación de plantillas

Puesto que el sistema propuesto se basará en la comparación de patrones de cromagrama para la estimación de acordes, será necesario definir un patrón de cromagrama que sirva como plantilla para la identificación de cada uno de los acordes.

Así, y pensando en la definición de acorde descrita en el marco teórico del trabajo, definimos un vector de doce elementos que representaría cada una de las alturas del cromagrama, en el que se tendría un valor absoluto normalizado para cada altura en función de la definición del acorde. De este modo, se define un vector binario que definirá los acordes mayores, y otro que lo hará con los menores:

Puesto que los acordes mayores o menores, sea cual sea su raíz guardarán la sucesión de

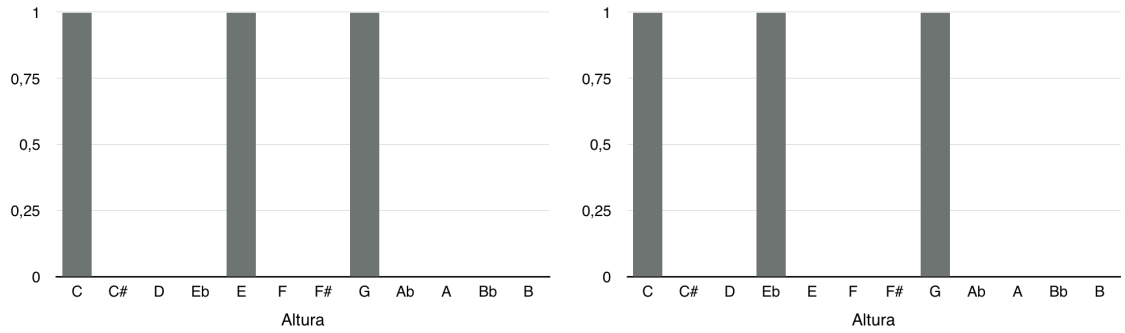

Figura 21: Plantillas binarias de C mayor a la izquierda, y de C menor a la derecha. Fuente: Elaboración propia

saltos definida (ver sección 2.4), con el simple hecho de rotarlos podremos obtener la plantilla correspondiente a cada uno de los 24 acordes de nuestro vocabulario armónico contenidas en una matriz de 12x12x3 (12 alturas para cada plantilla, por 12 raíces de acorde, y por 2 tipos de acorde y el silencio (será necesario definir también el silencio como un vector de ceros).

$$\{1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\} \Rightarrow \{0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\}...$$

Estas plantillas representan clara e inequívocamente la definición de todos los tipos de acordes contemplados, pero la realidad, incluso desde un enfoque ideal, no es así. Cada nota, individualmente, y a pesar de tener una altura marcada por su frecuencia fundamental, dispone también de una serie de armónicos. Múltiplos enteros de la frecuencia fundamental que se repiten de forma decreciente. Además, y teniendo en cuenta que el análisis de cromagrama que realiza el algoritmo NNLS pliega todas las frecuencias contenidas en el espectro, contribuyendo de este modo los armónicos múltiplos de la frecuencia correspondiente a cada elemento de cromagrama, resultará interesante la definición de unas plantillas alternativas a las binarias que representen mejor la realidad del comportamiento armónico de la señal.

Así, y con el fin de tener unas plantillas un poco más realísticas, se calcula el peso de cada uno de los elementos, para la superposición decreciente de los 11 primeros armónicos. Además, las rotaciones correspondientes se realizarán del mismo modo que para las plantillas binarias, disponiendo de forma análoga a las plantillas binarias, una vez más de una matriz de 12x12x3 que contendrá todos los acordes del vocabulario armónico empleado.

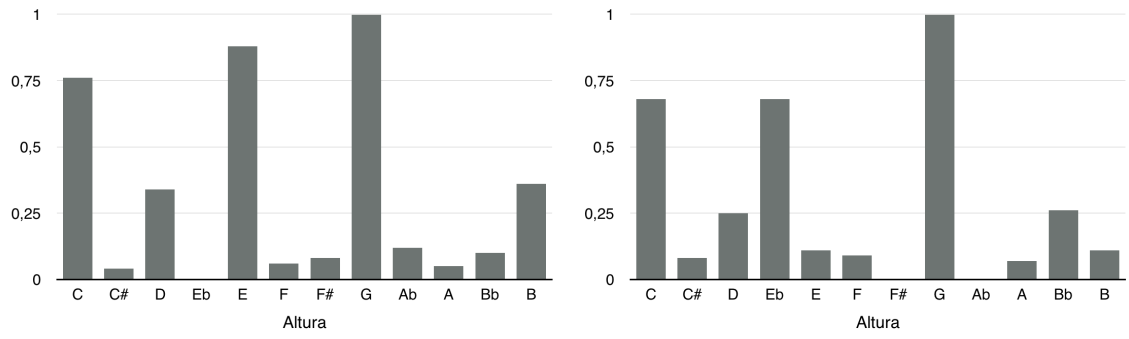

Figura 22: Plantillas con 11 armónicos de C mayor a la izquierda, de C menor a la derecha. Fuente: Elaboración propia

3.3.4. Comparación de patrones

Llegados al punto en el que disponemos del cromagrama de cada uno de los instantes temporales de la señal de audio que pretendemos analizar, además de unas plantillas que definen cada uno de los acordes presentes en nuestro vocabulario armónico, la comparación de similitud entre ambos elementos será la que decida las estimaciones realizadas.

Empleando un enfoque mucho más formal, se podría decir que se dispone de un espacio dodecadimensional (12 dimensiones) en el que en cada una de estas dimensiones se representaría el peso característico de cada uno de los elementos del cromagrama. Así, se dispondría de un hipercubo perteneciente a dicho espacio, el cual estaría delimitado por los vértices que formarían las plantillas. En este espacio multidimensional se localizarían cada uno de los cromagramas obtenidos del análisis como puntos dentro del hipercubo, calculando posteriormente la distancia euclídea de cada uno de estos puntos (vectores de cromagrama), con los vértices del hipercubo (plantillas).

De este modo, la decisión dependerá de la similitud entre los vectores de cromagrama y las plantillas generadas, siendo el parámetro que exprese dicha similitud su distancia euclídea.

$$d_E(C, P) = \sqrt{\sum_{i=1}^{12} (c_i - p_i)^2} \tag{5}$$

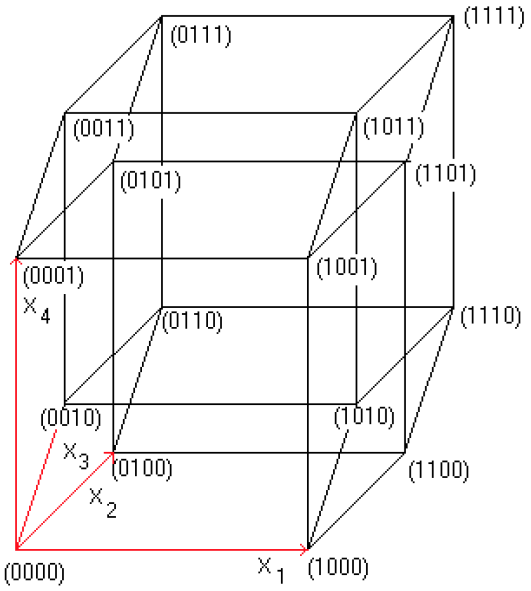

Figura 23: Hipercubo tetradimensional. Fuente:
`http://www.mathematische-basteleien.de`

3.3.5. Estimación de acordes

Una vez obtenida la distancia euclídea entre los vectores de cromagrama corresopondientes a cada uno de los instantes temporales para los que se ha calculado, con las plantillas definidas, la labor de asignar un acorde u otro estará exclusivamente reñida a la distancia mínima entre ellos. Es decir, el vector de plantilla que se encuentre más próximo al vector de cromagrama que se pretende analizar, será el que defina el acorde estimado.

Consecuentemente, una vez hallada la posición del vector de la plantilla que es más próxima al vector de cromagrama, se creará un fichero de estimación en CSV en el que se escribirá el instante temporal correspondiente al vector de cromagrama analizado, y las etiquetas correspondientes al acorde asignado (siguiendo el formato descrito en la sección 3.1.1).

3.4. Sistema basado en clasificación instrumental

La clasificación instrumental forma parte del título de este trabajo, no por menos que por ser uno de sus principales objetivos. Así pues, se pretende mejorar la precisión a la hora de identificar acordes del sistema inicial, creando una serie de plantillas específicas que sean capaces de adaptarse a las características armónicas de los instrumentos mejor que las

plantillas anteriormente propuestas.

3.4.1. Plantillas de familias de instrumentos GM

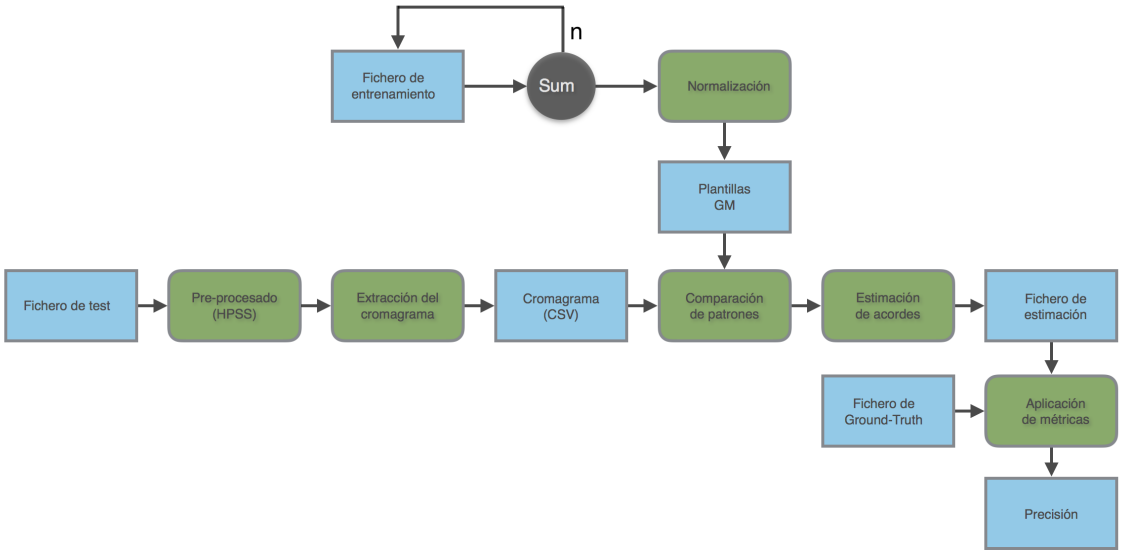

Figura 24: Diagrama del sistema basado en clasificación instrumental GM. Fuente: Elaboración propia

La idea de generar plantillas clasificadas por familias de instrumentos viene dada por la hipótesis de que, debido a que las características físicas de un instrumento (construcción, materiales, naturaleza de la vibración, etc) influyen en su espectro característico, y por tanto en su cromagrama, instrumentos de una misma familia deberán compartir determinadas similitudes armónicas. Por tanto, estas similitudes afectarían directamente al tipo de patrón de su cromagrama, pudiéndose generalizar para obtener plantillas específicas de familias de instrumentos que se pudiesen emplear en función de la familia instrumental a la que perteneciese la señal de audio que se pretende analizar en el sistema.

Así, y bajo la recomendación de José Manuel Iñesta (tutor del presente trabajo), se propone una clasificación basada en la asignación de bancos desarrollada por General Midi en 1991, contemplando exclusivamente las diez primeras familias (puesto que ampliar aún más el número de familias complicaría excesivamente el experimento).

Las familias de instrumentos que se utilizarán para el desarrollo de las plantillas serán pues: piano, percusión cromática, órgano, guitarra, bajo, cuerdas/orquesta, ensemble, viento-

metal, lead y viento-madera. Se ignorarán por tanto el resto de familias ya que, entre otros motivos, muchas de ellas resultan lo suficientemente similares en cuanto a tono y características de envolvente como para requerir de una nueva categoría.

Piano

1 - Piano 1
2 - Piano 2
3 - Piano 3
4 - Honky-Tonk Piano
5 - Electric Piano 1
6 - Electric Piano 2
7 - Harpsichord
8 - Clav.

Chromatic percussion

9 - Celesta
10 - Glockenspiel
11 - Music Box
12 - Vibraphone
13 - Marimba
14 - Xylophone
15 - Tubolar Bell
16 - Santur

Organ

17 - Organ 1
18 - Organ 2
19 - Organ 3
20 - Church Organ
21 - Reed Organ
22 - Accordion
23 - Harmonica
24 - Bandoneon

Guitar

25 - Nylon-str. Guitar
26 - Steel-str. Guitar
27 - Jazz Guitar
28 - Clean Guitar
29 - Muted Guitar
30 - Overdrive Guitar
31 - Distortion Guitar
32 - Guitar Harmonics

Bass

33 - Acoustic Bass
34 - Fingered Bass
35 - Picked Bass
36 - Fretless Bass
37 - Slap Bass 1
38 - Slap Bass 2
39 - Synth Bass 1
40 - Synth Bass 2

Strings/Orchestra

41 - Violin
42 - Viola
43 - Cello
44 - Contrabass
45 - Tremolo Strings
46 - Pizzicato
47 - Harp
48 - Timpani

Ensemble

49 - Strings
50 - Slow Strings
51 - Synth Strings 1
52 - Synth Strings 2
53 - Choir Aahs
54 - Voice Oohs
55 - Synth Voice
56 - Orchestra Hit

Brass

57 - Trumpet
58 - Trombone
59 - Tuba
60 - Muted trumpet
61 - French Horns
62 - Brass 1
63 - Synth Brass 1
64 - Synth Brass 2

Lead

65 - Soprano Sax
66 - Alto Sax
67 - Tenor Sax
68 - Baritone Sax
69 - Oboe
70 - English Horn
71 - Bassoon
72 - Clarinet

Pipe

73 - Piccolo
74 - Flute
75 - Recorder
76 - Pan Flute
77 - Bottle Blow
78 - Shakuhachi
79 - Whistle
80 - Ocarina

Synth Lead

81 - Square Wave
82 - Saw Wave
83 - Synth Calliope
84 - Chiffer Lead
85 - Charang
86 - Solo Vox
87 - 5th Saw Wave
88 - Bass & Lead

Synth Pad

89 - Fantasia
90 - Warm Pad
91 - Polysynth
92 - Space Voice
93 - Bowed Glass
94 - Metal Pad
95 - Halo Pad
96 - Sweep Pad

Synth FX

97 - Ice Rain
98 - Soundtrack
99 - Crystal
100 - Atmosphere
101 - Brightness
102 - Goblin
103 - Echo Drops
104 - Star Theme

Ethnic

105 - Sitar
106 - Banjo
107 - Shamisen
108 - Koto
109 - Kalimba
110 - Bagpipe
111 - Fiddle
112 - Shanai

Percussive

113 - Tinkle Bell
114 - Agogo
115 - Steel Drums
116 - Woodblock
117 - Taiko
118 - Melo Tom
119 - Synth Drum
120 - Reverse Cymbal

Sound FX

121 - Guitar FretNoise
122 - BreathNoise
123 - Seashore
124 - Bird
125 - Telephone
126 - Helicopter
127 - Applause
128 - GunShot

Figura 25: Listado de instrumentos General Midi. Fuente: `http://www.hispasonic.com`

De este modo, se ha confeccionado una amplia base de datos de ficheros de audio sintetizados que servirán como ficheros de entrenamiento de las plantillas GM (General Midi). Estos ficheros se han generado uno a uno, empleando para ello un programa en Csound que genera, de forma automática, una progresión ascendente de acordes (de C a B) ya sean mayores o menores. Además, y teniendo en cuenta que en función de las características del

instrumento, éste podrá generar silencios entre acordes, se obtendrá también información relativa al Onset y Offset de los acordes sintetizados (ver en el siguiente párrafo).

La intención es clara, generar plantillas generalizadas para familias de instrumentos, y para ello se ha recurrido al cálculo de la media aritmética de los vectores de cromagrama pertenecientes a una misma familia. Adicionalmente, puesto que todos los instrumentos tienen una evolución espectral que hace que no generen el mismo espectro de frecuencias al comienzo de la nota como al final (unas frecuencias se extinguen anteriormente a otras), se calculará la media aritmética de los cromagramas generados por un instrumento durante la duración del acorde.

Por lo tanto, el primer paso será obtener el cromagrama de cada uno de los instrumentos pertenecientes a una familia, después calcular la media aritmética de todos los cromagramas calculados para cada acorde (dentro del rango temporal definido por los Onsets y Offsets), obteniendo así una plantilla generalizada para cada instrumento. Para finalizar se calculará la media aritmética del cromagrama de cada tipo de acorde definido en las plantillas de los diferentes instrumentos.

Por consiguiente, se habrán generado 10 plantillas representativas cada una de ellas de 8 instrumentos pertenecientes a su correspondiente familia. Dicho de otro modo, las familias instrumentales serán conjuntos agrupados en los que cada una de las plantillas representará al centroide (centro de gravedad) de cada agrupación.

3.4.2. Estudio de las agrupaciones

Posteriormente a la generación de las nuevas plantillas, y principalmente movido por su puesta a prueba con algunos ficheros de test, se observó cómo los resultados obtenidos no se maximizaban necesariamente cuando se enfrentaba un fichero de un instrumento contra su correspondiente plantilla. Esto quiere decir, por ejemplo, que la precisión obtenida de la estimación de acordes de un fichero de test de piano no era máxima cuando se enfrentaba a la plantilla correspondiente a la familia de pianos, y sí con la plantilla correspondiente a la familia de guitarras. Por este motivo, y dada la incoherencia de la situación, se propone el desarrollo de un script que automáticamente nos ofrezca algunos parámetros estadísticos que arrojen información acerca de la eficiencia de las agrupaciones realizadas.

Consecuentemente, y generalizando los resultados al acorde de C mayor puesto que sería imposible alcanzar conclusiones fiables si el estudio se realizase para todo el vocabulario

armónico completo, se calcula la distancia euclídea de cada plantilla de instrumentos con respecto a la plantilla correspondiente a su familia de instrumentos (intra-clase), así como la distancia euclídea entre las plantillas de cada familia (inter-clase) para tener una ligera orientación de cuán dispersos puedan estar los instrumentos. Puesto que de los resultados obtenidos se deduce que existen solapes entre las distintas agrupaciones, se estima también la compacidad de las agrupaciones (cuán compactas son), como la suma de las distancias de cada instrumento a su correspondiente centroide (representado por una plantilla GM).

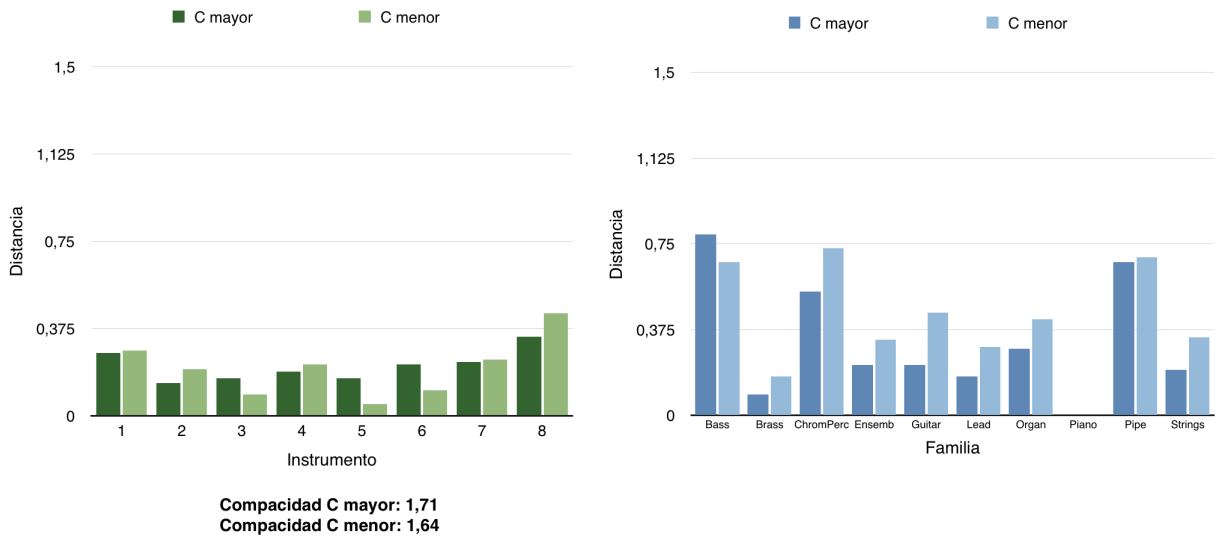

Figura 26: A la izqda: Tabla de distancias intra-clase y valor de compacidad para la familia de pianos / A la dcha: Tabla de distancias inter-clase. Fuente: Elaboración propia

Como se puede observar en la figura 26, la familia correspondiente a los pianos guarda bastante coherencia, no existiendo ningún instrumento perteneciente a la familia que tenga una distancia con respecto al centroide demasiado alta. De todo esto se puede deducir que se trata de una agrupación compacta.

Al contrario que en el caso anterior, y pudiéndose apreciar en la figura 27, para el caso de la familia de percusión cromática se encuentran valores considerablemente superiores. Así mismo, hay dos instrumentos que destacan por estar claramente distanciados del centroide de su agrupación, superando incluso la distancia que guarda esta agrupación con el resto. Esto nos demuestra que la agrupación formada por los instrumentos de percusión cromática no es suficientemente consistente, y muestra una gran dispersión.

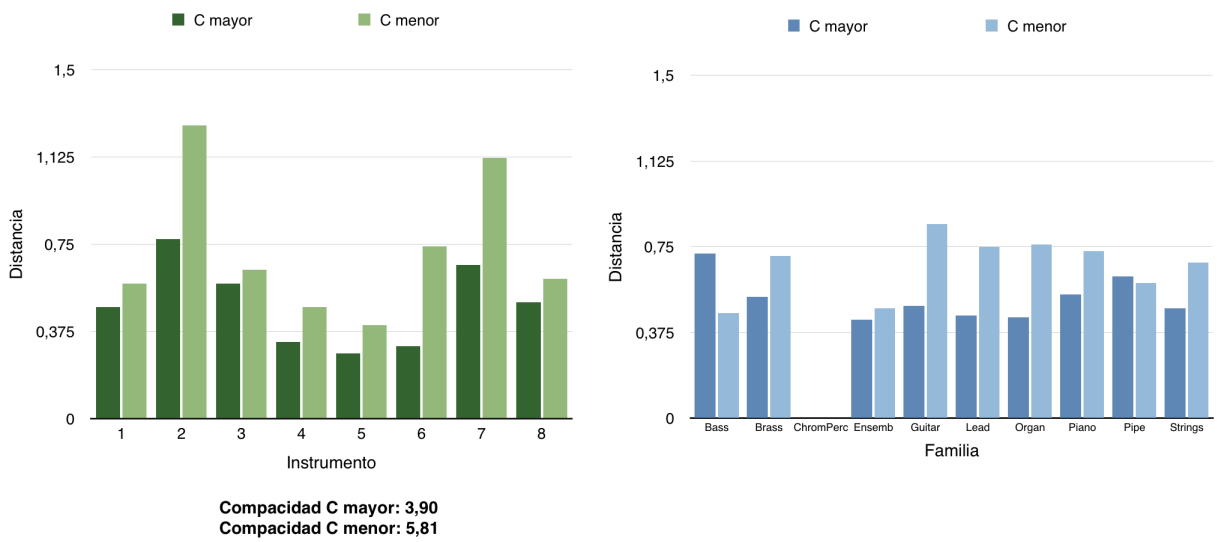

Figura 27: A la izqda: Tabla de distancias intra-clase y valor de compacidad para la familia de percusión cromática / A la dcha: Tabla de distancias inter-clase. Fuente: Elaboración propia

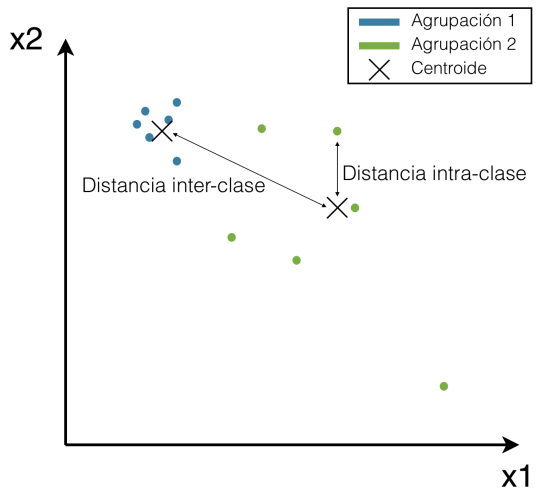

Figura 28: Representación significativa de dos conjuntos en un espacio bidimensional. Fuente: Elaboración propia

3.4.3. Plantillas obtenidas del clustering

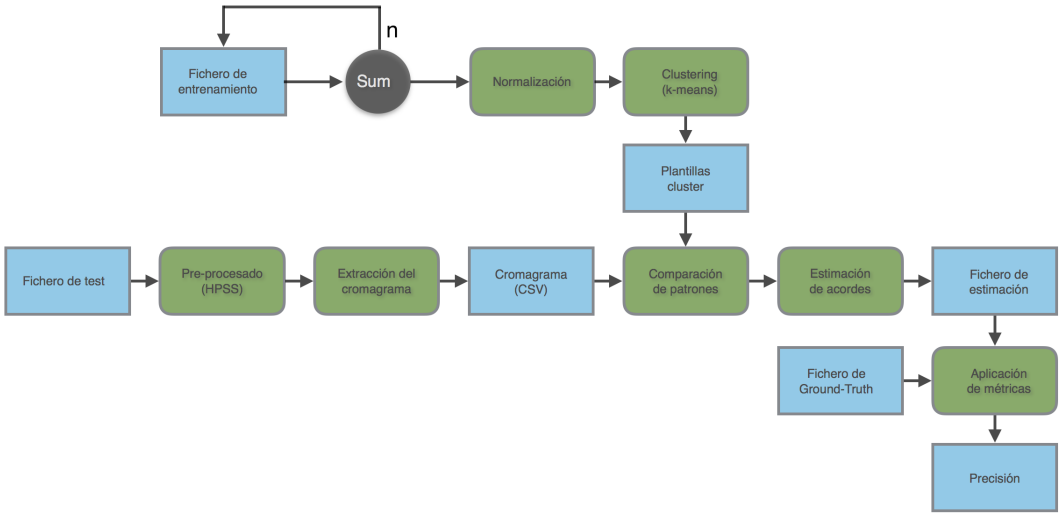

Figura 29: Diagrama del sistema basado en clasificación instrumental mediante clustering. Fuente: Elaboración propia

Como consecuencia de la dispersión que muestran algunas de las familias de instrumentos GM, se propone la implementación de un algoritmo de clustering que confeccione automáticamente unas plantillas de agrupaciones de instrumentos alternativas.

El algoritmo empleado para realizar dicha tarea será k-means, el cual realiza una agrupación sin mayor condición que el número deseado de agrupaciones (expresado como el valor k). Así pues, es necesario definir un vector que exprese las características que queremos que el algoritmo estime para la decisiones de agrupación. De este modo, cada una de las características expresadas en el vector de características será una dimensión dentro del espacio representativo, tomando el algoritmo decisiones de asignación de grupos en función de las representaciones de las variables de entrada y sus distancias dentro de dicho espacio.

Puesto que la finalidad de realizar todo este proceso es el de obtener plantillas de cromagrama de agrupaciones alternativas, y aunque resultaría muy interesante realizar un estudio tímbrico de los ficheros de entrenamiento para describir un vector de características tímbricas, nuestro sistema funcionará con un vector de características armónico que es el cromagrama. Consecuentemente, el algoritmo realizará agrupaciones basadas en un espacio dodecadimensional en el que las agrupaciones sean más compactas que en el caso de las agrupaciones basadas en las familias General Midi.

Al mismo tiempo tendremos que decidir la cantidad de agrupaciones (k) que mejor se adapta a nuestro caso. Para ello se ha obtenido la función J. Siendo esta función dependiente de k, tendremos una idea de cuán compactas son las agrupaciones. El valor de k, para el caso que nos concierne, estará comprendido entre 1 y los 80 instrumentos que conforman la base de datos de entrenamiento.

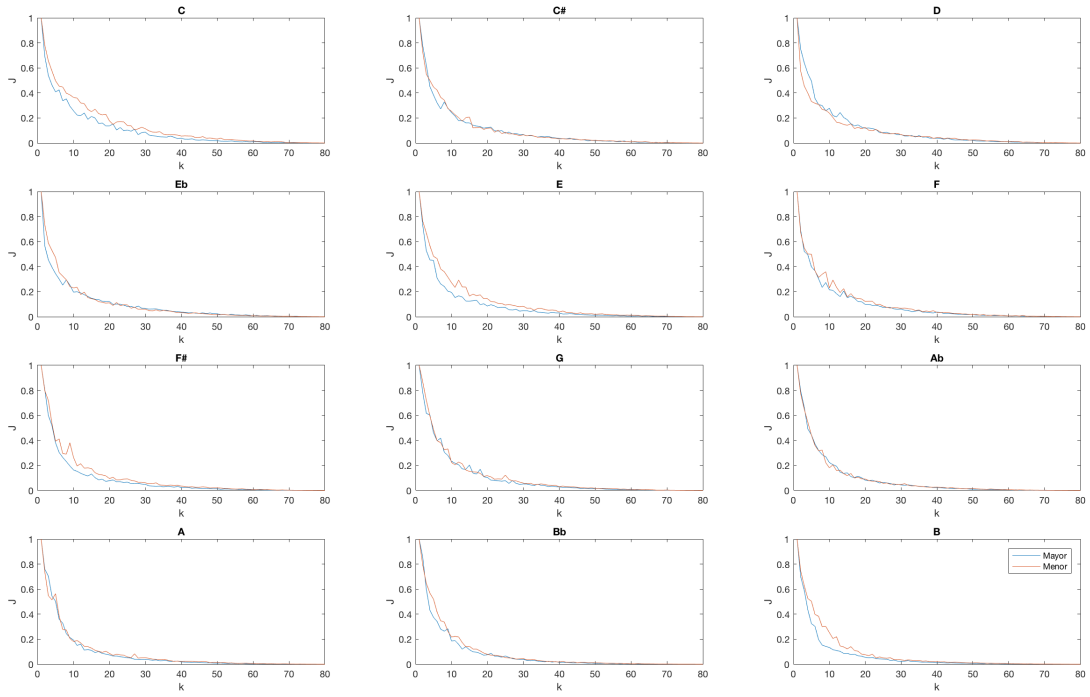

Figura 30: Función J para k = 1...80. Fuente: Elaboración propia

Teniendo en cuenta que deberíamos escoger el mínimo valor de k para el cual la función J tiende al eje de abscisas, se ha estipulado que el valor de k que mejor se adapta a nuestro sistema es 9.

De este modo, se ha implementado el algoritmo k-means para un total de 9 agrupaciones compuestas por los cromagramas de cada uno de los 80 instrumentos que conforman la base de datos de entrenamiento. Debido a las dificultades técnicas que supone realizar el experimento con plantillas completas de 12x12x3, se ha generalizado el problema a los acordes de C mayor y C menor, obteniendo así 18 cromagramas (9 agrupaciones mayores, y 9 menores), que representan los centroides de las agrupaciones generadas. Posteriormente, y del mismo modo a cómo se hizo con las primeras plantillas, se rotarán los acordes de C mayor y C menor para obtener las plantillas completas con todos los acordes del vocabulario armónico.

4. Evaluación

Esta sección tratará todos los temas relacionados con la evaluación del sistema desarrollado, explicando las métricas que definirán los parámetros que expresarán la precisión obtenida a la hora de estimar acordes, así como representando los resultados obtenidos de la fase de testeo.

Puesto que el sistema desarrollado trata de etiquetar cada instante temporal (con la resolución ofrecida por el cromagrama) del fichero analizado, consideramos que la tasa de acierto será una métrica que reflejará fielmente su eficiencia.

En el caso que nos atañe, y puesto que en el presente trabajo no se pretende segmentar armónicamente una pieza musical, si no más bien detectar acordes, se ha establecido un sistema de medición de la precisión en el que la evaluación solo se realice cuando exista información de acorde.

Así pues, se dispondrá de un fichero Ground-Truth (GT) que contendrá la información precisa y verdadera con respecto a los acordes que componen los ficheros de test. Esto quiere decir que para cada fichero de test existirá un fichero GT gemelo que dispondrá de la información real de acorde en cada instante temporal. Siendo más precisos, el fichero GT dispone de 3 columnas en las que se expresa el tiempo de inicio del acorde (Onset), el tiempo final del acorde (Offset), y la etiqueta de acorde (como en el caso del fichero estimado, siguiendo el formato propuesto por Harte).

De este modo, y gracias a que el script desarrollado en Csound para la síntesis de los ficheros con progresiones ascendentes de acordes mayores y menores genera al mismo tiempo el fichero de GT, podremos generar el corpus del trabajo sin ninguna dificultad.

La detección de Onsets y Offsets desde el script en Csound se consigue mediante la obtención de la envolvente de la señal de audio sintetizada. Posteriormente se establece un nivel umbral que delimite la intensidad necesaria para que la señal se pueda interpretar como acorde, haciendo que el programa escriba los instantes temporales donde la envolvente cruza el nivel umbral, y estableciendo así los instantes de Onset y Offset.

4.1. Métricas empleadas

Disponiendo entonces del fichero GT, se procederá a la comparación del fichero generado de la estimación con éste. De este modo, se compararán las etiquetas de acorde de cada ins-

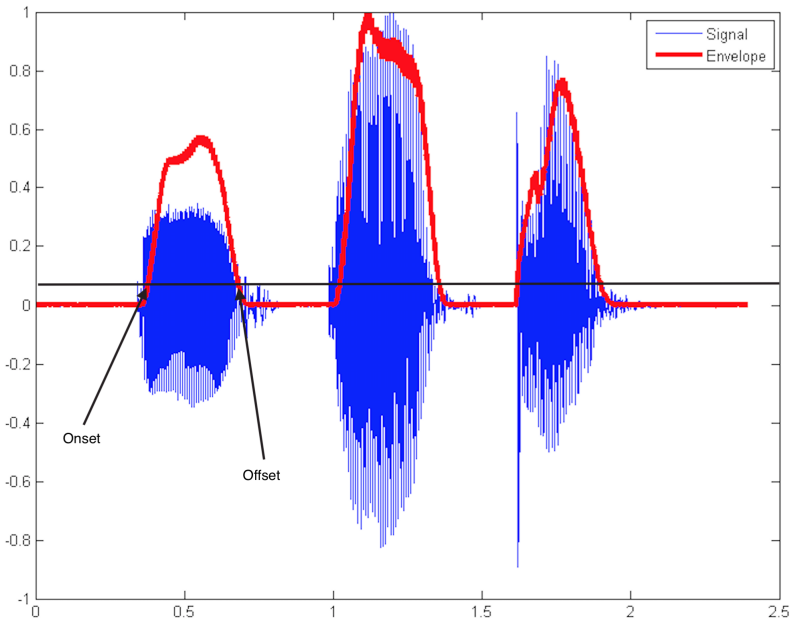

Figura 31: Detección de Onsets y Offsets Fuente: `http://archive.cnx.org/contents/c944c56d-860b-4ce9-bec3-c9b15bc3a168@2/envelope-detection-in-speech-signals`

tante temporal del fichero estimado con las etiquetas de acordes del fichero GT que coincidan con el periodo temporal definido por los Onsets y Offsets (es decir, donde hay información de acorde). Realizando la evaluación del sistema exclusivamente en los instantes en los que existe información de acorde, conseguimos evitar tasas de acierto en las que el error es debido prácticamente al rápido decaimiento típico de algunos instrumentos (como pueda ser la marimba), permitiéndonos de este modo tener una visión más clara sobre la eficiencia de la categorización de acordes.

Las normas de puntuación que se han seguido para calcular la tasa de acierto del sistema son:

- Puntuación total (1) para el acierto absoluto del acorde.
- Puntuación media (1/2) para el acierto de la raíz del acorde.

La puntuación obtenida se dividirá entre el número de frames del fichero analizado, lo que consecuentemente resultará en una tasa de aciertos comprendida entre 0 y 1. Ésta se multiplicará posteriormente por 100 para expresarse de forma porcentual, y definir la precisión

obtenida para cada fichero de test.

Además, teniendo en cuenta que el sistema desarrollado está completamente automatizado, realizando la estimación de acordes para todos los ficheros de la base de datos de test, será necesario definir un parámetro que exprese la precisión general del sistema. Por lo que se ha decido calcular también la tasa de acierto general como la media ponderada a la duración de cada fichero. Por consiguiente, por cada ejecución se calculará la tasa de acierto porcentual obtenida para cada fichero analizado, así como la tasa de acierto general del sistema ponderada a la duración de cada fichero (teniendo de este modo más peso en la tasa de acierto general los ficheros analizados con mayor duración).

4.2. Resultados

Como se explicó en la sección 3.1, se ha pretendido diseñar un corpus extenso que nos permita obtener resultados representativos del funcionamiento del sistema. Así, se ha probado el sistema utilizando ficheros de audio sintetizado de dos instrumentos de cada familia para dos motores de síntesis distintos, en los que existe una progresión ascendente de acordes mayores y menores, así como ficheros de audio de grabaciones de instrumentos reales en los que podremos encontrar acordes aislados de piano, guitarra, acordeón y violín.

La nomenclatura seguida para la denominación de los ficheros sigue el siguiente formato:

- NN: Número correspondiente a la asignación de instrumento según la tabla General Midi.
- CC: Etiqueta de acorde (solo presente en los ficheros de grabaciones de instrumentos reales).
- xx: Naturaleza del fichero de audio.
    - Tim = Fichero de audio sintetizado con TimGM6mb.
    - FluidR3 = Fichero de audio sintetizado con FluidR3GM2.
    - Rec = Fichero de audio de grabación de instrumentos reales.

Ejemplo: NN_CC_xx.wav $\longrightarrow$ *01_Bb_Rec.wav*

	Tasa de acierto (%)			
Nombre del fichero	Plantillas binarias	Plantillas con 11 armónicos	Plantillas GM	Plantillas clustering
00_Tim.wav	95.89	95.16	97.80	99.12
04_Tim.wav	94.76	94.19	97.31	32.29
12_Tim.wav	87.08	85.96	83.99	85.39
13_Tim.wav	32.64	31.60	30.56	30.56
16_Tim.wav	93.67	89.11	93.16	93.67
21_Tim.wav	88.94	89.86	85.60	70.97
25_Tim.wav	98.51	84.82	98.51	98.81
27_Tim.wav	83.75	51.24	90.11	78.80
33_Tim.wav	82.82	97.55	90.49	99.39
37_Tim.wav	91.18	62.53	91.18	93.53
40_Tim.wav	89.66	84.48	90.02	91.01
42_Tim.wav	84.91	84.55	86.91	91.16
48_Tim.wav	81.03	71.95	80.69	81.15
52_Tim.wav	70.87	75.23	67.66	57.22
56_Tim.wav	78.18	83.69	85.13	93.29
58_Tim.wav	93.36	90.82	93.36	86.23
68_Tim.wav	96.05	80.49	96.30	96.17
70_Tim.wav	84.74	85.09	88.62	83.80
73_Tim.wav	85.70	78.36	87.53	88.26
76_Tim.wav	87.19	93.09	86.31	84.67
General	**85.13**	**80.59**	**86.15**	**81.86**

Tabla 2: Resultados para ficheros sintetizados con el motor TimGM6mb.

	Tasa de acierto (%)			
Nombre del fichero	Plantillas binarias	Plantillas con 11 armónicos	Plantillas GM	Plantillas clustering
00_FluidR3.wav	98.53	84.16	98.97	100
04_FluidR3.wav	97.45	97.31	97.73	22.80
12_FluidR3.wav	62.36	60.11	59.83	60.67
13_FluidR3.wav	49.31	51.74	45.14	47.92
16_FluidR3.wav	95.44	91.77	93.42	90.89
21_FluidR3.wav	75.69	72.00	79.26	75.58
25_FluidR3.wav	77.98	63.10	77.83	98.66
27_FluidR3.wav	94.35	93.11	87.10	72.44
33_FluidR3.wav	100	96.32	97.24	98.16
37_FluidR3.wav	91.46	92.56	84.85	68.87
40_FluidR3.wav	77.96	75.86	78.82	92.73
42_FluidR3.wav	69.10	85.14	56.72	48.00
48_FluidR3.wav	84.60	75.98	86.09	87.13
52_FluidR3.wav	72.25	80.85	65.02	48.05
56_FluidR3.wav	92.69	86.21	93.17	93.29
58_FluidR3.wav	91.43	93.96	92.51	75.60
68_FluidR3.wav	94.07	71.11	94.69	94.69
70_FluidR3.wav	52.93	84.27	49.30	32.51
73_FluidR3.wav	81.91	70.90	77.51	83.62
76_FluidR3.wav	71.86	81.91	77.51	77.64
General	**81.60**	**80.48**	**79.68**	**73.50**

Tabla 3: Resultados para ficheros sintetizados con el motor FluidR3GM2.

	Tasa de acierto (%)			
Nombre del fichero	Plantillas binarias	Plantillas con 11 armónicos	Plantillas GM	Plantillas clustering
00_A_Rec.wav	95.45	72.73	100	100
00_Bm_Rec.wav	100	100	93.33	100
00_D_Rec.wav	100	90	100	100
00_E_Rec.wav	100	92.86	100	100
00_F_Rec.wav	80	86.67	86.67	93.33
22_A_Rec.wav	100	100	100	100
22_Bm_Rec.wav	100	100	100	0
22_D_Rec.wav	100	19.05	100	100
22_E_Rec.wav	100	73.17	100	100
22_F_Rec.wav	100	100	100	100
25_Am_Rec.wav	77.27	81.82	4.55	9.09
25_C_Rec.wav	42.86	19.05	100	57.14
25_Dm_Rec.wav	22.73	9.09	38.64	93.18
25_Em_Rec.wav	20	95.00	0	0
25_G_Rec.wav	91.30	91.30	82.61	100
40_Am_Rec.wav	100	100	85.71	71.43
40_C_Rec.wav	88.24	38.24	58.82	88.24
40_Dm_Rec.wav	87.50	100	93.75	87.50
40_Em_Rec.wav	100	100	0	0
40_G_Rec.wav	58.33	47.92	50	81.25
General	**82.63**	**74.57**	**76.71**	**76.57**

Tabla 4: Resultados para ficheros de instrumentos reales grabados.

5. Conclusiones

Después de realizar las correspondientes pruebas de test, obteniendo así los resultados que reflejan la precisión del sistema propuesto, se procederá a su análisis, así como la valoración de las decisiones tomadas durante la fase de diseño. Por último, y en parte como consecuencia de las conclusiones extraídas, se esbozarán las líneas futuras del presente trabajo.

5.1. Discusión de resultados

Observando los resultados obtenidos de la fase experimental del presente trabajo, se hace un poco complicado extraer conclusiones firmes y evidentes.

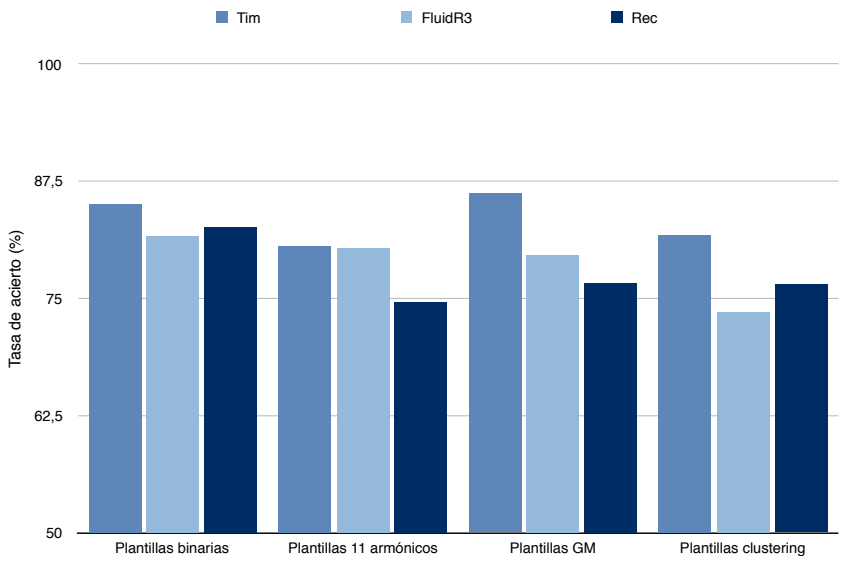

Tabla 5: Comparativa de tasas de acierto general.

En primer lugar, cabe resaltar la efectividad de las plantillas binarias pertenecientes al diseño más primitivo del sistema. Los resultados obtenidos empleando dichas plantillas se mantienen entorno al 82 % - 85 % para los valores de precisión general del sistema. Probable-

mente, debido a la simplicidad de su definición, aunque estas plantillas no se correspondan prácticamente con lo que cabría esperar del comportamiento espectral (y consecuentemente armónico) de una señal procedente de un instrumento musical, definen claramente las prominencias de cada uno de los acordes.

Del análisis de los resultados obtenidos para las plantillas elaboradas con la superposición de la serie de 11 armónicos se obtienen conclusiones similares a las de las plantillas binarias. Siendo más complejas que las plantillas binarias, y teniendo en principio una correspondencia superior a la realidad, los resultados generales obtenidos son inferiores para todos los casos. Quizás sí puedan ofrecer un resultado competitivo con las demás para los casos de los instrumentos sintetizados con ambos soundfonts (motores de síntesis), y probablemente esto sea debido a que la síntesis realizada para los ficheros de test no es de alto nivel, y por tanto no deben contemplar demasiados factores no lineales que puedan afectar al comportamiento armónico de una señal de audio producida por un instrumento. Quizás precisamente por estos factores no lineales (como puedan ser frecuencias de resonancia características del instrumento, o la dependencia a la intensidad de determinadas frecuencias presentes en el espectro), se puede observar claramente como este sistema no ofrece una precisión especialmente positiva con instrumentos reales grabados.

Entrando ya en el terreno de la clasificación instrumental, las plantillas obtenidas de la clasificación GM han conseguido una tasa de acierto general considerablemente buena con los ficheros sintetizados con TimGM6mb, estableciéndose como la cota más alta obtenida en el experimento. Esto tiene sentido ya que las plantillas se han entrenado precisamente con ficheros sintetizados con dicho soundfont, por lo que éstas parecen representar fielmente las características armónicas de los cromagramas de cada familia instrumental. Además, se trata de la plantilla no binaria que mejor precisión parece alcanzar con ficheros de grabación de instrumentos reales, por lo que se puede extraer que la clasificación por familias instrumentales ha sido una decisión acertada.

Para finalizar con el análisis de los resultados generales obtenidos, particularizados a las plantillas definidas, a pesar de ser las más costosas de obtener, y estar fundamentadas en la hipótesis de que las agrupaciones por familias de instrumentos propuesta por General Midi no son demasiado compactas (incluso corroborándolo con el estudio estadístico realizado), las plantillas obtenidas de la agrupación alternativa mediante clustering no demuestran ser demasiado efectivas. Siendo precisos, y para sorpresa del autor del presente trabajo, las

plantillas obtenidas de la agrupación alternativa a las familias GM muestran unos valores de precisión bastante bajos para los ficheros de audio sintetizados.

A pesar de ello, y analizando los resultados obtenidos para cada fichero individualmente, este tipo de plantillas ha generado en varias ocasiones tasas de acierto del 100 %, lo que hace pensar que el error quizás podría residir en la elección del valor de k.

En pos de extraer unas conclusiones generales de los resultados obtenidos, se podría decir que la clasificación instrumental ha favorecido principalmente a los ficheros pertenecientes a la base de datos de ficheros sintetizados con el mismo soundfont que los ficheros de entrenamiento, ofreciendo resultados mejorables en el resto de casos.

5.2. Líneas futuras de trabajo

Partiendo de las conclusiones extraídas de la fase experimental del presente trabajo, casi inevitablemente se generan determinadas vías de trabajo futuro para la mejora del sistema propuesto.

En primer lugar, y viendo los resultados obtenidos de la clasificación instrumental, en los que se puede apreciar claramente como la tasa de acierto mejora considerablemente con los ficheros de su misma naturaleza, cabe imaginar que este comportamiento podría extrapolarse al caso de instrumentos grabados. De este modo, y como principal objetivo de mejora a corto plazo, se plantea el desarrollo de plantillas de familias de instrumentos entrenadas con ficheros de audio procedentes de instrumentos reales grabados. A pesar de lo costoso que resultaría generar un repositorio de ficheros de audio lo suficientemente amplio como para entrenar las plantillas, se podría generalizar el problema a no más de 3 o 4 familias de instrumentos que comprendiesen los instrumentos más cotidianos en la cultura occidental (piano, guitarra, conjuntos de cuerda...). Además de ello, implementar otro algoritmo de clustering, definiendo a su vez un vector de características basado en parámetros tímbricos en lugar de armónicos, entra también dentro de los planes de desarrollo futuro a corto plazo.

Debido a que el objetivo principal de diseño de nuestro sistema está basado en la clasificación instrumental, no se ha incidido demasiado en disponer de un vocabulario armónico amplio, generalizando así el problema a dos tipos de acorde. Así pues, la ampliación del vocabulario armónico empleado, será también una vía de desarrollo necesaria para la consumación del sistema. Contemplar acordes de cuatriada, aumentados, disminuidos, incluso acordes invertidos, podrían tener cabida en las vías próximas de desarrollo, más si se com-

plementa la información obtenida del cromagrama con la información que nos ofrecería el cromagrama grave (Bass Chroma).

Por otro lado, y con vistas de enfocar el sistema desarrollado de una forma más práctica a nivel de usuario, se podría, mediante la detección de Onsets y Offsets en el fichero de audio estimado, realizar una segmentación armónica de una pieza musical. Quizás de mayor complejidad, y un poco más alejado del alcance del presente trabajo, podría llegar a desarrollarse el sistema para su uso en tiempo real, permitiendo de este modo la estimación automática e instantánea de acordes de una interpretación en directo.

6. Bibliografía

AIKIN, J. (2012). *Csound Power.*

BELLO, J.P., DAUDET L., ABDALLAH, S., DUXBURY, C, DAVIES, M., AND SANDLER M.B. (2005). *A Tutorial on Onset Detection in Music Signals.*

BOULANGUER, R. (2010). *The Csound Book.*

CHO, T, WEISS, R.J., AND BELLO, J.P. (2010). *Exploring Common Variations in State of the Art Chord Recognition Systems.*

CHUAN, C. AND CHEW, E. (2008). *Audio Onset Detection Using Machine Learning Techniques: The Effect and Applicability of Key and Tempo Information.*

DRIEDGER, J. AND PR ATZLICH, T. (2014). *Harmonic Percussive Source Separation (Lab Course).*

DOWNIE, J. (2008). *The Music Information Retrieval Evaluation Exchange (2005-2007)): A window into music information retrieval research.*

DOWNIE, J., EHMANN, F.A., BAY, M. AND JONES, M.C. (2010). *The Music Information Retrieval Evaluation eXchange: Some Observations and Insights.*

EVEREST, F.A. AND POHLMANN, K.C. (2009). *Master Handbook of Acoustics.*

FUJISHIMA, T. (1999). *Realtime Chord Recognition of Musical Sound: A System Using Comon Lisp Music.*

HARTE, C, SANDLER, M, ABDALLAH, S. AND GÓMEZ, E. (2005). *Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotations.*

HOWARD, D.M. AND ANGUS, J.A.S. (2009). *Acoustics and Psychoacoustics.*

IÑESTA QUEREDA, J.M. (2013). *Apuntes de la asignatura Síntesis Digital del Sonido.*

KLAPURI, A (2006). *Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes.*

MCVICAR, M., SANTOS-RODRIGUEZ, R., NI, Y., AND DE BIE, T. (2014). *Automatic Chord Estimation from Audio: A Review of the State of the Art.*

ROADS, C. (1996). *The Computer Music Tutorial.*

SEGURA SOGORB, M, 2015. *Estimación automática de acordes para uso en transcripción musical a partir de audio.*

DLSI, UNIVERSIDAD DE ALICANTE, 2013. *Diapositivas empleadas en sesión explicativa sobre Clustering.*

VALERO, J.J. AND IÑESTA, J.M., 2015. *Interactive Onset Detection in Audio Recordings.*

A. Anexo: Scripts desarrollados

Función main.m

```
function [] = main (fileType,pattSelector)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            FUNCION MAIN
%
% Nombre:   main
%
% Input:    fileType            Variable con el tipo de fichero de test.
%           ("Tim", "FluidR3": Audios sintetizados, "Rec": Audios grabados)
%           pattSelector        Variable con el tipo de plantilla.
%           (1:Binaria, 2:11 armonicos, 3:Familia GM, 4:Clustering)
%
% Descripcion:  Funcion encargada de automatizar el proceso, e implementar
%               la estimacion de acordes (mediante la llamada a chodsEstim)
%               de todos los ficheros contenidos en la carpeta del dataset,
%               asi como la obtencion de la tasa de aciertos ponderada de
%               toda la base de datos contemplada.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc

% Directorio con todos los ficheros WAVE del dataset de test que
% pertenezcan al tipo seleccionado.
listing =       dir (['dataset/test/*',fileType,'.wav']);

% Inicializacion de la duracion total y de la puntuacion media.
totalduration = 0;
score =         0;

% Generacion del fichero de resultados.
fid = fopen(['precision_',fileType,'.txt'],'w');

% Para cada uno de los ficheros WAVE de test.
for file = 1:length(listing)

    filename =  strrep (listing(file).name,'.wav','');

    % Si se eligen plantillas obtenidas por clustering.
    if pattSelector == 4

        % Obtencion de precision y duracion del fichero mediante el uso de
        % chordsEstim para la asignacion de plantillas mayores y menores.
```

```
        [evaluationMaj,duration] = chordsEstim(filename,pattSelector,'maj');
        [evaluationMin,duration] = chordsEstim(filename,pattSelector,'min');

        % Obtencion de la tasa de acierto mayor de usar ambas plantillas.
        evaluation =    max (evaluationMaj,evaluationMin);

    % Si se elige cualquier otro tipo de plantilla se ejecuta normalmente.
    else
        % Obtencion de precision y duracion del fichero.
        [evaluation,duration] = chordsEstim (filename,pattSelector,'null');
    end

    showAccuracy = ['Accuracy (',filename,'): ',ceil(num2str(evaluation)),'%'];
    disp (showAccuracy);

    fprintf(fid,'Accuracy (%s):    %4.2f %s\n',filename,evaluation,'%');

    % Calculo del valor de precision ponderado a la duracion del fichero.
    weightEval =  evaluation*duration;
    score = score+weightEval;                   % Acumulacion de precision.
    totalduration = totalduration+duration;     % Acumulacion de duracion.
end

% Calculo de la puntuacion media y escritura en el fichero.
score = score/totalduration;
fprintf(fid,'\nScore: %2.2f%s',score,'%');

fclose(fid);

% Visualizacion de la puntuacion media.
showScore = sprintf('\nScore: %2.2f%s',score,'%');
disp (showScore);

% Ordenacion del fichero de resultados.
movefile(['precision_',fileType,'.txt'],'results/precision/');

end
```

Función chordsEstim.m

```
function [evaluation,duration] = chordsEstim (filename,pattSelector,type)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   FUNCION DE ESTIMACION DE ACORDES
%
% Nombre:   chordsEstim
%
% Input:    filename            Variable con el nombre del fichero.
%           pattSelector        Variable con el tipo de plantilla.
%           type                Variable con la cadena correspondiente al
%                               tipo de acorde para la asginacion de
%                               plantillas cuando P = 4.
%
% Output:   evaluation          Variable con el valor de la tasa de acierto
%                               obtenida para el fichero procesado.
%           duration            Variable con el valor de la duracion del
%                               fichero procesado.
%
% Descripcion:  Funcion encargada de realizar todos los procesos
%               relacionados con la estimacion de acordes, asi como su
%               evaluacion, del fichero de entrada filename.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Localizacion y lectura de todos los ficheros WAVE de test.
filepath = ['dataset/test/' filename];
audioname =     strcat (filepath,'.wav');
audio =         audioread (audioname);

% Separacion armonica-percusiva.
[audioH,audioP] =   hpSep (audio);
audioHname =    strcat (filename,'H.wav');
audioPname =    strcat (filename,'P.wav');
audiowrite  (audioHname,audioH,44100);
audiowrite  (audioPname,audioP,44100);

% Extraccion y carga del cromagrama.
system (['./sonic-annotator -t config.txt ' audioHname ' -w csv --csv-force 2> temp.txt'...
    ]);
chroma =        load ([filename,'H_vamp_nnls-chroma_nnls-chroma_chroma.csv']);

% Generacion de plantillas.
pattern =       patternGen (pattSelector,filename,type);

% Obtencion del fichero resultado en CSV.
tags =          patternMatch (chroma,pattern);
```

```
estimname =     estimationGen (filename,chroma,tags);

% Calculo de la precision y obtencion de la duracion del fichero.
[evaluation,duration] =    GTMetrics (filename);

% Ordenacion de los ficheros generados.
movefile(audioHname,'dataset/test/harmonic/');
movefile(audioPname,'dataset/test/percussive/');
movefile([filename,'H_vamp_nnls-chroma_nnls-chroma_chroma.csv'],'chromas/');
movefile(estimname,'results/estimation/');

end
```

Función patternGen.m

```
function [pattern] = patternGen (pattSelector,filename,type)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   FUNCION DE GENERACION DE PLANTILLAS
%
% Nombre:   patternGen
%
% Input:    filename            Variable con el nombre del fichero.
%           pattSelector        Variable de seleccion de plantillas.
%           type                Variable con la cadena correspondiente al
%                               tipo de acorde para la asginacion de
%                               plantillas cuando P = 4.
%
% Output:   pattern             Matriz de 3x12x12 que contiene las
%                               plantillas correspondientes.
%
%
% Descripcion:  Funcion encargada de generar las plantillas
%               correspondientes en funcion del valor de pattSelector:
%
%               - pattSelector = 1 -> Plantilla binaria.
%               - pattSelector = 2 -> Plantilla con 11 armonicos.
%               - pattSelector = 3 -> Plantilla de familia General Midi.
%               - pattSelector = 4 -> Plantilla obtenida del clustering.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch pattSelector

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANTILLA BINARIA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 1

        % Triada mayor
        pattern(1,:,1) = [1 0 0 0 1 0 0 1 0 0 0 0];

        % Rotacion
        for pos = 2:12
            pattern(1,2:12,pos) = pattern(1,1:11,pos-1);
            pattern(1,1,pos) =   pattern(1,12,pos-1);
        end

        % Triada menor
        pattern(2,:,1) = [1 0 0 1 0 0 0 1 0 0 0 0];
```

```
        % Rotacion
        for pos = 2:12
            pattern(2,2:12,pos) = pattern(2,1:11,pos-1);
            pattern(2,1,pos) =   pattern(2,12,pos-1);
        end

        % Silencio
        pattern(3,:,1) = [0 0 0 0 0 0 0 0 0 0 0 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANTILLA CON 11 ARMONICOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 2

        % Triada mayor
        pattern(1,:,1) = [0.76 0.04 0.34 0.00 0.88 0.06 0.08 1.00 0.12 0.05 0.10 0.36];

        % Rotacion
        for pos = 2:12
            pattern(1,2:12,pos) = pattern(1,1:11,pos-1);
            pattern(1,1,pos) =   pattern(1,12,pos-1);
        end

        % Triada menor
        pattern(2,:,1) = [0.68 0.08 0.25 0.68 0.11 0.09 0.00 1.00 0.00 0.07 0.26 0.11];

        % Rotacion
        for pos = 2:12
            pattern(2,2:12,pos) = pattern(2,1:11,pos-1);
            pattern(2,1,pos) =   pattern(2,12,pos-1);
        end

        % Silencio
        pattern(3,:,1) = [0 0 0 0 0 0 0 0 0 0 0 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANTILLA DE FAMILIA GENERAL MIDI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 3

        % Obtencion del numero de instrumento del fichero.
        C =                 strsplit (filename,'_');
        instrumentNum =     str2double (C(1));

        if (instrumentNum ≥ 0) && (instrumentNum < 8)
            % Lectura y remodelado de la plantilla de pianos.
            pattern =     csvread ('patterns/family/piano.csv');
        end
```

```
if (instrumentNum ≥ 8) && (instrumentNum < 15)
    % Lectura y remodelado de la plantilla de percusion cromatica.
    pattern =     csvread ('patterns/family/chromperc.csv');
end

if (instrumentNum ≥ 16) && (instrumentNum < 23)
    % Lectura y remodelado de la plantilla de organos.
    pattern =     csvread ('patterns/family/organ.csv');
end

if (instrumentNum ≥ 24) && (instrumentNum < 31)
    % Lectura y remodelado de la plantilla de guitarras.
    pattern =     csvread ('patterns/family/guitar.csv');
    pattern =     reshape (pattern,[3 12 12]);
end

if (instrumentNum ≥ 32) && (instrumentNum < 39)
    % Lectura y remodelado de la plantilla de bajos.
    pattern =     csvread ('patterns/family/bass.csv');
end

if (instrumentNum ≥ 40) && (instrumentNum < 47)
    % Lectura y remodelado de la plantilla de cuerdas.
    pattern =     csvread ('patterns/family/strings.csv');
end

if (instrumentNum ≥ 48) && (instrumentNum < 55)
    % Lectura y remodelado de la plantilla de ensemble.
    pattern =     csvread ('patterns/family/ensemble.csv');
end

if (instrumentNum ≥ 56) && (instrumentNum < 63)
    % Lectura y remodelado de la plantilla de viento-metal.
    pattern =     csvread ('patterns/family/brass.csv');
end

if (instrumentNum ≥ 64) && (instrumentNum < 71)
    % Lectura y remodelado de la plantilla de solistas.
    pattern =     csvread ('patterns/family/lead.csv');
end

if (instrumentNum ≥ 72) && (instrumentNum < 79)
    % Lectura y remodelado de la plantilla de viento-madera.
    pattern =     csvread ('patterns/family/pipe.csv');
end

% Reordenacion de la plantilla.
pattern =     reshape (pattern,[3 12 12]);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLANTILLA OBTENIDA MEDIANTE CLUSTERING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    case 4

        % Obtencion del numero de instrumento del fichero.
        C =                 strsplit (filename,'_');
        instrumentNum =     str2double(C(1));

        % Asignacion de plantillas mayores.
        if  strcmp (type,'maj')

            if (instrumentNum == 2)||(instrumentNum == 6)||(instrumentNum == 7)||(...
                instrumentNum == 10)||(instrumentNum == 11)||(instrumentNum == 15)||(...
                instrumentNum == 20)||(instrumentNum == 25)||(instrumentNum == 29)||(...
                instrumentNum == 30)||(instrumentNum == 40)||(instrumentNum == 41)||(...
                instrumentNum == 42)||(instrumentNum == 43)||(instrumentNum == 44)||(...
                instrumentNum == 45)||(instrumentNum == 46)||(instrumentNum == 48)||(...
                instrumentNum == 49)||(instrumentNum == 53)||(instrumentNum == 59)||(...
                instrumentNum == 60)||(instrumentNum == 61)||(instrumentNum == 62)||(...
                instrumentNum == 64)||(instrumentNum == 65)||(instrumentNum == 66)||(...
                instrumentNum == 68)||(instrumentNum == 70)||(instrumentNum == 73)
                % Lectura de la plantilla del cluster 1.
                pattern =     csvread ('patterns/clustering/1.csv');
            end

            if (instrumentNum == 5)||(instrumentNum == 12)||(instrumentNum == 21)||(...
                instrumentNum == 23)||(instrumentNum == 24)||(instrumentNum == 28)||(...
                instrumentNum == 50)||(instrumentNum == 51)||(instrumentNum == 52)||(...
                instrumentNum == 54)||(instrumentNum == 58)||(instrumentNum == 63)||(...
                instrumentNum == 71)||(instrumentNum == 77)
                % Lectura de la plantilla del cluster 2.
                pattern =     csvread ('patterns/clustering/2.csv');
            end

            if (instrumentNum == 16)||(instrumentNum == 31)
                % Lectura de la plantilla del cluster 3.
                pattern =     csvread ('patterns/clustering/3.csv');
            end

            if (instrumentNum == 8)||(instrumentNum == 13)||(instrumentNum == 19)||(...
                instrumentNum == 32)||(instrumentNum == 67)
                % Lectura de la plantilla del cluster 4.
                pattern =     csvread ('patterns/clustering/4.csv');
            end

            if (instrumentNum == 14)||(instrumentNum == 69)
                % Lectura de la plantilla del cluster 5.
                pattern =     csvread ('patterns/clustering/5.csv');
```

```
    end

    if (instrumentNum == 0)||(instrumentNum == 1)||(instrumentNum == 3)||(...
        instrumentNum == 4)||(instrumentNum == 22)||(instrumentNum == 26)||(...
        instrumentNum == 56)||(instrumentNum == 57)
        % Lectura de la plantilla del cluster 6.
        pattern =     csvread ('patterns/clustering/6.csv');
    end

    if (instrumentNum == 33)||(instrumentNum == 34)||(instrumentNum == 35)||(...
        instrumentNum == 36)||(instrumentNum == 38)||(instrumentNum == 39)||(...
        instrumentNum == 47)||(instrumentNum == 55)
        % Lectura de la plantilla del cluster 7.
        pattern =     csvread ('patterns/clustering/7.csv');
    end

    if (instrumentNum == 9)||(instrumentNum == 78)
        % Lectura de la plantilla del cluster 8.
        pattern =     csvread ('patterns/clustering/8.csv');
    end

    if (instrumentNum == 18)||(instrumentNum == 72)||(instrumentNum == 74)||(...
        instrumentNum == 75)||(instrumentNum == 76)||(instrumentNum == 79)
        % Lectura de la plantilla del cluster 9.
        pattern =     csvread ('patterns/clustering/9.csv');
    end
end

% Asignacion de plantillas menores
if strcmp (type,'min')

    if (instrumentNum == 6)||(instrumentNum == 16)||(instrumentNum == 17)||(...
        instrumentNum == 19)||(instrumentNum == 20)||(instrumentNum == 24)||(...
        instrumentNum == 28)||(instrumentNum == 30)||(instrumentNum == 31)||(...
        instrumentNum == 44)||(instrumentNum == 46)||(instrumentNum == 58)||(...
        instrumentNum == 60)||(instrumentNum == 64)||(instrumentNum == 65)||(...
        instrumentNum == 66)||(instrumentNum == 68)
        % Lectura de la plantilla del cluster 1.
        pattern =     csvread ('patterns/clustering/1.csv');
    end

    if (instrumentNum == 2)||(instrumentNum == 4)||(instrumentNum == 10)||(...
        instrumentNum == 15)||(instrumentNum == 22)||(instrumentNum == 25)||(...
        instrumentNum == 26)||(instrumentNum == 27)||(instrumentNum == 40)||(...
        instrumentNum == 41)||(instrumentNum == 42)||(instrumentNum == 43)||(...
        instrumentNum == 48)||(instrumentNum == 51)||(instrumentNum == 53)||(...
        instrumentNum == 59)||(instrumentNum == 61)||(instrumentNum == 62)||(...
        instrumentNum == 67)||(instrumentNum == 70)||(instrumentNum == 72)||(...
        instrumentNum == 73)||(instrumentNum == 74)
```

```
        % Lectura de la plantilla del cluster 2.
        pattern =     csvread ('patterns/clustering/2.csv');
    end

    if (instrumentNum == 14)||(instrumentNum == 21)||(instrumentNum == 23)||(...
        instrumentNum == 69)
        % Lectura de la plantilla del cluster 3.
        pattern =     csvread ('patterns/clustering/3.csv');
    end

    if (instrumentNum == 5)||(instrumentNum == 8)||(instrumentNum == 32)||(...
        instrumentNum == 33)||(instrumentNum == 35)||(instrumentNum == 37)||(...
        instrumentNum == 38)||(instrumentNum == 39)||(instrumentNum == 49)||(...
        instrumentNum == 50)||(instrumentNum == 52)||(instrumentNum == 54)||(...
        instrumentNum == 63)||(instrumentNum == 75)||(instrumentNum == 76)
        % Lectura de la plantilla del cluster 4.
        pattern =     csvread ('patterns/clustering/4.csv');
    end

    if (instrumentNum == 11)||(instrumentNum == 12)||(instrumentNum == 18)||(...
        instrumentNum == 36)||(instrumentNum == 45)||(instrumentNum == 55)
        % Lectura de la plantilla del cluster 5.
        pattern =     csvread ('patterns/clustering/5.csv');
    end

    if (instrumentNum == 9)||(instrumentNum == 13)||(instrumentNum == 77)
        % Lectura de la plantilla del cluster 6.
        pattern =     csvread ('patterns/clustering/6.csv');
    end

    if (instrumentNum == 78)
        % Lectura de la plantilla del cluster 7.
        pattern =     csvread ('patterns/clustering/7.csv');
    end

    if (instrumentNum == 0)||(instrumentNum == 1)||(instrumentNum == 3)||(...
        instrumentNum == 57)||(instrumentNum == 71)||(instrumentNum == 79)
        % Lectura de la plantilla del cluster 8.
        pattern =     csvread ('patterns/clustering/8.csv');
    end

    if (instrumentNum == 7)||(instrumentNum == 29)||(instrumentNum == 34)||(...
        instrumentNum == 47)||(instrumentNum == 56)
        % Lectura de la plantilla del cluster 9.
        pattern =     csvread ('patterns/clustering/9.csv');
    end

end
```

```
        % Reordenacion de la plantilla.
        pattern =     reshape (pattern,[3 12 12]);

end

end
```

Función patternMatch.m

```
function [tags] = patternMatch (chroma,pattern)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 FUNCION DE CORRELACION DE PLANTILLAS
%
% Nombre:   patternMatch
%
% Input:    chroma              Matriz que contiene el vector de cromagrama
%                               (12x1) obtenido para cada frame.
%           pattern             Matriz de 3x12x12 que contiene las
%                               plantillas correspondientes.
%
% Output:   tags                Vector de cadena de caracteres que contiene
%                               la estimacion del acorde en cada frame.
%
% Descripcion:  Funcion encargada de estimar la correlacion entre el
%               cromagrama obtenido y las plantillas seleccionadas mediante
%               el calculo de su distancia euclidea.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Etiquetas de raices
rootTags =      {'A','Bb','B','C','C#','D','Eb','E','F','F#','G','Ab'};
% Numero total de muestras temporales de croma.
[T,¬] =         size(chroma);


% Para cada frame.
for frame = 1:T

        % Vector de cromagrama del frame.
        c = chroma(frame,2:13);

        % Calculo de 12 distancias por cada modelo de acorde.
        for bin = 1:12

            % Triada mayor
            distMaj(frame,bin) = norm (c - pattern(1,:,bin));
            % Triada menor
        distMin(frame,bin) = norm (c - pattern(2,:,bin));
            % Silencio
            distSil(frame,bin) = norm (c - pattern(3,:,bin));

        end

end
```

```
% Vector distancia.
dist(1,:,:) =   distMaj;
dist(2,:,:) =   distMin;
dist(3,:,:) =   distSil;

% Para cada muestra temporal.
for frame = 1:T

            % Inicializacion a infinito.
            distMinMaj = inf ('double');
            distMinMin = inf ('double');
            distMinSil = inf ('double');

                % Para cada muestra de croma.
                for bin = 1:12

                   % Encontramos la menor distancia.
                   if (dist(1,frame,bin) < distMinMaj)
                        distMinMaj = dist(1,frame,bin);

                        % Posicion del acorde mayor mas probable.
                        posChordMaj = bin;
                   end

                   if (dist(2,frame,bin) < distMinMin)
                        distMinMin = dist(2,frame,bin);

                        % Posicion del acorde menor mas probable.
                        posChordMin = bin;
                   end

                   if (dist(3,frame,bin) < distMinSil)
                        % Silencio
                        distMinSil = dist(3,frame,bin);
                   end

                end

                % Eleccion de la plantilla con menor distancia al
                % cromagrama obtenido.
                distMin =       [distMinMaj,distMinMin,distMinSil];
                [¬,typeID] =    min(distMin);

                % Seleccion de la etiqueta a escribir en tags.
                switch typeID
                    case 1
                        tags(frame) = strcat (rootTags(posChordMaj),':maj');
                    case 2
```

```
                    tags(frame) = strcat (rootTags(posChordMin),':min');
                case 3
                    tags(frame) = {'N'};
            end

end

end
```

Función estimationGen.m

```
function [estimname] = estimationGen(filename,chroma,tags)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              FUNCION DE GENERACION DEL FICHERO DE ESTIMACION
%
% Nombre:   estimationGen
%
% Input:    filename            Variable con el nombre del fichero.
%           chroma              Matriz que contiene el vector de cromagrama
%                               (12x1) obtenido para cada frame.
%           tags                Vector de cadena de caracteres que contiene
%                               la estimacion del acorde en cada frame.
%
% Output:   estimname           Nombre del fichero de estimacion generado.
%
% Descripcion:  Funcion encargada de generar, mediante la informacion
%               temporal de chroma, y de las etiquetas contenidas en tags,
%               el fichero de estimacion con la informacion de acorde en
%               cada frame.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Numero de frames.
[T,¬] =     size(chroma);
% Vector de tiempos.
time(:,1) = round(chroma(:,1),5,'significant');

% Generacion del fichero de estimacion.
fid = fopen([filename,'_estim.csv'],'w');

% Para cada frame.
for frame = 1:T

    % Escritura de la informacion temporal y de acorde.
    t = num2str(time(frame,1));
    fprintf(fid,'%s,%s\n',t,char(tags(frame)));

end

fclose(fid);

% Nombre del fichero.
estimname =    strcat (filename,'_estim.csv');

end
```

Función familyPatt.m

```
function [] = familyPatt (familyname)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             FUNCION DE GENERACION DE PLANTILLAS DE FAMILIAS GM
%
% Nombre:   familyPatt
%
% Input:    familyname          Variable con el nombre de la familia de
%                               instrumentos de la cual generar la
%                               plantilla.
%
% Descripcion:  Funcion encargada de generar, en funcion del nombre de
%               familyname, una plantilla resultado de la media de los
%               instrumentos de dicha familia General Midi.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc

% Localizacion de todos los ficheros WAVE de entrenamiento.
familypath =    strcat('dataset/train/',familyname,'/*.wav');
listing =       dir (familypath);

% Inicializacion de las matrices que contendran los cromas de cada
% instrumento y familia.
instChroMaj =   zeros (12,12);
famChroMaj =    zeros (12,12);
instChroMin =   zeros (12,12);
famChroMin =    zeros (12,12);

% Generacion de croma de todos los ficheros WAVE de entrenamiento.
for file = 1:(length(listing))
    filename =  listing(file).name;
    filepath =  strcat('dataset/train/',familyname,'/',filename);
    system (['./sonic-annotator -t config.txt ' filepath ' -w csv --csv-force 2> ...
        temp.txt']);
    movefile (['dataset/train/',familyname,'/',(strrep(listing(file).name,'.wav','...
        _vamp_nnls-chroma_nnls-chroma_chroma.csv'))],['dataset/train/',familyname,'/...
        chromas/']);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ACORDES DE TRIADA MAYOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Localizacion de los ficheros de croma de acordes de triada mayor
```

```
% pertenecientes a la familia.
familymajpath = strcat('dataset/train/',familyname,'/chromas/*maj*.csv');
listingmaj =       dir (familymajpath);

% Para cada fichero de croma mayor.
for fileMaj = 1:(length(listingmaj))

    % Carga de la informacion de croma, y de la informacion de Onset y
    % Offset.
    chroMaj =   load (['dataset/train/',familyname,'/chromas/',listingmaj(fileMaj).name...
        ]);
    timeMaj =   load (['dataset/train/',familyname,'/','onoff_',(strrep (listingmaj(...
        fileMaj).name,'_vamp_nnls-chroma_nnls-chroma_chroma.csv','.csv'))]);

    [T,¬] =     size(chroMaj);

    % Para cada elemento de croma.
    for bin = 1 : 12
        % Para cada instante temporal.
        for frame = 1:T
            % Acumulacion del valor del vector de croma para los frames
            % acotados al intervalo de Onset y Offset.
            if (chroMaj(frame,1)≥timeMaj(bin,1) && chroMaj(frame,1)<timeMaj(bin,2))
                instChroMaj(bin,:) = instChroMaj(bin,:)+chroMaj(frame,2:13);
            end
        end
        % Media normalizada del vector de croma de cada instrumento.
        instChroMaj(bin,:) = instChroMaj(bin,:)/(max(instChroMaj(bin,:)));
    end

    % Reordenacion de la matriz de croma para la creacion de la plantilla
    % del instrumento.
    instChroMaj =           transpose (instChroMaj);
    instPattMaj(3:12,:) =   instChroMaj(1:10,:);
    instPattMaj(1:2,:) =    instChroMaj(11:12,:);
    instPattern(fileMaj,1,:,:) =  instPattMaj;

    % Acumulacion de los vectores de croma de todos los instrumentos de
    % cada familia para realizar posteriormente su media aritmetica.
    famChroMaj =    famChroMaj+instChroMaj;
    instChroMajT(fileMaj,:) =  instChroMaj(1,:);
end

% Media normalizada del vector de croma de la familia de instrumentos.
for bin = 1:12
    famChroMaj(:,bin) = famChroMaj(:,bin)/(max(famChroMaj(:,bin)));
end

% Reordenacion de la matriz de croma para la creacion de la plantilla de la
```

```
% familia.
famPattMaj(3:12,:) =    famChroMaj(1:10,:);
famPattMaj(1:2,:) =     famChroMaj(11:12,:);
famPattern(1,:,:) =     famPattMaj;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ACORDES DE TRIADA MENOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Localizacion de los ficheros de croma de acordes de triada menor
% pertenecientes a la familia.
familyminpath = strcat('dataset/train/',familyname,'/chromas/*min*.csv');
listingmin =       dir (familyminpath);

% Para cada fichero de croma menor.
for fileMin = 1:(length(listingmin))

    % Carga de la informacion de croma, y de la informacion de Onset y
    % Offset.
    chroMin =   load (['dataset/train/',familyname,'/chromas/',listingmin(fileMin).name...
        ]);
    timeMin =   load (['dataset/train/',familyname,'/','onoff_',(strrep (listingmin(...
        fileMin).name,'_vamp_nnls-chroma_nnls-chroma_chroma.csv','.csv'))]);

    [T,¬] =     size(chroMin);

    % Para cada elemento de croma.
    for bin = 1 : 12
        % Para cada instante temporal.
        for frame = 1:T
            % Acumulacion del valor del vector de croma para los frames
            % acotados al intervalo de Onset y Offset.
            if (chroMin(frame,1)≥timeMin(bin,1) && chroMin(frame,1)<timeMin(bin,2))
                instChroMin(bin,:) = instChroMin(bin,:)+chroMin(frame,2:13);
            end
        end
        % Media normalizada del vector de croma de cada instrumento.
        instChroMin(bin,:) = instChroMin(bin,:)/(max(instChroMin(bin,:)));
    end

    % Reordenacion de la matriz de croma para la creacion de la plantilla
    % del instrumento.
    instChroMin =           transpose(instChroMin);
    instPattMin(3:12,:) =   instChroMin(1:10,:);
    instPattMin(1:2,:) =    instChroMin(11:12,:);
    instPattern(fileMin,2,:,:) =      instPattMin;
    instPattern(fileMin,3,:,1) =      [0 0 0 0 0 0 0 0 0 0 0 0];

    % Acumulacion de los vectores de croma de todos los instrumentos de
```

```
    % cada familia para realizar posteriormente su media aritmetica.
    famChroMin = famChroMin+instChroMin;
    instChroMinT(fileMin,:) = instChroMin(1,:);
end

% Media normalizada del vector de croma de la familia de instrumentos.
for bin = 1:12
    famChroMin(:,bin) = famChroMin(:,bin)/(max(famChroMin(:,bin)));
end

% Reordenacion de la matriz de croma para la creacion de la plantilla de la
% familia.
famPattMin(3:12,:) =    famChroMin(1:10,:);
famPattMin(1:2,:) =     famChroMin(11:12,:);
famPattern(2,:,:) =     famPattMin;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plantilla de silencio.
famPattern(3,:,1) =     [0 0 0 0 0 0 0 0 0 0 0 0];

% Generacion de los ficheros CSV con las plantillas de cada instrumento.
for file = 1:(length(listingmaj))
    csvwrite ((strrep(listingmaj(file).name,'_maj_vamp_nnls-chroma_nnls-...
        chroma_chroma.csv','.csv')),instPattern(file,:,:,:));
    movefile ((strrep(listingmaj(file).name,'_maj_vamp_nnls-chroma_nnls-...
        chroma_chroma.csv','.csv')),'patterns/instrument');
end

% Generacion del fichero CSV con la plantilla de la familia.
csvwrite ([familyname,'.csv'],famPattern);
movefile ([familyname,'.csv'],'patterns/family');

% Calculo estadistico sobre la dispersion de las plantillas mediante la
% funcion "standardDeviation".
standardDeviation (familyname,listingmaj,listingmin,instChroMajT,instChroMinT,famChroMaj...
    ,famPattMaj,famChroMin,famPattMin);

end
```

Función standardDeviation.m

```
function [] = standardDeviation (familyname,listingmaj,listingmin,instChroMajT,...
    instChroMinT,famChroMaj,famPattMaj,famChroMin,famPattMin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                FUNCION DE CALCULO DE LA DESVIACION TIPICA
%
% Nombre:   standardDeviation
%
% Input:    varios              Diversos ficheros necesarios para el
%                               calculo de las diferentes distancias.
%
% Descripcion:  Funcion encargada de realizar un estudio estadistico que
%               muestre cuan dispersos o compactos se encuentran cada uno
%               de los instrumentos de cada familia entre si (intra-family)
%               , asi como cada una de las familias (inter-family).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generacion y escritura del fichero de informacion estadistica.
fid = fopen (['STDdeviation_',familyname,'.txt'],'w');

% Para cada fichero de instrumentos con acordes mayores, se calcula la
% distancia entre cada uno de los instrumentos con su familia (media).
fprintf (fid,'%s\n\n','Intra-family distance (C major)');
for fileMaj = 1:(length(listingmaj))
    deviationMaj (fileMaj) = norm ( (instChroMajT(fileMaj,:)) - (famChroMaj(1,:)) );
    fprintf (fid,'Instrument %s standard deviation: %1.2f \n',(strrep(listingmaj(fileMaj...
        ).name,'_maj_vamp_nnls-chroma_nnls-chroma_chroma.csv','')),deviationMaj(fileMaj)...
        );
end

% Se obtiene el valor de compacidad de cada familia como el sumatorio de
% las distancias de todos los instrumentos que pertenezcan a esta.
fprintf (fid,'\nCompactness: %1.2f\n\n\n',(sum(deviationMaj)));

% Para cada fichero de instrumentos con acordes menores, se calcula la
% distancia entre cada uno de los instrumentos con su familia (media).
fprintf (fid,'%s\n\n','Intra-family distance (C minor)');
for fileMin = 1:(length(listingmin))
    deviationMin (fileMin) = norm ( (instChroMinT(fileMin,:)) - (famChroMin(1,:)) );
    fprintf (fid,'Instrument %s standard deviation: %1.2f \n',(strrep(listingmin(fileMin...
        ).name,'_min_vamp_nnls-chroma_nnls-chroma_chroma.csv','')),deviationMin(fileMin)...
        );
end

% Se obtiene el valor de compacidad de cada familia como el sumatorio de
```

```
% las distancias de todos los instrumentos que pertenezcan a esta.
fprintf (fid,'\nCompactness: %1.2f\n\n\n',(sum(deviationMin)));

% Directorio de los patrones pertenecientes a cada una de las familias de
% instrumentos.
famlisting =    dir ('patterns/family/*.csv');

% Para cada uno de los patrones se calcula la distancia entre este y los
% patrones pertenecientes al resto de familias de instrumentos.
fprintf(fid,'%s\n\n','Inter-family distance');
for fileFam = 1:(length(famlisting))
    famPattern =     csvread (['patterns/family/',famlisting(fileFam).name]);
    famPattern =     reshape (famPattern,[3 12 12]);
    famDistMaj =     norm ( (transpose(famPattMaj(:,1))) - (famPattern(1,:,1)) );
    famDistMin =     norm ( (transpose(famPattMin(:,1))) - (famPattern(2,:,1)) );
    fprintf (fid,'%s to %s distance (C major): %1.2f\n',familyname,(strrep(famlisting(...
        fileFam).name,'.csv','')),famDistMaj);
    fprintf (fid,'%s to %s distance (C minor): %1.2f\n',familyname,(strrep(famlisting(...
        fileFam).name,'.csv','')),famDistMin);
end

fclose (fid);

% Reordenacion del fichero generado.
movefile (['STDdeviation_',familyname,'.txt'],'results/deviation/');

end
```

Función clustering.m

```
function [] = clustering ()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                        FUNCION DE CLUSTERING
%
% Nombre:   clustering
%
% Descripcion:  Funcion encargada de realizar, en primer lugar un estudio
%               sobre la utilizacion de diferentes numeros de clusters (k)
%               para el algoritmo k-means, asi como de la obtencion de las
%               plantillas resultantes de la implementacion de dicho
%               algoritmo.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Estudio de determinacion de k en el algoritmo k-means
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Directorio con los cromas (en CSV) de todos los instrumentos.
listing =   dir ('patterns/instrument/*.csv');

% Etiquetado empleado en la representacion grafica.
rootTags =      {'C','C#','D','Eb','E','F','F#','G','Ab','A','Bb','B'};

% Lectura y remodelado de los cromas del directorio.
for file = 1:length(listing)
    pattern =  csvread (['patterns/instrument/',listing(file).name]);
    pattern =     reshape (pattern,[3 12 12]);
    patternMaj(file,:,:) = pattern(1,:,:);      % Cromas de acordes mayores.
    patternMin(file,:,:) = pattern(2,:,:);      % Cromas de acordes menores.
end

% Para cada raiz de acorde.
for root = 1:12

    % Para cada valor posible de k.
    for k = 1:80

        % Implementacion de k-means en acordes mayores.
        [¬,¬,distCentMaj] =     kmeans (patternMaj(:,:,root),k);
        Jmaj(k) =     sum(distCentMaj);

        % Implementacion de k-means en acordes menores.
```

```
        [¬,¬,distCentMin] =     kmeans (patternMin(:,:,root),k);
        Jmin(k) =     sum(distCentMin);

    end

    % Representacion grafica de J.
    subplot (4,3,root)
    plot    ((1:80),Jmaj/max(Jmaj),(1:80),Jmin/max(Jmin));
    title   (rootTags(root));
    xlabel  ('k');
    ylabel  ('J');

end

legend  ('Major','Minor');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Implementacion del algoritmo k-means para re-clasificacion de familias
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Numero de clusters.
k =             9;

% Implementacion de k-means (k=9) en acorde de Do mayor.
[idMaj,centMaj] =       kmeans (patternMaj(:,:,1),k);

% Implementacion de k-means (k=9) en acorde de Do menor.
[idMin,centMin] =       kmeans (patternMin(:,:,1),k);

% Para cada cluster.
for cluster = 1:k

    % Definicion de plantilla con el centroide de la agrupacion mayor.
    clustPatt (1,:,1) =     centMaj(cluster,:)/max(centMaj(cluster,:));
    % Definicion de plantilla con el centroide de la agrupacion menor.
    clustPatt (2,:,1) =     centMin(cluster,:)/max(centMin(cluster,:));
    % Definicion de la plantilla de silencio.
    clustPatt (3,:,1) =     [0 0 0 0 0 0 0 0 0 0 0 0];

    % Rotacion del acorde para la generacion de las plantillas.
    for pos = 2:12

        clustPatt(1,2:12,pos) = clustPatt(1,1:11,pos-1);     % Triada mayor.
        clustPatt(1,1,pos) =   clustPatt(1,12,pos-1);

        clustPatt(2,2:12,pos) = clustPatt(2,1:11,pos-1);     % Triada menor.
        clustPatt(2,1,pos) =   clustPatt(2,12,pos-1);

    end
```

```
    % Generacion y re-ordenacion del fichero CSV con la plantilla del
    % clustering.
    csvwrite ([(num2str(cluster)),'.csv'],clustPatt);
    movefile ([(num2str(cluster)),'.csv'],'patterns/clustering');

end

% Generacion y escritura del fichero con la clasificacion.
fid = fopen ('clustering.txt','w');
fprintf (fid,'Cluster Clasification\n\n');

% Para cada instrumento
for inst = 0:79

    % Se escribe en el fichero la correspondencia de cada instrumento al
    % cluster.
    fprintf (fid,'\n Instrument %d (C Maj) belongs to cluster %i',inst,idMaj(inst+1));
    fprintf (fid,'\n Instrument %d (C Min) belongs to cluster %i\n',inst,idMin(inst+1));

end

fclose(fid);

% Re-ordenacion del fichero generado.
movefile ('clustering.txt','results/');

end
```

Función GTMetrics.m

```
function [accuracy,duration] = GTMetrics (filename)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 FUNCION DE EVALUACION DE LA PRECISION
%
% Nombre:   GTMetrics
%
% Input:    filename            Variable con el nombre del fichero.
%
% Output:   accuracy            Variable con el valor de la tasa de acierto
%                               obtenida para el fichero procesado.
%           duration            Variable con el valor de la duracion del
%                               fichero procesado.
%
% Descripcion:  Funcion encargada de evaluar la precision del sistema de
%               estimacion de acordes, comparando el fichero de estimacion
%               generado con el fichero de Ground-Truth.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Localizacion de los ficheros de estimacion y Ground-Truth.
EstimatedChordsName =   strcat (filename,'_estim.csv');
GTChordsName =          strcat ('GT_',filename,'.csv');
GTChordsPath =          strcat ('dataset/gt/',GTChordsName);

% Lectura del fichero de GT y delimitacion del contenido de los vectores
% por las comas.
file = fopen(GTChordsPath);
    GTFile =        textscan(file,'%f %f %s','delimiter',',');
    GTIniTime =     GTFile{1};      % Vector de escala temporal inicial.
    GTFinTime =     GTFile{2};      % Vector de escala temporal final.
    GTChords  =     GTFile{3};      % Vector de acordes.
fclose(file);

% Lectura del fichero de estimacion y delimitacion del contenido de los
% vectores por las comas.
file = fopen(EstimatedChordsName);
    EstimFile =     textscan(file,'%f %s','delimiter',',');
    EstimTime =     EstimFile{1};   % Vector de escala temporal.
    EstimChords =   EstimFile{2};   % Vector de acordes.
fclose(file);

% Estimacion de la longitud de la escala temporal.
estimT =        length (EstimChords);
GTT =           length (GTChords);
```

```
% Inicializacion del parametro de precision y del contador de frames
% evaluados.
accuracy =      0;
contFrames =    0;

% Para todas las muestras del fichero GT.
for GTFrame = 1:GTT

    % Separacion de raiz y tipo.
    [GTRoot,¬] = roottypeSplit (GTChords(GTFrame),':');

    % Para todas las muestras del fichero de estimacion.
    for estimFrame = 1:estimT

        % Separacion de raiz y tipo
        [ECRoot,¬] = roottypeSplit (EstimChords(estimFrame),':');

        % Si el instante temporal del fichero estimado se encuentra
        % comprendido entre los tiempos de Onset y Offset del GT.
        if EstimTime(estimFrame)≥GTIniTime(GTFrame) && EstimTime(estimFrame)<GTFinTime(...
            GTFrame)

            % Si coincide el nombre del acorde
            if strcmp (EstimChords(estimFrame),GTChords(GTFrame))
                % Incremento de 1 la precision.
                accuracy =  accuracy+1;
            else

                % Si solo coincide la raiz del acorde.
                if strcmp(ECRoot,GTRoot)
                    % Incremento de 1/2 la precision.
                    accuracy =  accuracy+0.5;
                end
            end

            % Siempre que el tiempo de estimacion y GT coincidan, se
            % incrementa el contador de frames.
            contFrames = contFrames+1;

        end
    end
end

% Duracion del fichero de audio.
duration =  GTFinTime(GTT);

% Calculo y muestreo del porcentaje de precision.
accuracy =  round(double(100*(accuracy/contFrames)),2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCION DE SEPARACION RAIZ-TIPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [CR,CT] = roottypeSplit (C,delim)

CharC = char(C);
CR = '';
CT = '';

% Recorrido por la cadena de caracteres.
for c=1:(length(CharC))

    % Cuando se localiza el parametro delimitador.
    if (strcmp(CharC(c),delim));
        % Se establece su posicion como pivote.
        pos = c;
        % Separando en dos cadenas el contenido anterior y posterior.
        CR = CharC(1:pos-1);
        CT = CharC(pos+1:(length(CharC)));
    end

end

end

end
```