Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República Oriental del Uruguay

Separación de Voz Cantada (Singing Voice Separation)

Proyecto de fin de carrera
Plan 1997

Andrés Samas
Alessandro Palermo
Ariel Decarlini

Tutores:

Martín Rocamora
Ernesto López
Álvaro Tuzman

Montevideo, Julio 2008

Agradecimientos

Agradecemos a nuestros tutores Ernesto López y Martín Rocamora por el apoyo y la motivación inicial en este proyecto de fin de carrera.

En particular caben destacar los aportes y sugerencias de Ernesto en el arranque del proyecto, y muy especialmente a Martín quien apoyó al grupo en todo momento. Durante todo el transcurso del proyecto estuvo encima nuestro, preocupándose para que las tareas salieran adelante de manera exitosa.

Para los dos muchas gracias por todo el conocimiento brindado en esta área totalmente nueva para nosotros.

Índice general

Capítulo 1

Introducción

El problema principal que abordó este proyecto de fin de carrera es la extracción de la voz cantada en una grabación musical. El objetivo es construir un sistema que reciba como entrada un archivo de música, y que devuelva como salida la voz cantada. Con el fin de desarrollar la solución, se realizó un estudio de los enfoques existentes y finalmente se optó por implementar el propuesto en [1].

En este capítulo se presenta una motivación al problema, considerando las posibles aplicaciones, así como también la relación con el problema de la separación de voz hablada. Luego se da cuenta del trabajo realizado en este proyecto, y finalmente se describe cómo se organiza este documento.

1.1. Motivación

El sistema auditivo humano tiene la capacidad de separar los sonidos de diferentes fuentes acústicas, en especial puede escuchar y seguir la voz cantada en presencia de un acompañamiento musical. Esta tarea le requiere poco esfuerzo, sin embargo, un sistema computacional que realice lo anterior no es algo trivial, y se han propuesto pocas soluciones.

En la actualidad hay un gran tráfico de datos digitales, esto es debido a que los algoritmos de compresión de datos existentes tanto para audio, por ejemplo `mp3`®,[1] y para video, por ejemplo `MPEG-4`,[2] así como también los formatos contenedores multimedia como `avi`[3] y `ogg`,[4] permiten almacenar grandes cantidades de información multimedia. Asimismo, con el surgimiento de dispositivos portátiles reproductores de audio y video,[5] junto con el uso masivo de Internet, estos formatos se han popularizado y son de uso corriente.

[1] MPEG-1 Audio Layer 3 es un formato de audio digital comprimido con pérdida, fue desarrollado por el Moving Picture Experts Group (MPEG)

[2] Es un estándar de compresión de video desarrollado por MPEG.

[3] Audio Video Interleave, es un formato de archivo contenedor de audio y vídeo lanzado por Microsoft® en 1992.

[4] Ogg es un formato de archivo contenedor multimedia, fue desarrollado por la Fundación Xiph.org. A diferencia del formato mp3®, es libre de patentes y abierto.

[5] Ipod®, por ejemplo

Ante este gran volumen de datos circulando por las redes, se tiene la necesidad de implementar sistemas que permitan realizar búsquedas por contenido, ya sea con el fin del ordenamiento, del almacenamiento o de la clasificación de los datos. Esto permitirá luego, por ejemplo, buscar en una base de datos algún archivo con cierta característica, como ser canciones de un mismo cantante o de un mismo género. Un sistema de separación de voz cantada es muy útil para realizar esta tarea, ya que la voz cantada contiene información como la melodía. Por lo tanto se podría usar como un primer bloque de procesamiento para este tipo de sistemas. Por ejemplo, la extracción de la voz cantada podría ir seguida de algún sistema de búsqueda de canciones mediante tarareo, como el presentado en [2].

Otras áreas de aplicación son el reconocimiento automático de las letras de las canciones y el alineamiento. Los sistemas de reconocimiento de letras a menudo necesitan que la entrada sea la voz cantada [3]. El alineamiento de letras con la voz cantada es una tarea fundamental para sistemas del tipo *karaoke*, tal proceso es muy difícil cuando está presente el acompañamiento, sin embargo, al tener la voz cantada separada se pueden lograr mejores resultados [4]. También se encuentra especial uso en la identificación de cantantes [5], [6].

El problema de la separación de voz cantada es un problema que está ligado directamente con un problema similar pero menos complejo, que es el de la separación de voz hablada. Este último intenta separar la voz hablada de alguna o algunas otras fuentes interferentes no correlacionadas con ella. Un problema típico de este tipo es el de *cocktail party*, el cual fue introducido en la década del 50 por Cherry [7], [8]. El mismo consiste en separar una voz objetivo ante la presencia de otras voces y sonidos de fondo. Dicha situación se tiene cuando se establece un diálogo con otra persona en un ambiente donde hay mucha gente.

La diferencia fundamental entre el problema de separación de voz hablada y la separación de voz cantada, es que en el caso de la voz hablada las distintas fuentes que interfieren con la voz objetivo no están correlacionadas con ella. Por otro lado, en el problema de separación de voz cantada, debido al hecho de que en una canción la melodía se compone con cierto ritmo y que el cantante esfuerza su voz para ajustarse al acompañamiento, se tiene como consecuencia que la señales de voz cantada y de acompañamiento musical estén altamente correlacionadas. Dicha correlación es tanto en el tiempo como en frecuencia, ya que se busca tener tanto sincronismo temporal, así como también un solapamiento frecuencial de los distintos armónicos pertenecientes a la voz cantada y a los instrumentos musicales.

A diferencia del problema de separación de voz hablada, que ha sido ampliamente estudiado debido a su gran aplicación en las telecomunicaciones, el problema de separación de voz cantada es un problema relativamente nuevo, lo cual es una gran motivación para profundizar y seguir su desarrollo.

1.2. Trabajo realizado

El proyecto se puede dividir en cuatro grandes etapas. Al no contar la facultad con un curso formal de temas de audio, en una primera etapa se realizó el estudio de los

aspectos concernientes a las señales auditivas, los modelos de generación del habla y el sistema auditivo humano. En particular se consultó material de la Universidad de Surrey acerca del análisis del habla [9].

La segunda etapa consistió en la búsqueda de información sobre el problema planteado inicialmente. Se encontraron aquí distintos enfoques, se pudo ver que hay distintas corrientes y que una de las más estudiada y de la que se encuentra más teoría es la basada en el análisis de la escena auditiva computacional (CASA, por sus siglas en inglés *Computational Auditory Scene Analysis*) [10], [11]. Este enfoque se basa en modelos computacionales que intentan imitar el comportamiento del sistema auditivo humano. Como parte final de esta etapa, se estudió la viabilidad de cada una de las implementaciones de los distintos métodos, optándose finalmente por el artículo escrito por Li-Wang [1], ya que resultó ser el más completo y el que a priori presentaba mejores resultados.

La tercera etapa del proyecto fue la implementación en Matlab® del enfoque propuesto en [1]. En esta etapa se tuvo que analizar a fondo la teoría e investigar la manera de llevar a cabo las implementaciones. Desde el punto de vista de la dificultad y tiempo requerido, esta etapa fue la más exigente. Se obtuvieron buenos resultados. Las medidas de desempeño son comparables con las del artículo estudiado [1], además las señales de voz cantada extraídas de los archivos musicales son inteligibles.

Finalmente, la última etapa consistió en documentar todo el proceso y en recopilar toda la información utilizada. Se puso especial énfasis en la documentación ya que los temas aquí desarrollados no se encuentran muy difundidos, tratando de dejar una buena base para trabajos futuros. También se creó una pequeña base de datos con fragmentos musicales con el fin de testear el algoritmo implementado, la cual puede verse en el apéndice G.

1.3. Descripción del documento

A continuación se presenta una descripción de cómo se organiza el resto del documento.

En el capítulo 2 se describe una introducción al enfoque propuesto en [1]. Se desarrolla la técnica utilizada para la extracción de la voz cantada, se presenta un diagrama de bloques de la solución y se da una introducción a cada uno de ellos.

En los capítulos 3, 4, 5, 6, 7 y 8 se detallan cada uno de los bloques presentados en el capítulo 2.

En el capítulo 9 se presenta el método utilizado para la evaluación del algoritmo implementado y los resultados obtenidos.

En el capítulo 10 se señalan las conclusiones de todo el trabajo realizado, se comentan las mejoras a realizar sobre el algoritmo implementado y se describen las posibles líneas de trabajo a futuro.

Capítulo 2

Introducción al Método Implementado

El enfoque propuesto en [1] es el que se decidió implementar para resolver el problema de separación de voz cantada. A continuación se presenta un resumen de la técnica utilizada.

El objetivo es separar la voz cantada del acompañamiento musical en el caso monoaural, es decir, cuando se tiene una grabación musical en un sólo canal. Ésta pudo haber sido obtenida al grabar utilizando un micrófono solamente, o como resultado de una mezcla final de audio. Como señales de entrada se utilizan archivos de audio muestreados a 16 kHz, los cuales deben ser fragmentos musicales de corta duración que contengan voz cantada y acompañamiento al mismo tiempo. Esto es debido a que el sistema presenta un gran costo computacional, y la demora para procesar archivos de más duración es considerable.

El punto de partida de la teoría sobre la que se basa el artículo se encuentra en los resultados de los estudios psicofísicos del sistema auditivo humano, en los cuales se busca conocer la forma en que nuestro sistema auditivo separa las distintas fuentes sonoras que constituyen la entrada acústica. El libro más influyente es el escrito por Bregman [10], en este libro se propone que el sonido que llega al oído humano está sujeto a un proceso llamado análisis de la escena auditiva (ASA, por sus siglas en inglés *Auditory Scene Analysis*). Este proceso se realiza en dos grandes etapas: segmentación y agrupamiento. En la etapa de segmentación, la entrada acústica es descompuesta en una serie de regiones tiempo-frecuencia (T-F) locales, llamadas segmentos. Cada uno de ellos se espera que sea originado por una sola fuente de la entrada acústica. La segunda etapa es la de agrupamiento, en donde los segmentos que probablemente pertenezcan a una misma fuente son agrupados juntos, formando unas estructuras perceptuales llamadas *stream* para cada fuente sonora. La segmentación y el agrupamiento son llevados a cabo por mecanismos perceptuales, que determinan cómo la escena auditiva es organizada de acuerdo a principios auditivos del ASA.

A partir del ASA, se ha investigado y se han desarrollado sistemas para obtener la representación computacional de la teoría del análisis de la escena auditiva. Se abordó a lo que se conoce como análisis de la escena auditiva computacional (CASA, por sus siglas en inglés *Computational Auditory Scene Analysis*) [12], [13], [14], [11]. El objetivo de los

sistemas CASA es realizar la separación de las distintas fuentes sonoras que componen la entrada acústica. Se tiene por tanto una señal acústica de entrada compuesta por varias fuentes sonoras, y el objetivo es obtener a la salida cada una de ellas en forma separada. Una de las grandes ventajas que tienen los sistemas CASA frente a otros métodos es que no se suponen grandes restricciones sobre las propiedades acústicas de las señales de entrada, lo cual permite abordar un espectro amplio de problemas. En la figura 2.1 se puede ver un diagrama de bloques de un sistema CASA típico.

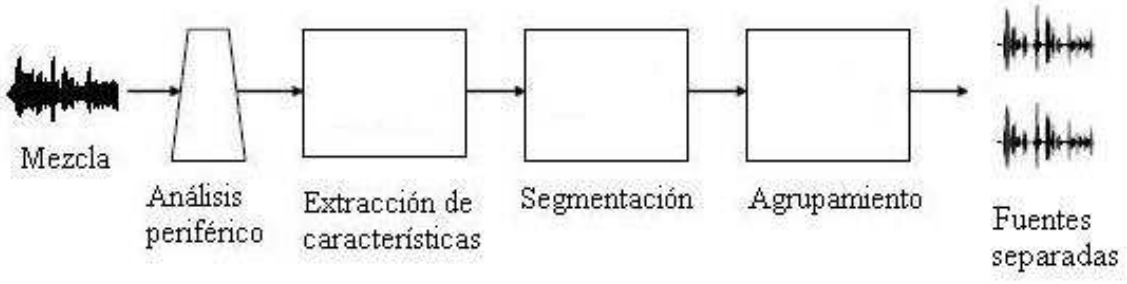

Figura 2.1: Sistema CASA.

En la etapa de procesamiento periférico se descompone la entrada acústica en una representación T-F, que se realiza a través de filtrados pasabandas y un proceso de enventanado temporal. En la segunda etapa se extraen características de acuerdo a principios auditivos del ASA. Finalmente en las etapas de segmentación y agrupamiento, utilizando las características halladas anteriormente, el sistema genera segmentos para las distintas fuentes acústicas, y luego se agrupan los segmentos perteneciente a cada fuente en distintos *stream*. Una vez que se tiene el *stream* de cada fuente, el próximo paso consiste en obtener la forma de onda temporal de cada una de ellas.

El artículo que se desarrolla en este documento es un caso particular de un sistema CASA. Se considera que la entrada acústica está compuesta por dos fuentes sonoras, la voz cantada y el acompañamiento musical. El objetivo es lograr extraer la señal de voz cantada de la entrada acústica.

El sistema realiza la separación del instrumento armónico predominante. Se tienen que cumplir por tanto dos hipótesis para el correcto funcionamiento del algoritmo de separación. La primera es que esté presente en el fragmento de audio la señal de voz cantada, ya que de otra manera se realizaría la separación de algún otro instrumento armónico. Asimismo para que se extraiga correctamente la señal de voz cantada, ésta debe predominar por sobre el acompañamiento musical.

El artículo está fuertemente basado en [15] y [16], los cuales presentan un sistema para separar la voz hablada de alguna interferencia acústica. Cabe destacar que se decidió basarse en [16] para realizar la implementación del algoritmo de separación. Se realizaron las modificaciones y las adaptaciones necesarias para el caso de la separación de voz cantada. En la figura 2.2 se puede observar un diagrama de bloques de punta a punta del enfoque propuesto en [1].

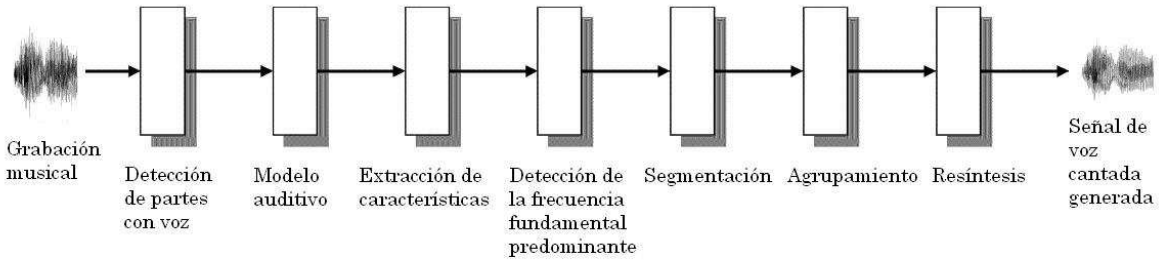

Figura 2.2: Diagrama de bloques del enfoque propuesto por Li-Wang en [1].

El primer bloque tiene como objetivo particionar la señal de entrada en regiones en donde esté presente la voz cantada, de forma tal de quedarse con estos fragmentos y utilizarlos a lo largo de todo el algoritmo, ya que en las partes subsiguientes se requiere contar con una señal que contenga la voz cantada y el acompañamiento musical al mismo tiempo. Dicho bloque no fue implementado, debido a que el desarrollo del mismo suponía un gran esfuerzo, sumado al hecho de que se procesan fragmentos de corta duración (por lo que se decidió poner como restricción al sistema que la señal de entrada contenga voz cantada y acompañamiento musical al mismo tiempo).

Antes de proseguir, es bueno aclarar que el sistema separa solamente las partes de la señal de voz que son sonoras, no así las partes que son sordas.[1] El algoritmo hace la separación en base a la detección de la frecuencia fundamental predominante, por lo que está pensado para realizar la separación de una señal armónica. Los sonidos sonoros son periódicos o cuasi-periódicos, es por ello que el sistema lleva a cabo su separación. Por otro lado, los sonidos sordos carecen de una estructura armónica, y para realizar su separación se deben utilizar otras técnicas, en [16] se presenta una forma de llevar a cabo esta tarea. Esta limitación no es tan severa, ya que en diversos estilos musicales el porcentaje de sonidos sonoros en la voz cantada llega al 90 %. Por lo mencionado anteriormente hay que tener en cuenta que cada vez que se hable de la separación de la señal de voz cantada, se estará hablando de la separación de las partes sonoras de dicha señal. Las partes sordas quedan por tanto relegadas de la separación, incurriendo en errores perceptibles.

La primera etapa del sistema consiste en realizar una representación en tiempo-frecuencia de la señal de entrada, para ello se utiliza un modelo computacional del sistema auditivo humano. Éste consta de un banco de filtros pasabandas llamados gammatone que imitan el filtrado que realiza la cóclea en el oído humano [17], seguido por el modelo de Meddis que simula los procesos no lineales que allí suceden [18]. Los anchos de banda de los filtros se incrementan cuasi-logarítmicamente, a medida que aumenta su frecuencia central. Por lo tanto, para una señal armónica se diferencian dos casos, en el rango de las bajas frecuencias un filtro auditivo tiene un ancho de banda angosto y generalmente contiene sólo un armónico, mientras que en el rango de las altas frecuencias tiene un ancho de banda más amplio y usualmente contiene múltiples armónicos.

[1] Por más información sobre la voz y el sistema auditivo ver el apéndice A.

Un armónico es llamado resuelto si existe un canal del banco de filtros que responde primariamente a él, en caso contrario, es llamado no resuelto. Se puede ver entonces que con el modelo del banco de filtros auditivos, una serie armónica es dividida en armónicos resueltos y no resueltos. En el rango de las bajas frecuencias se encuentran los primeros armónicos, en general son resueltos ya que en cada banda hay sólo uno de ellos. Por otro lado, en el rango de las altas frecuencias es encuentran los armónicos más altos, son frecuentemente no resueltos ya que en una misma banda se combinan varios [19], [20]. Tanto en el caso de voz hablada como cantada, el valor que se toma para diferenciar el rango de las bajas y las altas frecuencias es generalmente de 1 kHz, y es por tanto el que se utiliza en el algoritmo de separación.

A la salida de cada filtro, la señal es dividida en tramas temporales consecutivas. Este proceso resulta en una descomposición de la señal en un mapa de dos dimensiones, tiempo y frecuencia. Cada unidad del mapa es llamada unidad T-F, que corresponde a un determinado canal del banco de filtros en una determinada trama temporal. A partir de esta descomposición T-F, se define el objetivo computacional del sistema, el cual consiste en retener las unidades T-F en donde la voz predomina por sobre el acompañamiento, y remover las otras. Lo anterior se puede ver también como el hecho de identificar una máscara binaria en donde un 1 indica que la voz predomina sobre el acompañamiento, y un 0 indica lo contrario. Esta máscara es llamada máscara ideal binaria (IBM, por sus siglas en inglés *Ideal Binary Mask*) [11], [21]. La base de lo planteado anteriormente se encuentra en lo que se denomina fenómeno de enmascaramiento auditivo, el cual dice que dentro de una banda crítica[2] una señal tiende a ser enmascarada por otra más fuerte [22]. La máscara binaria estimada que se obtiene en todo el proceso de separación se pretende que se parezca lo más posible a la IBM.

Una propiedad muy importante es que la respuesta de los filtros auditivos a múltiples armónicos es modulada en amplitud, y la frecuencia de la envolvente de dicha respuesta es la frecuencia fundamental (f_0)[3] [23]. Además, estudios psicofísicos muestran que el sistema auditivo humano utiliza diferentes mecanismos para procesar armónicos resueltos y no resueltos [19], [24]. Los primeros sistemas basados en CASA empleaban la misma estrategia para tratar a los diferentes armónicos, esa técnica funcionaba razonablemente bien para los armónicos resueltos pero tenía un pobre desempeño para los armónicos no resueltos. Tomando en cuenta lo anterior, el artículo propone utilizar distintos métodos para realizar la separación de los armónicos resueltos y de los no resueltos, o lo que es lo mismo, se trabaja de distintas maneras en bajas y en altas frecuencias.

En el siguiente bloque se extraen características para cada unidad T-F, las cuales son utilizadas en las etapas posteriores. Una de ellas es una estructura llamada correlograma, que consiste en hallar la autocorrelación de las respuestas de los filtros en cada unidad, y es utilizada como medida de la periodicidad de la señal. La otra característica es la correlación cruzada entre canales, la cual mide la similitud entre las autocorrelaciones de dos canales adyacentes, o lo que es lo mismo, la similitud entre los patrones de periocidad. Es utilizada como indicador de si dos canales adyacentes responden a

[2]Por más información sobre bandas críticas ver el apéndice C.

[3]La suma de dos señales sinusoidales puede verse como una señal de frecuencia intermedia, cuya amplitud está modulada por una sinusoide de frecuencia igual a la resta de las frecuencias originales.

un mismo componente acústico. Ambas características son calculadas en los canales de bajas frecuencias. Teniendo en mente lo explicado en párrafos anteriores, en altas frecuencias estas características no son buenos indicadores, debido a que las respuestas de los filtros están moduladas en amplitud y su envolvente fluctúa a f_0. Por lo tanto en altas frecuencias se calculan las mismas características, pero sobre la envolvente de amplitud de la respuesta de los filtros.

El bloque de detección de la frecuencia fundamental predominante o *pitch* merece una consideración especial, ya que desde el punto de vista del esfuerzo de implementación y de su base teórica, es tan importante como todo el algoritmo de separación. Este bloque es utilizado en la etapa de agrupamiento. Está fuertemente basado en [25], el cual a su vez está basado en [26]. El objetivo es obtener la frecuencia fundamental predominante en cada trama temporal, la cual se espera que pertenezca a la voz cantada, ya que la misma tiende a ser dominante cuando está presente. Es posible que en alguna trama la frecuencia fundamental del acompañamiento sea más dominante que la de la voz, en este caso se incurriría en errores en la separación.

El próximo bloque es el de segmentación. Aquí se busca agrupar regiones contiguas de unidades T-F, denominadas segmentos, con el objetivo de que pertenezcan a un mismo componente acústico de una misma fuente sonora. Luego en la etapa de agrupamiento se decide si los segmentos formados pertenecen o no a la voz cantada. Para realizar dicha segmentación se toman en cuenta dos aspectos. El primero es que las señales de audio tienen cierta duración en el tiempo y son continuas, por lo tanto las unidades T-F que son vecinas en el tiempo tienden a ser originadas por la misma fuente. La segunda cuestión es que debido a que las bandas de paso de los filtros auditivos tienen bastante solapamiento, un armónico activa varios canales adyacentes. Esto se ve reflejado en que los canales del correlograma que estén cerca de la frecuencia de dicho armónico van a presentar similares patrones de periocidad, y eso lleva a que tengan una alta correlación cruzada entre canales. Por lo tanto también las unidades vecinas en frecuencia tienden a ser originadas por la misma fuente. Esto sucede en los canales de baja frecuencia en donde los armónicos son resueltos, por lo que en estos canales se formarán segmentos teniendo en cuenta la continuidad temporal y la correlación cruzada entre canales. En los canales de alta frecuencia, serán las envolventes de las respuestas de los filtros las que presenten similares patrones de periocidad en canales adyacentes. Es así que en los canales de alta frecuencia se realiza la segmentación en base a la continuidad temporal y a la correlación cruzada entre las envolventes de los canales.

La etapa de agrupamiento tiene como objetivo identificar los segmentos que fueron originados por la señal de voz, y agruparlos en una estructura llamada *stream*, la cual es una máscara binaria en donde las unidades T-F que se hayan identificado como pertenecientes a la voz toman el valor de 1, mientras que las restantes toman el valor de 0. El objetivo es que el *stream* formado se parezca lo más posible a la máscara ideal binaria. En una primera etapa se etiquetan las unidades T-F de los segmentos, ya sea como voz dominante o acompañamiento dominante, se utiliza para ello la frecuencia fundamental predominante. Se procede de distinta manera según los segmentos hayan sido formados teniendo en cuenta si sus armónicos eran resueltos o no. En las bajas frecuencias, para etiquetar una unidad se compara su periocidad con el período del

pitch predominante en esa trama. Por otro lado, en las altas frecuencias, teniendo en mente lo que se ha dicho hasta ahora, lo que se compara es la periocidad de la envolvente con el período del *pitch* predominante. Finalmente se agrupa un segmento en el *stream* final si la suma de la energía acústica correspondiente a sus unidades etiquetadas como voz dominante, es mayor a la energía total del segmento.

La última etapa es la de resíntesis. El objetivo es reconstruir la señal de voz cantada a partir de las regiones T-F del *stream* final. Se utiliza para ello la salida del banco de filtros gammatone. El objetivo es quedarse con las partes de la señal en cada filtro en donde la voz predomina sobre el acompañamiento, o lo que es lo mismo, con las muestras contenidas dentro de las unidades T-F con valor 1 en el *stream*. Por último se suman las señales que resultan de este proceso en todos los canales, y se obtiene la forma de onda en el dominio del tiempo [27], [28], [11].

Capítulo 3

Modelo Auditivo

En este capítulo se describe la primera etapa del procesamiento de la señal de entrada, en la cual se realiza una representación en tiempo-frecuencia de la misma. Esto se logra usando un modelo computacional del sistema auditivo periférico. El objetivo es hacer un análisis en el dominio de la frecuencia que sea consistente con las propiedades de selectividad del sistema auditivo [11].

Se presentan dos importantes modelos de uso corriente en la resolución de los problemas de separación y análisis de voz, especialmente utilizados en los enfoques de tipo ASA (*Auditory Scene Analysis*). Estos modelos son, el banco de filtros auditivos, los cuales imitan el filtrado que realiza la cóclea en el oído humano, y el modelo de Meddis, que modela procesos no lineales que ocurren en el sistema auditivo. Ambos se utilizan en conjunto como un primer bloque en esta clase de problemas, ya que representan un buen compromiso entre exactitud y eficiencia computacional.

Este capítulo se organiza de la siguiente manera. En la sección 3.1 se describen los filtros auditivos utilizados, mientras que en la sección 3.2 se introduce el modelo de Meddis, luego en la sección 3.3 se dan los detalles de la implementación de cada uno y se muestran resultados y conclusiones. En la sección 3.4 se explica la descomposición de la señal de entrada en unidades T-F, y finalmente en la sección 3.5 se introduce el concepto de cocleagrama, el cual es una representación de la señal como una matriz tiempo-frecuencia.

3.1. Filtros gammatone

Los filtros gammatone, también llamados filtros auditivos,[1] surgen de modelos psicofísicos y representan al filtrado en el dominio de la frecuencia que hace la cóclea en el oído humano. Son un banco de filtros con bandas de paso no uniformes y solapadas, en donde cada filtro o canal modela la respuesta en frecuencia asociada con un punto en particular de la membrana basilar [11].

El modelo que se utiliza para la construcción de los filtros es el propuesto en [17], la respuesta al impulso de cada uno de ellos viene dada por:

[1]De aquí en adelante se hablará indistintamente de filtros gammatones o filtros auditivos.

$$g(t) = t^{l-1}e^{-2\pi t 1{,}019 ERB}cos(2\pi f_c t) \qquad \text{con } t \geq 0 \tag{3.1}$$

Donde $l = 4$ es el orden del filtro y f_c es su frecuencia central asociada. Por otro lado, el ancho de banda de cada filtro es ajustado de acuerdo a las medidas del ancho de banda rectangular equivalente (ERB por sus siglas en inglés *Equivalent Rectangular Bandwidth*)[2] de los filtros auditivos humanos [29]. Están relacionados con las bandas críticas[3] y dependen de cada frecuencia central, su expresión es la siguiente:

$$ERB(f_c) = 24{,}7(4{,}37\frac{f_c}{1000} + 1)\ Hz \tag{3.2}$$

Las frecuencias centrales se distribuyen uniformemente en una escala de bandas críticas, la cual se deriva de la ecuación 3.2. Es una función aproximadamente logarítmica, que relaciona la frecuencia con el número de canales del banco de filtros. Está dada por:

$$\xi(f) = 21{,}4\log_{10}(0{,}00437f + 1) \tag{3.3}$$

Para realizar la implementación se utilizó un banco de 128 (canales) filtros gammatone, con las frecuencias centrales dentro de un rango de interés de entre 80 y 5000 Hz. Por lo tanto los límites en la escala de banda crítica son $\xi(80Hz) = 2{,}786$ y $\xi(5000Hz) = 29{,}08$. Como se quieren tener 128 filtros en esa banda, la separación entre dos frecuencias centrales en dicha escala debe ser de $\frac{\xi(5000Hz)-\xi(80Hz)}{128-1} = 0{,}207$. Si se invierte la ecuación 3.3, se obtiene una expresión que permite obtener la frecuencia central de cada canal:

$$f(\xi) = 229(10^{\xi/21{,}4} - 1)\ Hz \tag{3.4}$$

Donde ξ varía entre 2.786 y 29.08, con pasos de 0.207.

En el apéndice D se encuentra una tabla con los correspondientes valores de la frecuencia central y el ERB para cada canal. En la figura 3.1 se puede observar una representación gráfica de estos valores en función del número de canal. El canal número 1 corresponde a la frecuencia más alta y el canal 128 a la frecuencia más baja.

[2]El ERB de un filtro dado, se define como el ancho de banda de un filtro rectangular ideal con la misma ganancia de pico, y que deja pasar la misma cantidad de potencia cuando la señal de entrada es ruido blanco.

[3]Por más información sobre las bandas críticas ver el apéndice C.

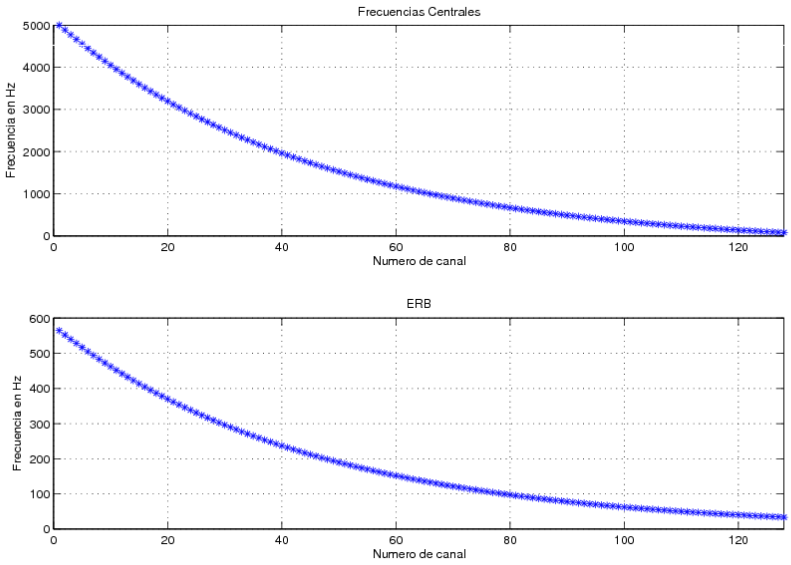

Figura 3.1: Frecuencias centrales y ERB vs número de canal. Cabe destacar que los canales más bajos corresponden a las frecuencias más altas.

Se puede apreciar que las frecuencias centrales quedan distribuidas en una escala cuasi-logarítmica, entre los valores de 80 y 5000 Hz. Del mismo modo, los anchos de banda de los filtros también quedan distribuidos en una escala cuasi-logarítmica. Para valores de frecuencias bajas los filtros presentan anchos de banda angostos, y luego en los valores de frecuencias más altas los anchos de banda son mayores.

En la figura 3.2 se pueden ver las respuestas al impulso para 8 filtros gammatone, así como también sus respuestas en frecuencia.

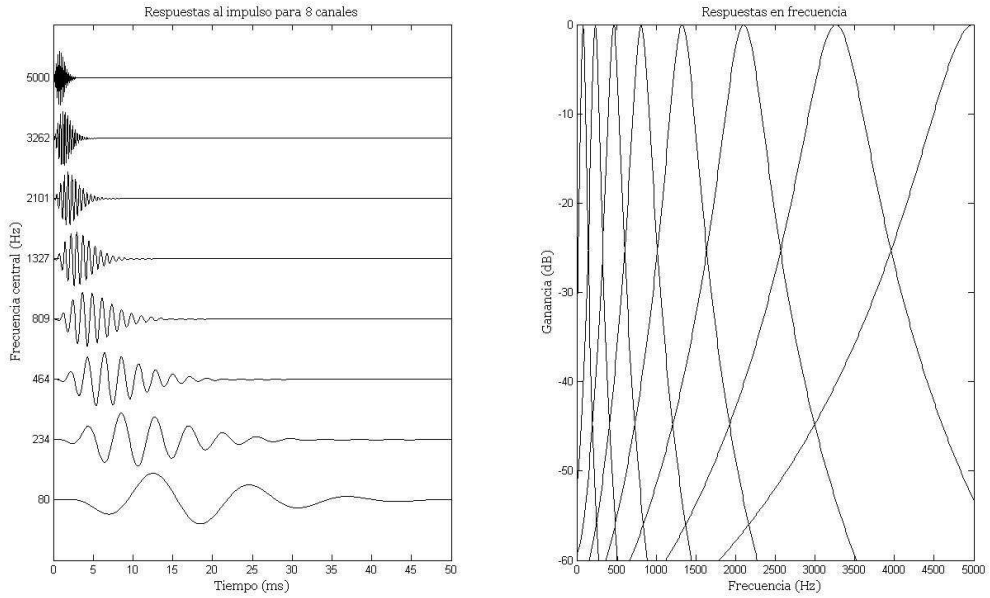

Figura 3.2: Filtros gammatone. En la figura de la izquierda se observa la respuesta al impulso para 8 filtros gammatone. En la figura de la derecha se puede ver la respuesta en frecuencia de estos filtros.

Las respuestas en frecuencia de la figura 3.2 muestran que los filtros son pasabanda, y que sus frecuencias centrales y sus anchos de banda aumentan logarítmicamente con la frecuencia. Otra propiedad relevante que se observa en la figura es el solapamiento de los espectros de los filtros.

3.2. Modelo de Meddis

Luego de que la señal pasa por el banco de filtros gammatone, se le aplica a cada canal el modelo de Meddis [18]. Éste intenta representar los fenómenos no lineales que ocurren en el oído humano, tales como: rectificación, saturación y bloqueo de fase. El modelo simula las propiedades de las células ciliadas,[4] las cuales se especializan en la transducción de movimiento a impulsos eléctricos.

El modelo surge entre los años 1986 y 1990 y consta de varios parámetros que se detallan en la tabla 3.1 [30].

[4]Por información de las células ciliadas ver:
http://www.rau.edu.uy/universidad/medicina/actas5/coclea/coclea.htm

Constante	Modelo	Valor	Unidades
A	Constante de permeabilidad	100	Unidades/seg
B	Constante de permeabilidad	6000	Unidades/seg
g	Tasa de liberación	2000	Unidades/seg
y	Tasa de *replenishment*	5,05	Unidades/seg
l	Tasa de pérdidas	2500	Unidades/seg
x	Tasa de reproceso	66,31	Unidades/seg
r	Tasa de recuperación	6580	Unidades/seg
m	Núm.máx. de paquetes transmitidos en *free pool*	1	Unidades/seg
h	Tasa de generación de impulso	50000	Picos/seg

Tabla 3.1: Parámetros del modelo de Meddis.

3.3. Implementación

En esta sección se presentan los detalles sobre la implementación del banco de filtros y del modelo de Meddis. Para llevarla a cabo se utilizó el toolbox de Matlab® desarrollado por uno de los autores del artículo [1], DeLiang Wang, cuyo código se encuentra disponible en [31].

3.3.1. Filtros gammatone

A continuación se describen las funciones del toolbox utilizadas en la implementación de los filtros gammatone.

gammatone.m: Es la función principal, implementa los filtros gammatone. Tiene como entradas: la señal de entrada a filtrar, el número de canales (128), el rango de frecuencias de interés (80 a 5000 Hz) y la frecuencia de muestreo (16 kHz).

Para llevar a cabo la implementación, se utiliza la propiedad de que la convolución en el dominio del tiempo es igual a la multiplicación en el dominio de la frecuencia. Primero se calcula la transformada rápida de Fourier para la señal de entrada y para la respuesta al impulso de cada filtro, a través de la función *fft (Fast Fourier Transformation)*. Luego se realiza la multiplicación de la transformada de la señal con cada una de las transformadas de la respuesta al impulso. Finalmente se antitransforma cada resultado a través de la función *ifft (Inverse Fast Fourier Transformation)*, y se obtienen las señales en el dominio del tiempo correspondientes a cada canal.

loudness.m: Simplemente carga parámetros y valores para realizar los filtros.

erb2hz.m: Es la implementación de la ecuación 3.2.

hz2erb.m: Es la función inversa de *erb2hz.m*.

Si se observa la ecuación 3.1, se puede ver que los filtros gammatone introducen un desfasaje entre muestras, distinto para cada canal. Si se quieren hacer comparaciones

entre canales, como es el caso del algoritmo de separación descrito en este documento, tal característica se vuelve un inconveniente. Se implementaron dos métodos para tratar de resolver este problema, uno es el de doble filtrado y el otro es el presentado por Hu-Wang en [16].

El método de doble filtrado consiste en primero filtrar la señal en un sentido, aquí se introduce un desfasaje que viene dado por el argumento de la transferencia del filtro, φ, luego se invierte todo el sentido de las muestras de la señal, y finalmente se pasa por el filtro nuevamente introduciendo un desfasaje de $-\varphi$. Por lo tanto el desfasaje total luego de este proceso es cero.[5] Si bien este método funciona correctamente para lograr un desfasaje nulo, tiene como desventaja que la ganancia del filtro se duplica, por lo que el filtrado es más severo en cada banda. Para realizar este método se implementó la función *gammatoneccf.m*.

Por otro lado, el método descrito en [16] consiste en retardar las muestras por una cantidad constante en cada canal, la cual está dada por: $\frac{l-1}{2\pi t 1{,}019 ERB} f_s$, donde l y ERB son los mismos que en la ecuación 3.1 y f_s es la frecuencia de muestreo (16 kHz). Los detalles exactos del cálculo pueden encontrarse en [32]. Si bien no es una corrección exacta, ya que los filtros tienen fase no lineal, se comprobó experimentalmente que se obtienen buenos resultados.

En la implementación se usó para la corrección de fase el método de [16] en todo el algoritmo, salvo en la etapa final de resíntesis. En esta etapa se utilizó el método de doble filtrado, ya que aquí no interesa tanto que las señales en cada canal se vean afectadas por el filtrado más severo, sino que las muestras estén con el desfasaje correcto.

En la figura 3.3 se puede ver para una entrada sinusoidal de 300 Hz la salida de los filtros sin ninguna corrección de fase, con la corrección de fase de Hu-Wang y con la corrección de fase de doble filtrado. Las frecuencias centrales están distribuidas entre 200 y 400 Hz.

[5] Matlab brinda la posibilidad de hacer este proceso mediante la función filtfilt.

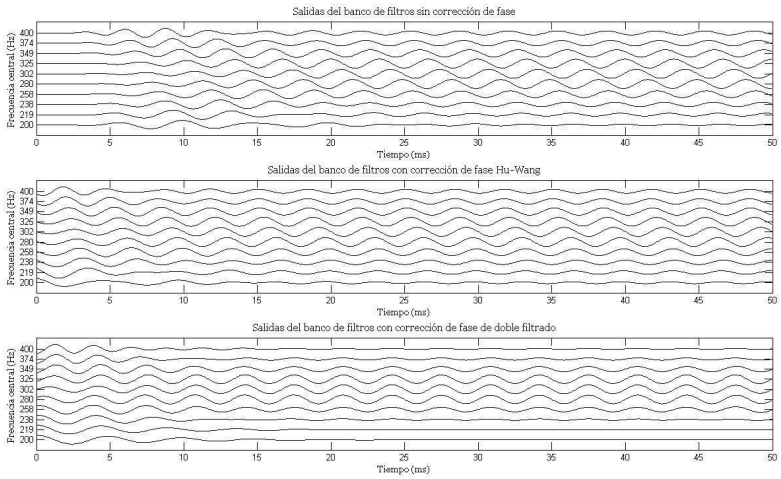

Figura 3.3: Comparación entre la salida original de los filtros y las correcciones de fase para una entrada sinusoidal de 300 Hz. En la figura superior se muestra la salida sin ninguna corrección de fase, en la figura del medio la salida con corrección de fase de Hu-Wang, mientras que en la figura inferior está la salida con corrección de fase de doble filtrado. Las frecuencias centrales varían entre 200 y 400 Hz.

En la figura superior se puede observar que efectivamente se introduce un retardo a la salida de cada canal, y que es distinto para cada uno de ellos. Se puede ver que los picos quedan desalineados entre los diferentes canales. Asimismo se aprecia que debido al solapamiento de las bandas de paso de los filtros, la entrada sinusoidal activa el canal de 300 Hz y los adyacentes.

Luego en las siguientes figuras se muestran las salidas de los filtros con los dos métodos de corrección de fase. Con la corrección de fase de Hu-Wang se puede ver que se logra una mejora entre la alineación vertical de los picos, aunque no es exacta. Por otro lado con la corrección de fase de doble filtrado se obtiene una corrección exacta, los picos quedan alineados verticalmente. Sin embargo se aprecia el efecto del filtrado más severo, ya que en los canales que están más alejados de los 300 Hz, la señal está más atenuada que en los casos anteriores.

3.3.2. Modelo de Meddis

Para llevar a cabo la implementación del modelo de Meddis, se utilizó la función ***meddis.m***, la cual pertenece al mismo toolbox utilizado con los filtros gammatone [31]. Es una función sencilla, que altera levemente a la señal de entrada utilizando las constantes mostradas en la tabla 3.1.

3.4. Mapa de unidades T-F

En cada canal de la salida del banco de filtros, se realiza un enventanado en el tiempo con ventanas de 16 ms (256 muestras a $f_S = 16$ kHz) y solapamientos entre ellas de 10 ms (160 muestras a $f_S = 16$ kHz). Ambos valores son sugeridos en el artículo [1]. Esta duración de la ventana permite tomar como hipótesis que la señal de voz es estacionaria en ese intervalo de tiempo.

Como resultado del filtrado pasabanda y del enventanamiento temporal, la señal de entrada queda descompuesta en tiempo y en frecuencia. Se forma así, un mapa de unidades en el cual cada unidad corresponde a un tiempo y una frecuencia determinada. Cada unidad de ese mapa se conoce como unidad T-F. La figura 3.4 muestra un esquema de este proceso.

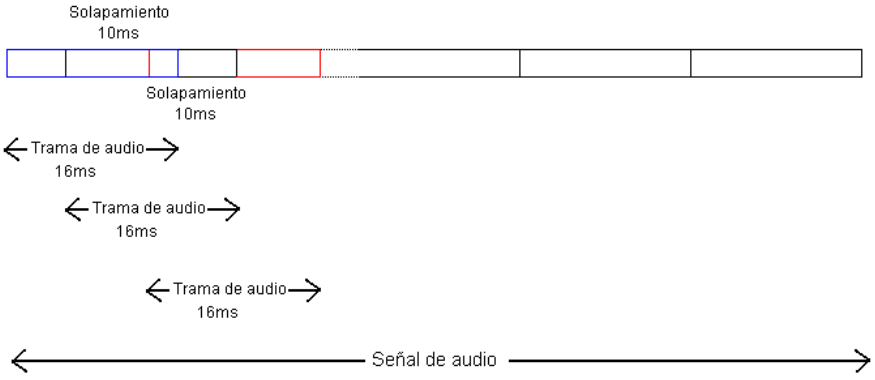

Figura 3.4: Esquema de enventanado en el tiempo.

De esta manera, el mapa de unidades T-F puede ser visto como una matriz de 128 filas y m columnas (siendo m la cantidad de ventanas que entran en la señal considerando el solapamiento). Cada elemento de la matriz es la parte de la señal que está dentro de la ventana.

3.5. Cocleagrama

Para realizar una representación gráfica de la señal en tiempo y en frecuencia generalmente se utiliza un espectrograma, el cual consiste en calcular el espectro de las frecuencias en consecutivas tramas temporales solapadas. Se obtiene por tanto una representación de las variaciones de la energía de la señal en una matriz de dos dimensiones, cuyos ejes vertical y horizontal son la frecuencia y el tiempo respectivamente. Cabe destacar que el eje de las frecuencias tiene una escala lineal.

Al utilizar el modelo auditivo para descomponer la señal en una representación de tiempo-frecuencia, se realiza un filtrado pasabanda a través del banco de filtros gammatone, seguido por una rectificación no lineal al utilizar el modelo de Meddis. El

cocleagrama consiste en graficar el logaritmo de la energía de cada unidad T-F. Aquí se obtiene una escala en frecuencia cuasi-logarítmica, y los anchos de banda de cada filtro dependen de la frecuencia. Asimismo el cocleagrama tiene mucho más resolución en bajas frecuencias que en altas, debido a la distribución de las frecuencias centrales de los distintos canales. El cocleagrama es por tanto una representación más apropiada cuando se utiliza el modelo auditivo, ya que se obtiene una resolución variable de los anchos de banda de cada filtro y de sus frecuencias centrales asociadas.

En la figura 3.5 se puede ver el espectrograma y el cocleagrama para una señal de voz hablada, y en la figura 3.6 lo mismo para una señal de voz cantada.

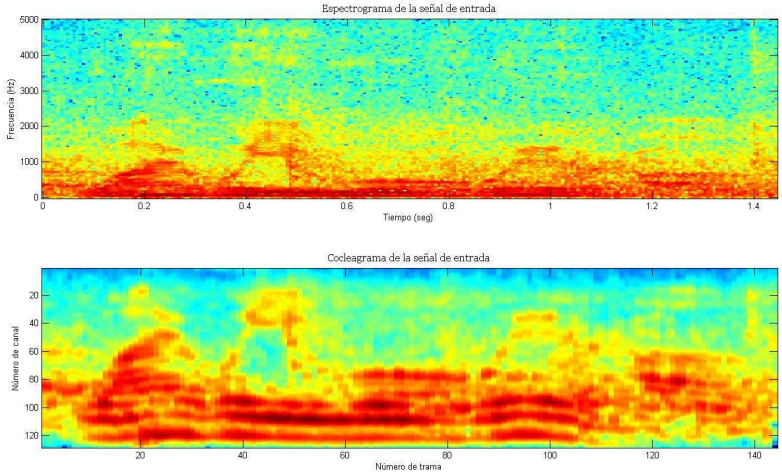

Figura 3.5: Comparación entre el espectrograma y el cocleagrama. En la figura superior se presenta el espectrograma de la señal, mientras que en la inferior está el cocleagrama. El ejemplo es para una señal de voz hablada más *cocktail party* (v3n3.wav).

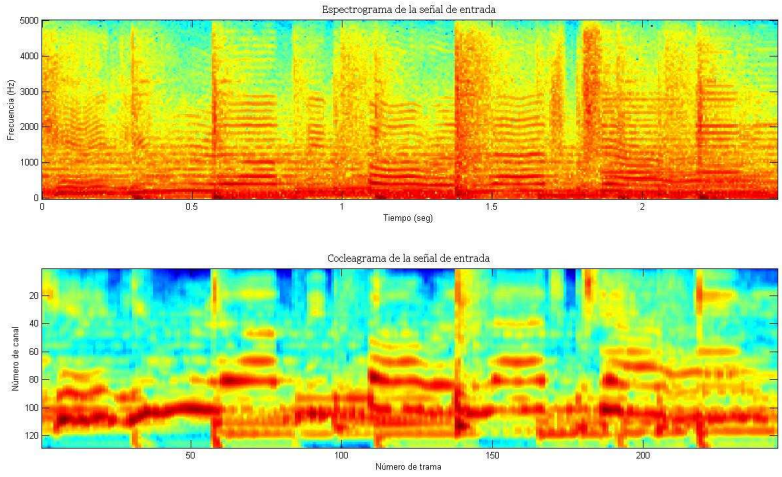

Figura 3.6: Comparación entre el espectrograma y el cocleagrama. En la figura superior se presenta el espectrograma de la señal, mientras que en la inferior está el cocleagrama. El ejemplo es para una señal de voz cantada más acompañamiento musical (todo61.wav).

En ambas figuras se puede apreciar que la representación en bajas frecuencias está expandida en el cocleagrama.

3.6. Conclusiones

Se presentó en este capítulo el modelo auditivo, constituido por los filtros gammatone y el modelo de Meddis. Se desarrolló su base teórica y se dieron detalles de la implementación. Asimismo de introdujeron los conceptos de unidades T-F y de cocleagrama.

En el siguiente capítulo se detalla sobre las características que se extraen de cada unidad T-F.

Capítulo 4

Extracción de Características

En el capítulo anterior se presentó el bloque del procesamiento auditivo. Se mostró que la señal de entrada es descompuesta en una matriz bidimensional que está formada por las unidades T-F. En las próximas etapas, las de segmentación y agrupamiento, se juntan regiones contiguas de unidades T-F para formar segmentos, y luego se agrupan los segmentos que son voz dominante para formar el *stream* final. Estas tareas se llevan a cabo tomando decisiones que se basan en distintas características de la señal.

En este capítulo se presentan tales características, las cuales son: la autocorrelación de las unidades T-F, la autocorrelación de las envolventes de las unidades T-F, la correlación cruzada entre dos unidades T-F de canales adyacentes, la correlación cruzada entre las envolventes de dos unidades T-F de canales adyacentes y la energía de las unidades T-F.

La segmentación se realiza teniendo en cuenta la continuidad de la señal en el tiempo y en frecuencia. Para medir la continuidad en frecuencia se utiliza la correlación cruzada entre dos unidades T-F de canales adyacentes. Debido al solapamiento de las bandas de paso de los filtros auditivos, un armónico activa varios canales adyacentes. Por tal motivo, si el valor de la correlación cruzada entre dos unidades T-F de canales adyacentes es cercano a uno, se puede establecer que las dos unidades son generadas por la misma fuente. En los canales de alta frecuencia,[1] los armónicos son generalmente no resueltos y por lo tanto las respuestas de los filtros están moduladas en amplitud y su envolvente fluctúa a la frecuencia fundamental f_0. En este caso si el valor de la correlación cruzada de las envolventes es cercano a uno, se puede establecer que las dos unidades T-F son generadas por la misma fuente.

En la etapa de agrupamiento, para los canales de baja frecuencia, se etiquetan los segmentos como voz dominante o acompañamiento dominante comparando la periocidad de cada unidad T-F con el período del *pitch* predominante en esa trama. Por otro lado, para los canales de alta frecuencia, se compara la periocidad de las envolventes de las unidades T-F con el período del *pitch* predominante en la respectiva trama. Para medir la periocidad de la señal en las unidades T-F se utiliza la función de autocorrelación. En los canales de baja frecuencia se calcula sobre las respuestas de los filtros en cada unidad

[1] Recordar de la sección 3.1, que se denomina canal de alta frecuencia a aquellos que tienen frecuencia central mayor a 1 kHz, mientras que los canales de baja frecuencia son los que tienen frecuencia menor.

T-F, mientras que en los canales de alta frecuencia se calcula sobre las envolventes de las respuestas de los filtros. Los segmentos que son voz dominante son agrupados luego al *stream* final, para tomar tal decisión se utiliza la energía de sus unidades T-F. La energía se halla utilizando la propiedad de que la autocorrelación de una señal evaluada en cero es su energía.

4.1. Correlograma

El correlograma consiste en un arreglo tridimensional, donde cada componente es la autocorrelación de las respuestas a la salida del modelo auditivo. Es decir, la autocorrelación de cada unidad T-F para determinado instante de tiempo. En la figura 4.1 se puede ver una representación esquemática del correlograma.

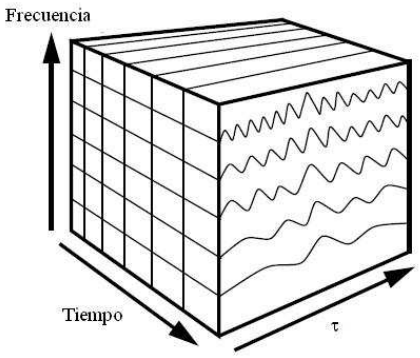

Figura 4.1: Representación esquemática del correlograma, donde el tiempo, la frecuencia y la autocorrelación son vistos en ejes ortogonales.

El correlograma es generalmente usado para obtener una representación de la periocidad de la señal, es útil tanto en las etapas de segmentación y agrupamiento, como en la etapa de detección de f_0. En la figura 4.2 se puede ver un ejemplo de un correlograma de una vocal con frecuencia fundamental de 100 Hz (período fundamental de 10 ms) en una trama específica.

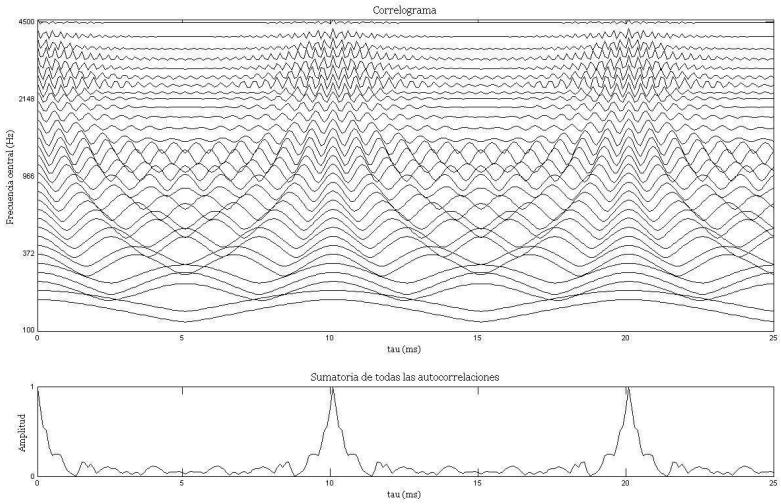

Figura 4.2: En la figura superior se muestra el correlograma de una vocal con frecuencia fundamental de 100 Hz para una trama específica, mientras que en la figura inferior está la suma de las autocorrelaciones en todos los canales.

En este ejemplo se pueden observar varios detalles relevantes. En primera instancia, se observa el efecto de los armónicos resueltos y no resueltos. Recordando que en los canales de baja frecuencia los ancho de banda de los canales son angostos, se puede ver como en los de más baja frecuencia que están cerca de 100 Hz, la autocorrelación muestra un pico en el período fundamental de 10 ms. Luego, en los canales con frecuencia central un poco mayor, se activa el segundo armónico de 200 Hz (5 ms) y la autocorrelación pasa a tener picos en 5 ms (también aparecen picos en 10, 15, 20 ms debido a los múltiplos del período fundamental).

A medida que se va aumentando de canal y los armónicos siguen siendo resueltos, los canales responden a un armónico solamente. Luego los anchos de banda de los canales son cada vez más grandes y ya comienzan a existir interacciones entre más de un armónico, lo que produce un efecto de batido, es decir, la respuesta de los filtros está modulada en amplitud y su envolvente fluctúa a f_0. Este efecto se aprecia claramente en los primeros canales de la figura 4.2. Cabe destacar que en estos canales la autocorrelación también presenta un pico en 10 ms.

Debido al hecho de que en los canales de baja frecuencia todas las autocorrelaciones presentan picos en el período fundamental, y que en alta frecuencia las envolventes fluctúan a f_0, es que el correlograma es muy utilizado también en los algoritmos de detección de *pitch*. En la figura 4.2 debajo del correlograma se puede observar una sumatoria de todas las autocorrelaciones por sobre todos los canales. Se aprecia que presenta un pico predominante en 10 ms, lo que muestra su utilidad para detectar el *pitch*.

Otro aspecto a destacar de la figura 4.2, es que se puede ver cómo efectivamente debido al solapamiento de las bandas de paso de canales adyacentes, un armónico activa un cierto número de canales consecutivos.

4.1.1. Función de correlación

La función de correlación entre dos secuencias independientes $\{s(n)\}$ y $\{t(n)\}$ se define como:

$$c(l) = \frac{1}{N} \sum_{n=0}^{N-1} s(n)t(n-l)$$

Esta función es una medida de la similitud estadística entre dos señales. Es decir, como su nombre lo indica, da una idea de que tan correlacionadas se encuentran ambas.

4.1.2. Función de autocorrelación

Se define como la correlación de la señal consigo misma:

$$A(l) = \frac{1}{N} \sum_{n=0}^{N-1} s(n)s(n+l)$$

La función de autocorrelación (así como también su versión normalizada) sirve para obtener una estimación de la periodicidad de la señal, por lo que es usada por muchos algoritmos de detección de *pitch* [33].

El valor de $A(0)$ es siempre el valor máximo de la función, ya que indica una perfecta correspondencia entre la señal consigo misma. Asimismo este valor es la energía media de la señal [34]. Para los siguiente instantes de desplazamientos temporales, un valor grande de $A(l)$ indica una similitud entre la secuencia y ella misma desplazada un valor l, lo que muestra una periocidad de la señal. En caso de que la señal sea aleatoria, la función de autocorrelación decae a cero luego del valor inicial $A(0)$.

Las señales de audio se pueden considerar periódicas en períodos cortos de tiempo (cuasi-periódicas[2]), por lo que resulta necesario separar la señal en pequeñas tramas. Debido a esto, el algoritmo implementado utiliza ventanas de 16 ms, con un solapamiento de 10 ms entre dos ventanas consecutivas.

En la figura 4.3 se muestra un fragmento de un sonido sonoro de la palabra “hola”, mientras que en la figura 4.4 se puede ver la autocorrelación normalizada de una trama de la señal, la cual está ubicada en $t = 1{,}14$ segundos con una ventana de 16 ms.

[2]Las señales cuasi-periódicas son señales periódicas en un intervalo de tiempo, independientemente del comportamiento fuera del mismo[1].

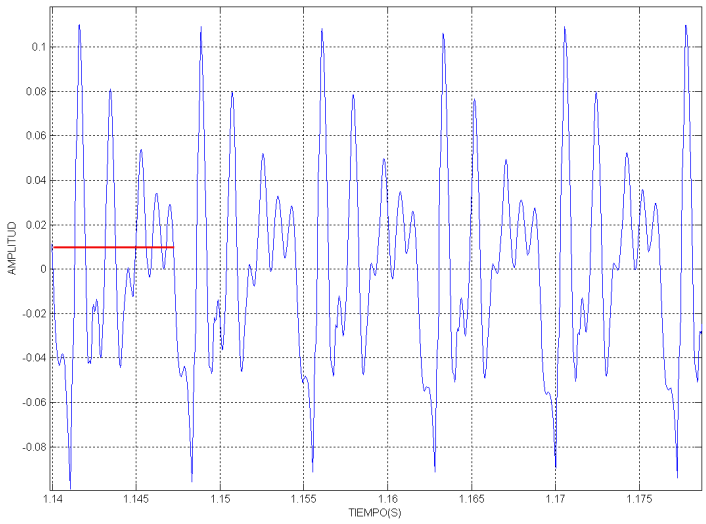

Figura 4.3: Fragmento de un sonido sonoro perteneciente a la palabra "hola". El intervalo de tiempo es de $t = 1,14$ a $t = 1,18$ segundos. En rojo se marca el período de la señal, el cual tiene un valor aproximado de $t = 0{,}007$ segundos.

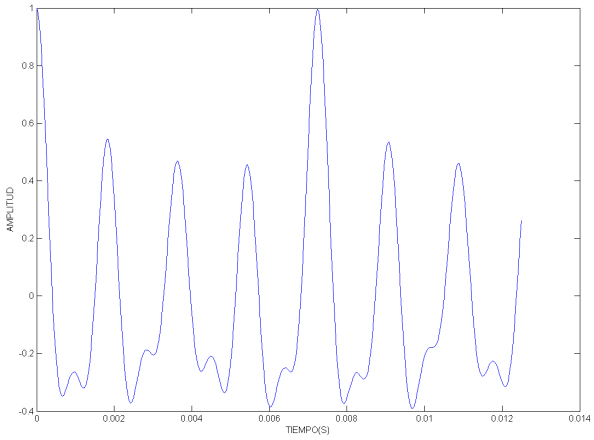

Figura 4.4: Autocorrelación normalizada de la señal, para una ventana de 16 ms ubicada en $t = 1,14$ segundos.

En la figura 4.4 se puede observar que para un retardo τ_0 de 0,007 segundos aproximadamente, la autocorrelación normalizada es casi 1 (el valor real es de 0,992), por lo se puede considerar que la señal tiene un período aproximado de τ_0 segundos. De hecho, en la figura 4.3, se puede ver que efectivamente la señal es periódica con período aproximado de $\tau = 0{,}007$ segundos.

4.1.3. Implementación

Para computar el correlograma, se calcula la autocorrelación en todas las unidades T-F. Para esto, en el algoritmo implementado se utiliza la versión normalizada de la autocorrelación. Sea la unidad T-F $u_{c,m}$ para el canal c y la trama m, el correlograma normalizado a la salida del bloque auditivo viene dado por:

$$A_H(c,m,\tau) = \frac{\sum_{n=0}^{N-1} h(c,mT+n)h(c,mT+n+\tau)}{\sqrt{\sum_{n=0}^{N-1} h^2(c,mT+n)}\sqrt{\sum_{n=0}^{N-1} h^2(c,mT+n+\tau)}} \tag{4.1}$$

Donde h es la salida del modelo auditivo, m es el índice de la trama y c es el índice del canal. En la implementación del algoritmo se utiliza una frecuencia de muestreo de 16 kHz, por lo que el valor de N es de 256 muestras ya que se utilizan ventanas de 16 ms, de esta forma $N/fs = 16$ ms. Del mismo modo $T = 160$ al utilizar solapamientos entre ventanas de 10 ms. El correlograma es calculado para valores de τ comprendidos entre 32 y 200, debido a que corresponden al rango definido de variación del *pitch* de la señal: entre 80 y 500 Hz. El valor de $\tau = 200$ equivale a 12,5 ms u 80 Hz.

Los cálculos de la autocorrelación implican un elevado costo computacional, debido a la cantidad de cuentas que se realizan.

4.2. Correlación cruzada entre canales

La correlación cruzada entre dos canales adyacentes mide la similitud entre las autocorrelaciones de ambos. Es una medida de que tan similares son los patrones de periocidad de sus respuestas. Como un armónico activa varios canales consecutivos, si dos canales adyacentes presentan alta correlación cruzada, se puede suponer que responden a un mismo componente acústico.

Para cada unidad T-F $u_{c,m}$, su correlación de canal cruzada con $u_{c+1,m}$ viene dada por:

$$C_H(c,m) = \sum_{\tau=0}^{L-1} \tilde{A}(c,m)\tilde{A}(c+1,m) \tag{4.2}$$

Donde c indica el canal, m la trama y $\tilde{A}$ es la autocorrelación con media cero y varianza uno de la señal. Se utiliza la autocorrelación de esta forma para asegurarse de que $C_H(c,m)$ sea sensible sólo a los patrones de periocidad, y no a las variaciones de la media que introducen los filtros auditivos.

4.3. Extracción de la envolvente

En los canales de alta frecuencia donde en general los armónicos son no resueltos, interesa obtener la envolvente de las respuestas. Esto se puede realizar utilizando varios métodos que se presentan a continuación.

4.3.1. Rectificado de media onda

Este método consiste en realizar un rectificado media onda de la señal, seguido de un filtrado pasabanda en el rango de variación definido del *pitch* ($80 - 500$ Hz).

4.3.2. Transformada de Hilbert

La extracción de la envolvente de la señal a partir de la transformada de Hilbert es un método muy usado. La transformada de Hilbert de una función $f(x)$ se define como:

$$F(t) = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{f(x)}{t-x} dx \tag{4.3}$$

La ecuación 4.3 puede verse como una convolución:

$$F(t) = \frac{1}{\pi t} * f(t) \tag{4.4}$$

Si se toma la transformada de Fourier a ambos lados de la ecuación 4.4, utilizando la propiedad de que la convolución en el tiempo es igual a la multiplicación en la frecuencia, dicha ecuación puede ser evaluada como el producto de $-i \times sgn(x)$ con la transformada de $f(x)$. Por lo tanto, la transformada de Hilbert es equivalente a pasar la señal por un filtro, el cual hace un corrimiento de la fase de la señal de entrada de $-\frac{\pi}{2}$, para todas las frecuencias.

Una señal analítica compleja $Y(t)$ puede ser construida a partir de la señal real $f(t)$ de la siguiente manera:

$$Y(t) = f(t) + iF(t) = A(t)e^{i\phi(t)} \tag{4.5}$$

Finalmente una vez que se tienen la señal y su transformada de Hilbert, se calcula la envolvente de la señal:

$$A(t) = \sqrt{f(t)^2 + F(t)^2} \tag{4.6}$$

En la parte final de este detector de envolvente se realiza un filtrado pasabajo, el cual se implementó con un filtro Butterworth de orden 3 y f_c= 800 Hz. El objetivo es suavizar la envolvente y quedarse con las componentes de interés.[3]

4.3.3. Operador de energía de Teager

El operador de energía de Teager (OET) fue propuesto por Kaiser [35], es definido en tiempo discreto como:

$$\Psi_d[x(n)] = x^2(n) - x(n+1)x(n-1) \tag{4.7}$$

Cuando el OET es aplicado a una señal oscilatoria, el resultado es igual a la energía de la fuente que produce la oscilación. Si se aplica a una señal $x(t) = A\cos(w_0 t + \theta)$,

[3]Recordar que el rango del *pitch* es hasta 500 Hz y las envolventes de las respuestas en alta frecuencia fluctúan a f_0.

el resultado es proporcional al cuadrado del producto de la amplitud y la frecuencia de oscilación:

$$\Psi_d[A\cos(w_0 t + \theta)] = A^2 w_0^2 \tag{4.8}$$

En [36] se presentan algoritmos para hallar la amplitud y la frecuencia de la señal, cada una por separado.

Al igual que en el anterior método, se finaliza con un filtro pasabajo con $f_c = 800$ Hz.

4.3.4. Resultados

Para el cálculo de la envolvente se decidió utilizar el método de la transformada de Hilbert, ya que es el que presenta mejores resultados.

En la figura 4.5 se puede ver la respuesta de un canal de alta frecuencia de los filtros auditivos, junto con la envolvente hallada con el método de Hilbert. Se puede observar que se obtiene un buen resultado.

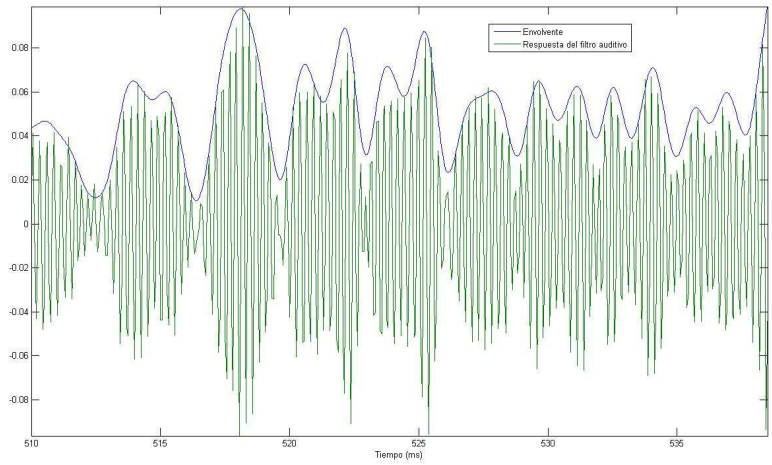

Figura 4.5: Respuesta de un canal de alta frecuencia y su envolvente de amplitud calculada mediante la transformada de Hilbert.

4.4. Resultados

Como se mencionó en este capítulo, las características presentadas se calculan de diferente forma según se esté en alta o baja frecuencia. En baja frecuencia se calculan sobre la unidad T-F y en el caso de alta frecuencia se computan sobre las envolventes de las unidades T-F. Las siguientes figuras ilustran cómo cambia el correlograma al utilizar las envolventes para los canales de alta frecuencia.

La figura 4.6 muestra el correlograma para una trama de audio. En el mismo, no se realiza la discriminación entre canales de baja y alta frecuencia. Se calcula las autocorrelaciones sobre las unidades T-F para todos los canales.

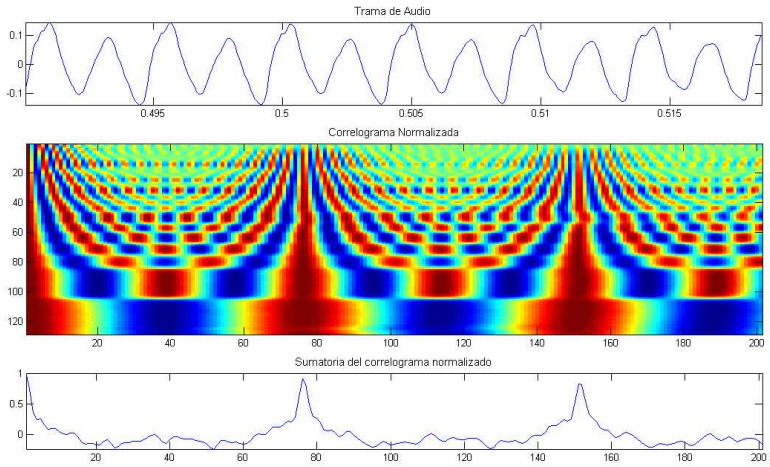

Figura 4.6: Correlograma computado sobre todas las unidades T-F, tanto en los canales de baja frecuencia como en los de alta.

En la parte superior de la figura 4.6 se observa la trama de audio en estudio. En la parte central, el correlograma computado sobre todas las unidades T-F. En bajas frecuencias (canales inferiores sobre el eje vertical), se observa en rojo, los picos de la autocorrelación. Se aprecian los armónicos f_0, $2f_0$ y $3f_0$, por ejemplo. En los canales de alta frecuencia se observa el efecto del batido. La figura inferior muestra el resultado de sumar todos los canales del correlograma. En ella se observan los picos de la autocorrelación, los cuales indican el período de la señal. Cabe destacar que también aparecen picos espurios, esto es no deseable ya que puede inducir a errores. Para evitar estos picos espurios es justamente que se utiliza la envolvente.

La figura 4.7 muestra, para la misma trama de audio, el correlograma utilizando la envolvente para los canales de alta frecuencia.

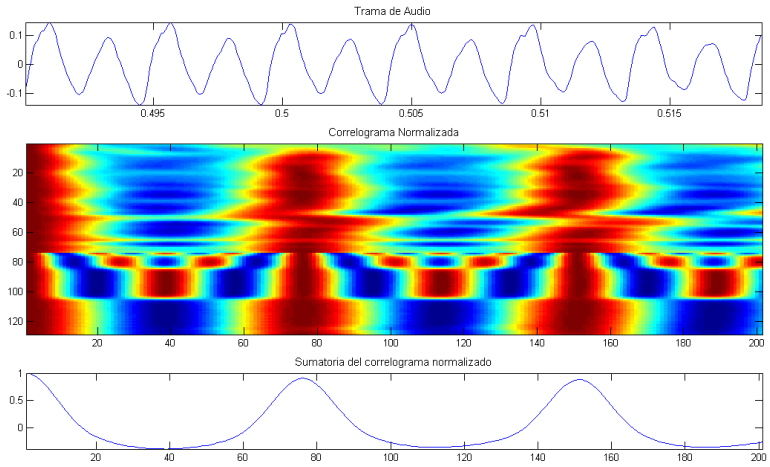

Figura 4.7: Correlograma computado sobre todas las unidades T-F en baja frecuencia y sobre las envolventes de las unidades en alta frecuencia.

La figura anterior muestra en la parte superior nuevamente la trama de audio en estudio. En la parte central se tiene el correlograma. En los canales de baja frecuencia el correlograma es idéntico al de la figura 4.6, nuevamente se aprecian los armónicos f_0, $2f_0$ y $3f_0$. En la parte de alta frecuencia se realiza el correlograma sobre las envolventes de las unidades T-F. Se observa claramente cómo el correlograma en alta frecuencia presenta máximos (zonas rojas) sólo en la frecuencia fundamental f_0. De este modo, se justifica lo que se comentó inicialmente, que la envolvente en alta frecuencia fluctúa entorno a la frecuencia fundamental f_0. En la parte inferior de la figura 4.7 se calcula nuevamente la suma del correlograma. Los máximos que se observan indican el período de la señal en estudio. Se puede ver, que si bien estos picos son más suaves que los que se muestran en la figura 4.6, no aparecen los picos espurios. Por lo tanto, al utilizar un detector de picos (o máximos) se obtendrían mejores resultados al haber menos probabilidad de cometer error. La figura 4.7 muestra claramente el efecto que tiene la discriminación entre canales de baja y alta frecuencia y justifica el trabajar de este modo.

4.5. Conclusiones

En este capítulo se han presentado las distintas características que se extraen de cada unidad del mapa T-F. Se mostraron ejemplos y se explicó la utilidad de tales características. Los conceptos y las funciones que se mencionaron en este capítulo se utilizan en las siguientes etapas del algoritmo de separación.

Capítulo 5

Detección del *pitch* predominante

5.1. Introducción

Este capítulo describe el método de detección de *pitch* o frecuencia fundamental f_0. Está basado en el método propuesto en [25], el cual permite detectar múltiples *pitches* (de voz cantada y otras fuentes armónicas provenientes de instrumentos musicales, por ejemplo). Esta etapa es de vital importancia para obtener buenos resultados al final del sistema de separación de voz cantada.

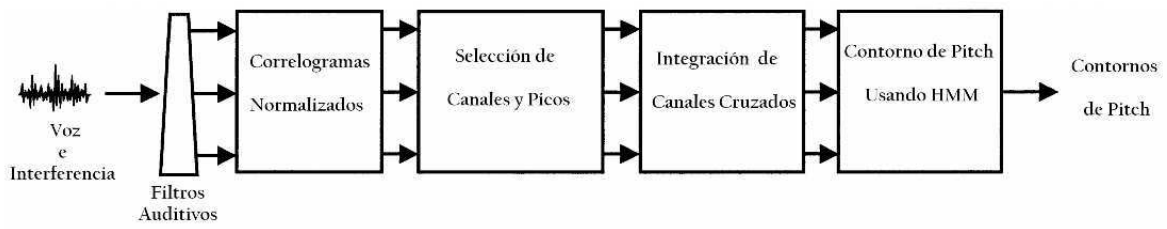

Figura 5.1: Representación esquemática del método para la obtención del *pitch*.

A diferencia de otros algoritmos de detección de *pitch*, como son por ejemplo los provistos por los programas *WaveSurfer* [37] o *Praat* [38], se calcula el *pitch* para el caso polifónico, es decir, cuando está presente más de una fuente armónica. Una hipótesis fuerte que se realiza en esta etapa es que en una trama donde esté presente la voz cantada y el acompañamiento musical, el *pitch* predominante será el de la voz cantada.

Los dos primeros bloques del algoritmo de detección de *pitch* son similares a los explicados en los capítulos 3 y 4. En este caso se usan los correlogramas para obtener los picos de la autocorrelación, los cuales se utilizan para calcular la frecuencia fundamental f_0.

El bloque de selección de canales y picos se encarga primero de seleccionar distintos canales, con el fin de eliminar aquellos que posean interferencia y así de este modo, evitar cometer errores en el cálculo de la frecuencia fundamental. Luego para cada uno de los canales resultantes, se seleccionan diferentes picos y se guardan los valores de los retardos de los picos en un conjunto de picos.

Para modelar el proceso de generación de *pitch* se utiliza un modelo oculto de Markov (HMM), el cual se ilustra en la figura 5.2. En el apéndice E se encuentra una breve introducción al HMM.

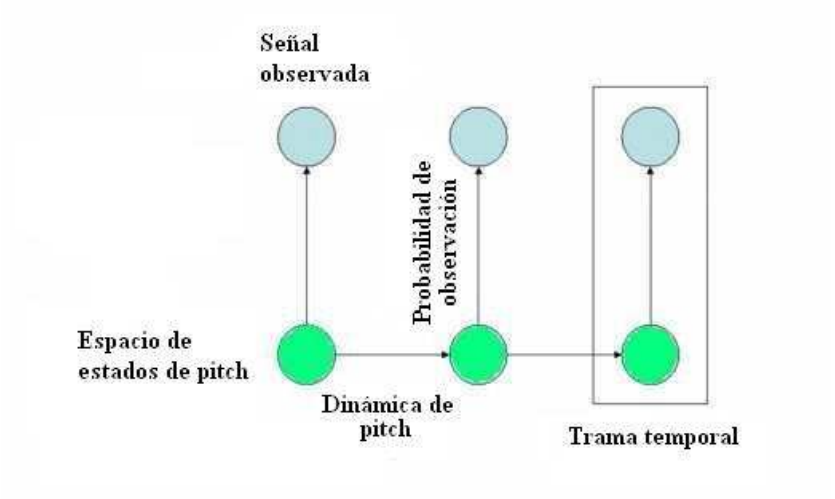

Figura 5.2: Diagrama esquemático del HMM. En verde se muestran los nodos ocultos, los cuales representan posibles estados de *pitch* en cada trama. En celeste están los nodos observados, los cuales representan el conjunto de picos seleccionados en cada trama.

El espacio de estados de *pitch* Ω es la unión de tres subespacios:

$$\Omega = \Omega_0 \cup \Omega_1 \cup \Omega_2 \tag{5.1}$$

Donde Ω_0, Ω_1 y Ω_2 son subespacios de cero, una y dos dimensiones respectivamente. Cada uno de ellos representa una colección de hipótesis de *pitch* con cero, uno y dos *pitches* respectivamente. Un estado en el espacio de estados es representado por el par:

$$x = (y, Y) \tag{5.2}$$

Donde $y \in R^Y$ e $Y \in 0, 1, 2$ es el índice del subespacio.

En cada trama temporal, un nodo oculto del HMM indica un posible estado de *pitch*, mientras que el nodo observado indica el conjunto de picos seleccionados. En el bloque de integración de canales se calculan las probabilidades de observación, que son las probabilidades de observar un conjunto de picos (nodo observado) estando en un estado en particular (nodo oculto). Por último, en el bloque final se calculan las probabilidades asociadas a la dinámica de *pitch* (las transiciones entre distintos estados), y finalmente se halla la secuencia de estados más probable, o lo que es lo mismo, el valor del *pitch* predominante en cada trama.

5.2. Concepto de *pitch*

El concepto de *pitch* se refiere básicamente a la altura tonal (una de las propiedades de la música), como por ejemplo la de una nota musical o la voz humana. El uso del

término *pitch* a veces puede inducir a confusión, dado que es empleado para referirse tanto a la frecuencia fundamental de un sonido en particular, como a una cualidad de la sensación auditiva. En general, cuando se hace procesamiento de voz, dicho término hace referencia a la frecuencia fundamental de la oscilación glótica (vibración de las cuerdas vocales). Esta última es la definición que se adoptó en este documento, por lo tanto se habla indistintamente de *pitch* o de frecuencia fundamental f_0.

Otras definiciones que pueden encontrase son:

- En Psicoacústica el término *pitch* es empleado para referirse a una cualidad (subjetiva) asociada a la percepción de los sonidos [39].
- Se entiende como *pitch* a la altura de una nota musical, o mejor dicho, la frecuencia fundamental de un sonido [40].
- El término *pitch* es utilizado de manera intercambiable con frecuencia fundamental. La Psicoacústica utiliza el término *pitch* para referirse a la frecuencia fundamental percibida de un sonido, no importando si esta frecuencia se encuentra actualmente presente en el espectro del sonido [41].

5.3. Filtros auditivos

El primer bloque del cálculo de f_0 consiste en el banco de filtros auditivos, los cuales ya se presentaron en el capítulo 3. En la salida de cada canal, la señal se divide en tramas de 32 ms, con un solapamiento de 10 ms, similar al enventanado que se realizó en la primera etapa del algoritmo de separación de voz cantada. Cabe destacar que en esta ocasión se usa una ventana de mayor duración para generar las tramas (la anteriormente utilizada era de 16 ms).

5.4. Correlogramas normalizados

En este segundo bloque se computan similares características a las presentadas en el capítulo 4, con la diferencia de que aquí los canales de alta frecuencia son considerados para frecuencias mayores a 800 Hz. En este caso las características extraídas en cada unidad T-F son el correlograma normalizado y el correlograma normalizado de las envolventes. Cabe destacar que aumentar la ventana a 32 ms produce que la autocorrelación sea computacionalmente más costosa que para 16 ms.

5.5. Selección de canales y picos

Una vez calculados los correlogramas, recordando que los picos de la autocorrelación sirven para obtener la periodicidad de la señal,[1] se realiza una selección de canales en

[1]Hay que considerar también que la presencia de música, puede llevar a resultados erróneos, por ejemplo, el acompañamiento de percusión, contiene mucha energía en canales de baja frecuencia, haciendo que la consideración anterior sea a veces poco fiable.

función de la amplitud de los picos de la autocorrelación. La selección de canales y picos es diferente según se esté en baja o alta frecuencia.

Un canal de baja frecuencia es seleccionado si el máximo valor del correlograma normalizado dentro del rango de *pitch* de interés, $f_0 \in [80, 500]\ Hz$, excede un umbral $\mu = 0,945$. Para cada canal de baja frecuencia seleccionado, los retardos τ de todos sus picos se guardan en una matriz Φ. Esta matriz se utiliza en la etapa del cálculo de las probabilidades de observación.

Por otra parte, para los canales de alta frecuencia, no se descarta ningún canal, y se incluye en el conjunto Φ al primer retardo τ distinto de 0 donde se produzca un pico de la autocorrelación de la envolvente de la salida (que esté en el rango de *pitch* de interés). Así de este modo puede ajustarse al fenómeno de batido que se produce en alta frecuencia. Como se mencionó anteriormente, las salidas de los canales de alta frecuencia responden a múltiples armónicos, y sus envolventes fluctúan a la frecuencia fundamental. El no realizar ningún tipo de selección en los canales de alta frecuencia, puede llevar a seleccionar picos cuyos retardos no corresponden al período fundamental de la voz cantada. De todos modos en [1] se comprueba experimentalmente que proceder de este modo sigue siendo un buen indicador para la obtención del *pitch*. Por otro lado, debido al hecho de quedarse con todos los canales de alta frecuencia, se tienen más canales disponibles, lo que es importante para distinguir diferentes fuentes armónicas.

El método de selección de canales y picos descrito anteriormente permite lidiar con el problema introducido por los instrumentos de percusión, conocido como *beat phenomenon*, el cual causa confusión en los algoritmos de detección de *pitch*.

5.6. Integración estadística entre canales

En esta sección de derivan las ecuaciones de las probabilidades de observación del modelo oculto de Markov. Estas ecuaciones son la probabilidad condicional $p(\Phi|x)$ de observar el conjunto de picos Φ dado un estado de *pitch* x. El proceso se realiza en dos etapas. Primero se calcula la contribución de cada canal a una hipótesis de *pitch*, y luego se combina la contribución de todos los canales en un solo resultado.

El alineamiento de picos en el correlograma normalizado señala un período de *pitch*. La probabilidad de que un canal respalde una hipótesis de *pitch*, se deriva estudiando la diferencia entre el valor verdadero de *pitch* y el tiempo de retardo del pico más cercano del correlograma normalizado en ese canal. Sea el canal c, se define la distancia δ como:

$$\delta = l - d \tag{5.3}$$

Donde d es el valor verdadero de *pitch* y l es el valor del retardo del pico más cercano. Si la voz cantada es dominante en un canal, la distancia δ tiende a ser pequeña. La estadística de δ puede ser extraída a partir de muestras que contengan sólo voz cantada. En el artículo [26] se halla que esta estadística puede ser descrita por una distribución Laplaciana centrada en cero y que decae exponencialmente con $|\delta|$:

$$L(\delta; \lambda_c) = \frac{1}{2\lambda_c} \exp(-\frac{|\delta|}{\lambda_c}) \tag{5.4}$$

Donde el parámetro λ_c indica la dispersión de la distribución Laplaciana. La distribución de probabilidad de δ en un canal c es definida como:

$$p_c(\delta) = (1-q)L(\delta;\lambda_c) + qU(\delta;\eta_c) \tag{5.5}$$

Donde $U(\delta;\eta_c)$ es la distribución de probabilidad uniforme y modela el ruido de fondo. El parámetro η_c indica el rango de la distribución. En los canales de baja frecuencia $\eta_c = (\frac{-f_s}{2f_c}, \frac{f_s}{2f_c})$, donde f_s es la frecuencia de muestreo y f_c es la frecuencia central del canal c. Mientras que en los canales de alta frecuencia η_c es el posible rango de *pitch*. Por otro lado, q es un parámetro de partición de probabilidad ($0<q<1$).

El parámetro λ_c decrece a medida que la frecuencia central de los canales aumenta. Experimentalmente se puede aproximar de forma lineal, $\lambda_c = a_0 + a_1 c$, y con métodos de maxima verosimilitud aplicados a fragmentos de voz cantada sola, es posible estimar los parámetros a_0, a_1 y q. La estimación de estos parámetros se realiza para baja y alta frecuencia en forma separada. Los valores de estos parámetros se obtuvieron directamente del artículo [26] y se presentan en la tabla 5.1.

	a_0	a_1	q
1 *pitch* (BF)	1.21	-0.011	0.016
1 *pitch* (AF)	2.60	-0.008	0.063
2 *pitches* (BF)	1.56	-0.018	0.016
2 *pitches* (AF)	3.58	-0.016	0.108

Tabla 5.1: Valores de los parámetros del modelo. Donde BF es baja frecuencia y AF es alta frecuencia.

Una vez que se tiene la distribución de δ, las probabilidades condicionales en cada canal para las hipótesis de 1 y 2 *pitch* pueden ser formuladas. Se considera primero la hipótesis de 1 *pitch* d, la probabilidad condicional para un canal c se define como:

$$p(\Phi_c|x_1) = \begin{cases} p_c(\delta) & \text{si } c \text{ es un canal seleccionado} \\ q_1(c)U(0;\eta_c) & \text{de otra manera} \end{cases} \tag{5.6}$$

Donde Φ_c es el conjunto de picos seleccionados en el canal c, δ es la diferencia entre d y el retardo del pico más cercano en Φ_c, $x_1 = (d,1) \in \Omega_1$ es el estado de *pitch* y $q_1(c)$ es el factor de partición en el canal c para el caso de 1 *pitch*.

La probabilidad condicional de cada canal es combinada con la de los otros canales, de forma de obtener la probabilidad condicional total en cada trama. Debido a que la voz cantada tiene un ancho de banda amplio y que las bandas de paso de los canales del banco de filtros tienen solapamiento, no se puede considerar la independencia estadística entre los distintos canales. En los artículos [42] y [26] se propone combinar la información entre los canales de la siguiente manera:

$$p(\Phi|x_1) = k_1 \sqrt[b]{\prod_{c=1}^{C} p(\Phi_c|x_1)} \tag{5.7}$$

Donde Φ es la matriz que contiene los conjuntos de picos en todos los canales, C es el número de canales (128 en el algoritmo implementado), el parámetro $b = 6$ es el factor de suavización para compensar la dependencia entre canales y k_1 es un factor de normalización.

Para el caso de la hipótesis de 2 *pitches* d_1 y d_2 correspondientes a diferentes fuentes armónicas, la probabilidad condicional para un canal c se define como:

$$p(\Phi_c|(d_1,d_2) = \begin{cases} q_2(c)U(0;\eta_c) & \text{si } c \text{ es un canal no seleccionado} \\ p_c(\Phi_c|d_1) & \text{si el canal } c \text{ pertenece a } d_1 \\ max(p_c(\Phi_c|d_1), p_c(\Phi_c|d_2)) & \text{de otra manera} \end{cases} \tag{5.8}$$

Donde $q_2(c)$ es el factor de partición para el caso de 2 *pitches*. El canal c pertenece a d_1 si la distancia entre d_1 y el retardo del pico más cercano en ese canal es menor a $5\lambda_c$. Esta condición testea si el canal c es dominado por la fuente d_1. La probabilidad combinada es definida como:

$$p_2(\Phi|(d_1,d_2)) = \sqrt[b]{\prod_{c=1}^{C} p(\Phi_c|(d_1,d_2)} \tag{5.9}$$

Finalmente la probabilidad condicional para una trama temporal es la mayor ya sea asumiendo que d_1 o d_2 es la fuente predominante:

$$p(\Phi|x_2) = k_2 max(p_2(\Phi|(d_1,d_2)), p_2(\Phi|(d_2,d_1))) \tag{5.10}$$

Donde $x_2 = ((d_1,d_2),2) \in \Omega_2$ y k_2 es un factor de normalización.

De esta manera el algoritmo estima hasta dos *pitches* simultáneamente, y se queda con el *pitch* predominante en cada trama, ya que se asume como hipótesis que la voz cantada es predominante cuando está presente.

Para el caso de 0 *pitch* se fija su probabilidad condicional:

$$p(\Phi|x_0) = k_0 \tag{5.11}$$

Donde $x_0 \in \Omega_0$.

5.7. Obtención del *pitch* predominante utilizando HMM

En esta sección se comenta sobre las transiciones entre diferentes nodos ocultos vecinos del modelo de Markov, o sea, entre diferentes estados del estado de *pitch*, lo cual es llamado dinámica de *pitch*. Ésta tiene dos aspectos: la probabilidad de transición entre diferentes configuraciones de *pitch* en el mismo subespacio, y la probabilidad de salto entre diferentes subespacios.

El comportamiento de transición dentro del subespacio Ω_1 es descrito por una distribución Laplaciana, la cual se obtuvo directamente del artículo [26]:

$$p(\Delta) = \frac{1}{2\lambda} \exp(-\frac{|\Delta - \mu|}{\lambda}) \tag{5.12}$$

Donde Δ son los cambios del período de *pitch* en dos tramas consecutivas, $\mu = 0$ y $\lambda = 0{,}7$ son parámetros de la distribución y fueron tomados directamente del artículo [1].

El comportamiento de transición dentro de Ω_2 es descrito por $p(\Delta_1)p(\Delta_2)$, asumiendo que los dos *pitches* son independientes. Donde Δ_i son los cambios del período de *pitch* del i *pitch* en dos tramas consecutivas.

Por otro lado, las probabilidades de salto entre diferentes subespacios también se determinan utilizando datos de entrenamiento. Éstas fueron tomadas directamente del artículo [1] y se muestran en la tabla 5.2.

	$\rightarrow\Omega_0$	$\rightarrow\Omega_1$	$\rightarrow\Omega_2$
Ω_0	0.2875	0.7125	0.0000
Ω_1	0.0930	0.7920	0.1150
Ω_1	0.0000	0.0556	0.9444

Tabla 5.2: Probabilidad de salto entre diferentes subespacios del estado de *pitch*.

Una vez que se calcularon todas las probabilidades de observación y de transición entre diferentes estados, se utiliza el algoritmo de Viterbi para obtener la secuencia de estados más probable [43]. Para reducir el costo computacional se realizan una serie de procedimientos en el algoritmo implementado. Como los períodos de *pitch* son continuos, la diferencia entre dos períodos en tramas consecutivas se reduce a un valor de 30. Debido a la gran cantidad de secuencias de estados que se forman, se limita el número de ellas a un valor de 100, o sea que solamente un cierto número de secuencias de estados más probables son mantenidos y considerados en la siguiente trama.

La secuencia de estados que se obtiene es una mezcla de cero, uno y dos *pitches*. En el caso de que se tengan dos *pitches*, el *pitch* predominante es el que se considera como el de la voz cantada.

5.8. Limitaciones

En esta sección se comentan algunas limitaciones del algoritmo de detección de la frecuencia fundamental predominante.

Una primera observación, es que el método de detección de *pitch* implementado considera a toda la señal compuesta por sonidos sonoros, lo cual no es necesariamente cierto. Si bien se trabaja con fragmentos de corta duración y en las canciones en general el porcentaje de sonidos sonoros alcanza un 90 %, hay un porcentaje de sonidos sordos en los cuales se comete error. La presencia de sonidos sordos en la señal influye de manera importante, ya que cuando el algoritmo se encuentra frente a un sonido de este tipo, lo trata como sonoro y en general falla; ya que el *pitch* en ese caso debería de ser cero pero computa un valor distinto de cero, que viene dado por el acompañamiento musical presente en ese momento.

Otra observación importante viene por el lado de las probabilidades. Debido a que el algoritmo implementado utiliza parámetros que fueron ajustados utilizando una cierta base de datos, puede ocurrir que para algunas señales los parámetros utilizados puedan no ajustarse según la base de datos utilizada. Este inconveniente no tiene en principio una solución alternativa, ya que la única solución sería ajustar nuevamente los parámetros con una nueva base de datos.

Un último detalle de la implementación de esta función es un error al calcular la frecuencia fundamental f_0. Este error es común en este tipo de funciones y se conoce como error de $2f_0$, el cual provoca que el algoritmo detecte $2f_0$ en lugar de f_0 (un armónico de f_0).

5.9. Ejemplos y resultados obtenidos

En esta sección se presentan ejemplos y resultados obtenidos tanto para voz cantada como para voz hablada. Los algoritmos para voz cantada y voz hablada funcionan de manera similar, cambiando casi únicamente los valores de las constantes y parámetros utilizados.

Para la evaluación del algoritmo de detección de *pitch* se procedió de la siguiente manera. Primero se calcula el *pitch* con *WaveSurfer* para la señal de sólo voz cantada extraída a partir de los discos compactos de *karaoke*,[2] luego se compara con el obtenido por el algoritmo implementado. Calcular de este modo el *pitch* con *WaveSurfer* es lo más cercano al *pitch* ideal que se puede obtener.

Cabe destacar que a diferencia de *WaveSurfer* y *Praat*, el algoritmo implementado calcula la frecuencia fundamental f_0 de la voz cantada a partir de la grabación musical. Puede verse que si se utiliza tanto *WaveSurfer* como *Praat* para intentar calcular el *pitch* para voz cantada a partir de una grabación musical (es decir, voz cantada y acompañamiento musical juntos), los mismos muestran resultados totalmente erróneos.

Además de realizar la evaluación descrita anteriormente, también se probó una comparación utilizando las mismas señales de entrada en el algoritmo implementado y en *WaveSurfer*. Se tomaron archivos de sólo voz cantada y se calculó el *pitch* con *WaveSurfer* y con el código implementado para comparar resultados en las mismas condiciones, como era de esperarse se obtuvieron mejores resultados de este modo.

A continuación se presentan algunos ejemplos y se mencionan los lugares en donde se cometió error.

Las siguientes figuras tienen como objetivo comparar el *pitch* obtenido con el algoritmo implementado con el calculado con *WaveSurfer*.

La figura 5.3 muestra el *pitch* calculado utilizando *WaveSurfer* junto con el *pitch* calculado con el algoritmo implementado, para el caso del archivo de voz hablada “v3.wav”

[2]En el apéndice F.4 se explica cómo obtener el *pitch* para señales de sólo voz cantada utilizando *WaveSurfer*.

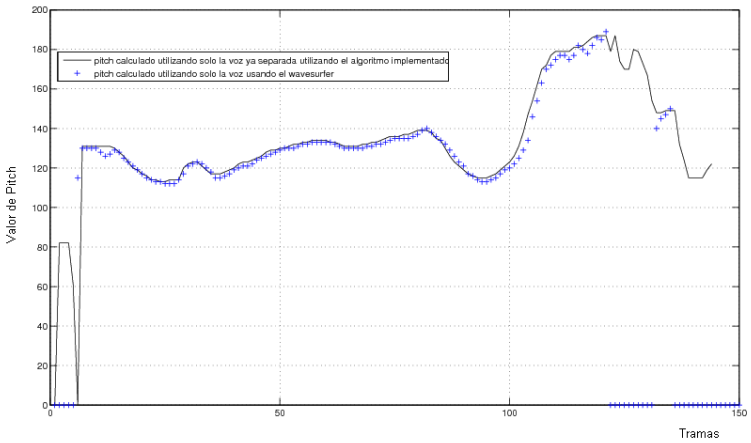

Figura 5.3: *Pitch* obtenido con el algoritmo implementado comparado con el obtenido con *WaveSurfer*. El ejemplo es de voz hablada ("v3.wav").

En la figura anterior se aprecia con cruces azules el *pitch* calculado utilizando *WaveSurfer*, y en negro continuo el obtenido con el algoritmo implementado. Puede observarse claramente que el *pitch* obtenido está cerca del obtenido con *WaveSurfer*, sobretodo al principio y casi hasta el final. El error que se observa al final es debido a que el algoritmo implementado no tiene en cuenta los sonidos sordos.

En la siguiente figura se muestra el *pitch* obtenido para el archivo "v3n3.wav". El mismo consiste en voz hablada (la misma frase que el archivo "v3.wav") con interferencia del tipo *cocktail party* de fondo.

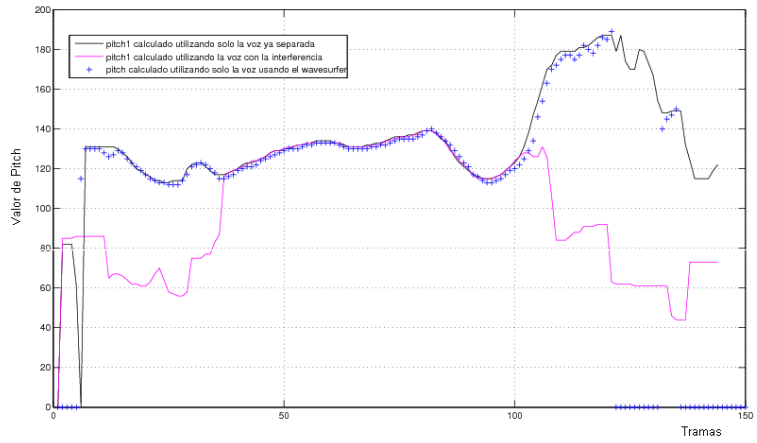

Figura 5.4: *Pitch* obtenido para la señal de voz hablada e interferencia del archivo "v3n3.wav", comparada con el obtenido para la misma señal de voz pero sin interferencia (archivo "v3.wav") utilizando *WaveSurfer* y el código implementado.

En la figura 5.4 puede observarse que inicialmente el *pitch* calculado con el algoritmo

implementado (trazo rosado) está por debajo del verdadero *pitch* (cruces azules), pero sin embargo la forma es similar. Realizando un cálculo sencillo se puede observar que en esos momentos el "pitch1" está tomando el valor de $f_0/2$ y no con f_0. Luego en la parte central el *pitch* calculado coincide en gran parte con el *pitch* verdadero. En la parte final el *pitch* calculado desciende su valor y no vuelve a subir, y cuando debería caer a cero no lo hace y queda en un valor intermedio. Cabe destacar que en este caso se está utilizando voz hablada y en el código implementado no se modificaron los parámetros, se mantuvieron los parámetros de voz cantada. Si se cambian los mismos el resultado obtenido mejora sensiblemente.

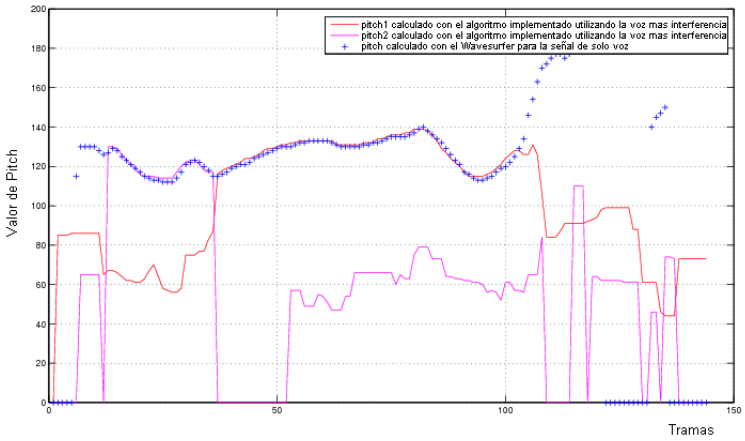

Figura 5.5: *Pitch* obtenido con el algoritmo implementado comparado con el *pitch* obtenido con *WaveSurfer*. En este caso se modificaron los parámetros para voz hablada. La señal es la misma de la figura 5.4.

En la figura 5.5 se observa la misma mezcla utilizada en la figura 5.4 con la diferencia de que en el código se cambiaron los parámetros de voz cantada para voz hablada. Puede observarse que nuevamente en la parte central el *pitch* obtenido (señal en rojo) es muy bueno. En la parte inicial del mismo ocurren dos hechos importantes a destacar, que ya se habían mencionado anteriormente. Primero se observa que al principio se tiene un error de $f_0/2$, este error ocurre por la misma razón que el error de $2f_0$. Luego de este error se puede ver que es el "pitch2" (señal rosada) el que por unos instantes toma el valor del *pitch* verdadero (cruces azules), y no el "pitch1" como se esperaba inicialmente. En la parte final nuevamente se comete un error de $f_0/2$.

En las siguientes figuras se muestran ejemplos para señales de voz cantada.

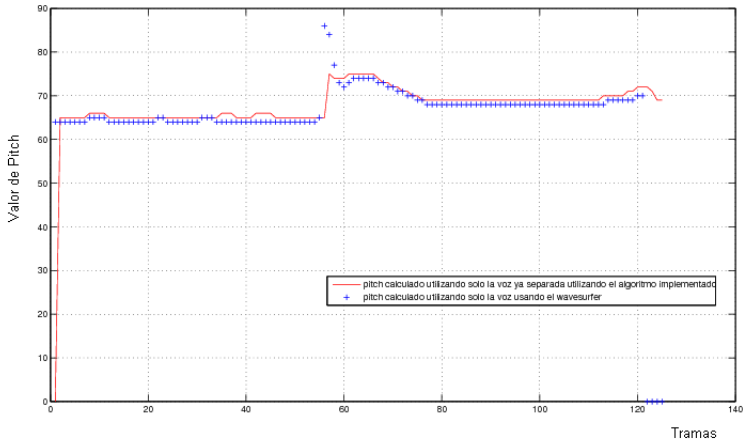

Figura 5.6: *Pitch* obtenido para voz cantada limpia sin acompañamiento musical, comparado contra el *pitch* verdadero calculado con *WaveSurfer*.

En la figura 5.6 se observa que el *pitch* obtenido en este caso es muy bueno, y que está cerca del verdadero *pitch*. En este caso la señal en cuestión es una señal limpia, sin acompañamiento musical, por lo tanto se está en la mismas condiciones que para el cálculo de *pitch* utilizando *WaveSurfer*. El fragmento utilizado se encuentra en la base de datos con el nombre "song10voz.wav".

La figura 5.7 muestra el *pitch* obtenido para el mismo fragmento de la figura 5.6, pero con el acompañamiento musical presente. Ahora la señal de entrada al algoritmo es la mezcla, es decir, la voz cantada con el acompañamiento musical (archivo utilizado "song10all.wav").

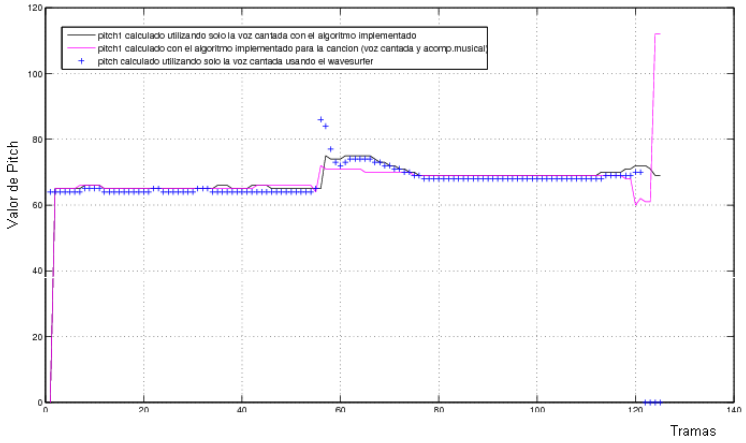

Figura 5.7: *Pitch* obtenido para la canción song10-all, comparado contra el *pitch* verdadero de *WaveSurfer*.

En la figura 5.7 se aprecia claramente que el resultado obtenido es nuevamente muy

bueno, hay que tener en cuenta que se está estimando el *pitch* para la voz cantada limpia a partir de la canción, es decir, la voz cantada con el acompañamiento musical. Se recuerda que en este caso ambas señales, la voz cantada y el acompañamiento musical están correlacionadas.

5.10. Evaluación y conclusiones

Para medir el desempeño del algoritmo de detección de *pitch* implementado, se utiliza como medida un error relativo, el cual está dado por la siguiente ecuación:

$$\Delta f = \frac{|PDA_{output} - f_0|}{f_0} \tag{5.13}$$

Donde PDA_{output} es la salida del algoritmo de detección de *pitch*, la cual es en este caso "pitch1" y f_0 es la frecuencia fundamental, el *pitch* verdadero, cuyo valor es el obtenido con *WaveSurfer* como se mencionó anteriormente.

Luego se define la tasa de error como el porcentaje de tramas temporales en las cuales $\Delta f > 20\,\%$. En la tabla 5.3 se muestran las tasas de errores obtenidas para distintos ejemplos.

Tipo de señal	Archivo	Tasa de error (%)
voz hablada	v3	16
voz hablada + cocktail party	v3n3	50
voz cantada	song-10-voz	2.71
voz cantada+acompañamiento musical	song-10-all	3.13
voz cantada+acompañamiento musical	todo51	16
voz cantada+acompañamiento musical	todo52	17
voz cantada+acompañamiento musical	todo53	43
voz cantada+acompañamiento musical	todo11c	9
voz cantada+acompañamiento musical	todo12c	13
voz cantada+acompañamiento musical	todo13c	33
voz cantada+acompañamiento musical	todo61	20
voz cantada+acompañamiento musical	todo62	57
voz cantada+acompañamiento musical	todo63	16
voz cantada+acompañamiento musical	todo71	49
voz cantada+acompañamiento musical	todo72	3
voz cantada+acompañamiento musical	todo73	12
voz cantada+acompañamiento musical	todo91	20
voz cantada+acompañamiento musical	todo92	21
voz cantada+acompañamiento musical	todo93	66
voz cantada+acompañamiento musical	todo101	58
voz cantada+acompañamiento musical	todo102	14
voz cantada+acompañamiento musical	todo103	72

Tabla 5.3: Tasa de error para fragmentos de audio.

La tasa de error promedio considerando sólo los fragmentos de voz cantada con acompañamiento musical es un poco menor al 30 %. Si bien es un porcentaje de error

alto, este valor es comparable con las tasas de error presentadas en los artículos [26], [1], [25]. En [25] se presenta una tabla con las tasas de error de diferentes métodos de detección de *pitch*. Las mismas son de 16 % para el método propuesto en [25], 44.3 % para un método propuesto por los mismos autores de [26] y de similares características, 45.3 % para el método presentado por Klapuri en [44] y 73 % para un método de detección de *pitch* basado en correlogramas propuesto en [45]. Por lo tanto estamos en condiciones de afirmar que el algoritmo de detección de *pitch* implementado, basado en [25] está a la par de los algoritmos existentes, y que el mismo lleva a buenos resultados.

Capítulo 6

Segmentación

En las etapas anteriores, la señal de entrada ha pasado por el modelo auditivo, donde se realizó una descomposición en unidades T-F. Luego a cada una de ellas se le calcularon varias características, además de la frecuencia fundamental predominante.

Este capítulo presenta la siguiente etapa, que es la etapa de segmentación de las unidades T-F. Un segmento es una región de unidades T-F, se pretende que cada uno de ellos pertenezca a una misma fuente (una de las fuentes que genera la señal de entrada). Cabe destacar que el segmento contiene más información que una unidad T-F aislada, ya que contiene información sobre la continuidad temporal y en frecuencia.

Considerando que las señales de voz son continuas tanto en el tiempo como en frecuencia, las unidades T-F vecinas en el tiempo tienden a ser originadas por la misma fuente. Asimismo, debido a la forma de los filtros del modelo auditivo, los cuales hacen que haya un solapamiento significativo entre los canales de frecuencia, un armónico activa cierto número de canales adyacentes. Esto lleva a realizar la formación de segmentos teniendo en cuenta la continuidad temporal y la correlación cruzada entre canales. En los canales de alta frecuencia, serán las envolventes de las respuestas de los filtros las que presenten similares patrones de periocidad en canales adyacentes. Es así que en los canales de alta frecuencia se realiza la segmentación en base a la continuidad temporal y a la correlación cruzada entre las envolventes de los canales.

Debido a que el correlograma es una representación de la periocidad de la señal, este método está pensado para que funcione bien sólo en las partes sonoras, no esperándose buenos resultados en aquellas partes no periódicas, como las sordas.[1]

6.1. Marcado de unidades T-F

En una primera instancia se realiza el marcado de las unidades T-F, el cual consiste en marcar las unidades con alta correlación cruzada entre canales (implica que el origen de esas señales provienen de la misma fuente). Tomando en cuenta el efecto de la modulación en amplitud para los armónicos no resueltos en canales de alta frecuencia, en forma separada se marcan las unidades con alta correlación cruzada entre las envolventes de los canales.

[1]Por más información sobre sonidos sonoros y sordos ver el apéndice A.

En los canales de baja frecuencia se marcan dos unidades T-F adyacentes $u_{c,m}$ y $u_{c+1,m}$[2] con un valor de 1 si se cumple que:

$$C_H(c,m) > \theta_H \tag{6.1}$$

Donde $C_H(c,m)$ es la correlación cruzada entre $u_{c,m}$ y $u_{c+1,m}$ y $\theta_H = 0{,}986$.[3]

En los canales de alta frecuencia se marcan unidades de dos tipos. Dos unidades T-F se marcan con un valor de 1 utilizando el mismo criterio que en baja frecuencia. Por otro lado, con el fin de identificar las unidades T-F que responden a armónicos no resueltos, se marcan dos unidades T-F adyacentes con un valor de 2 si se cumple que:

$$C_E(c,m) > \theta_E \tag{6.2}$$

Donde $C_E(c,m)$ es la correlación cruzada entre las envolventes de $u_{c,m}$ y $u_{c+1,m}$ y $\theta_E = 0{,}975$.[4]

Las figuras 6.1 y 6.2 muestran ejemplos del marcado de unidades T-F, para el caso de voz hablada y voz cantada respectivamente.

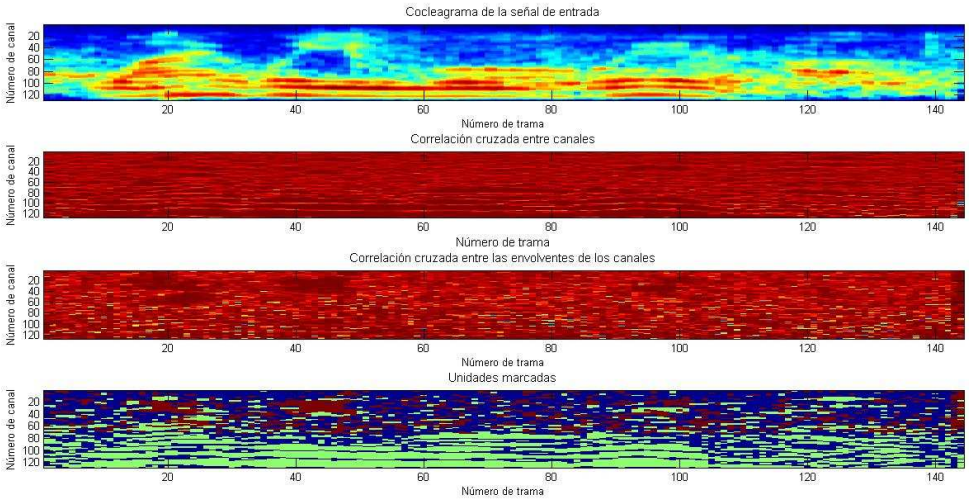

Figura 6.1: En la figura superior se observa el cocleagrama de la señal de entrada. En las siguiente figuras se observan los resultados de las funciones de correlación cruzada entre canales, y de la correlación cruzada entre las envolventes de los canales respectivamente. Las zonas más oscuras son las que presentan mayores valores de correlaciones. Finalmente en la figura inferior se pueden ver las unidades T-F marcadas, en verde están las unidades marcadas con 1, mientras que en rojo se aprecian las que fueron marcadas con 2. El ejemplo es voz hablada más "cocktail party".

[2]Donde $u_{i,j}$ es la unidad T-F del canal c, en la trama m-ésima.
[3]Sobre el valor de θ_H ver el capítulo 9.
[4]Sobre el valor de θ_E ver el capítulo 9.

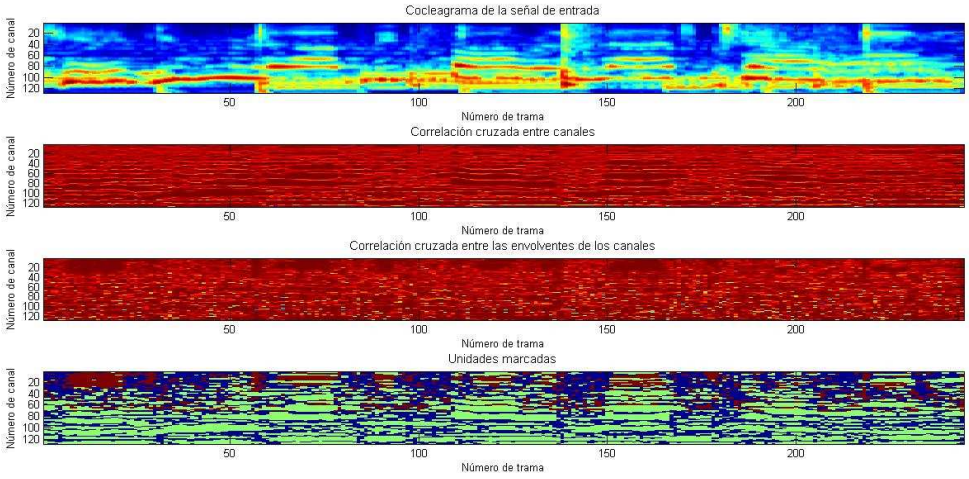

Figura 6.2: En la figura superior se observa el cocleagrama de la señal de entrada. En las siguiente figuras se observan los resultados de las funciones de correlación cruzada entre canales, y de la correlación cruzada entre las envolventes de los canales respectivamente. Las zonas más oscuras son las que presentan mayores valores de correlaciones. Finalmente en la figura inferior se pueden ver las unidades T-F marcadas, en verde están las unidades marcadas con 1, mientras que en rojo se aprecian las que fueron marcadas con 2. El ejemplo es voz cantada más acompañamiento musical.

Analizando estas figuras se pueden observar varias cosas. En primer lugar se aprecia claramente que la figura que se forma al marcar las unidades es muy parecida al cocleagrama, lo que indica que realmente se están identificando las zonas que pertenecen a un mismo componente acústico, ya que cada uno activa una región en el mapa de unidades. También puede verse que las zonas más oscuras en las gráficas de las correlaciones cruzadas finalmente coinciden con las unidades marcadas en verde o rojo, ya sea que son marcadas con 1 o 2 respectivamente. Asimismo se observa la utilidad de marcar las unidades considerando los efectos de los armónicos resueltos y no resueltos, ya que es la concatenación de ambos marcados de unidades lo que lleva a que la gráfica completa se parezca al cocleagrama.

6.2. Formación de segmentos

Las unidades T-F vecinas con la misma marca se juntan en segmentos, formando por tanto segmentos tipo-1 y segmentos tipo-2, de acuerdo a la naturaleza de sus armónicos, es decir, si son resueltos o no.

Dos unidades son consideradas vecinas si comparten el mismo canal y aparecen en tramas temporales consecutivas, o si comparten la misma trama y están en canales adyacentes.

En la figura 6.3 se observa un ejemplo de la formación de segmentos.

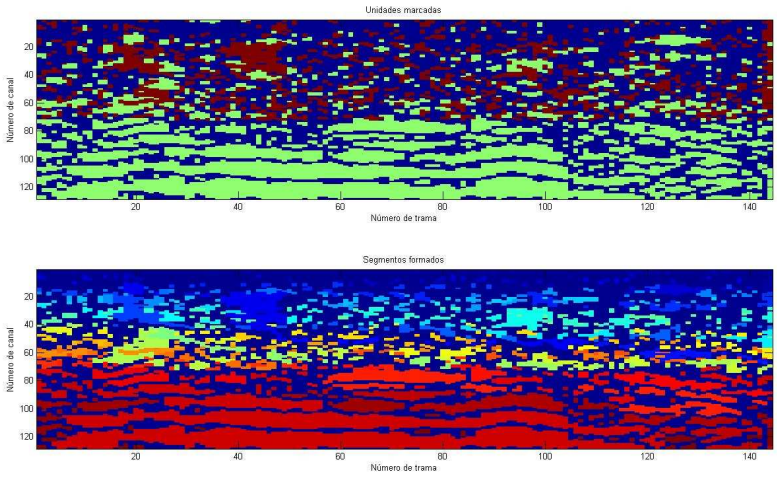

Figura 6.3: La figura superior muestra las unidades marcadas, en la figura inferior se observan los segmentos formados, cada uno de ellos representado por un color distinto.

Una vez que se termina de formar los diferentes segmentos, el próximo paso es agruparlos con el fin de formar el *stream* final, el cual está constituido por segmentos provenientes de una misma fuente, en este caso de la voz cantada. Los criterios para agrupar los segmentos se presentan en el siguiente capítulo.

Capítulo 7

Agrupamiento

En este capítulo se presenta la etapa de agrupamiento. El objetivo es obtener una máscara binaria llamada *stream* formada por las unidades T-F. Un valor de 1 en dicha máscara indica que la unidad pertenece a la señal de voz cantada, mientras que un valor de 0 indica que pertenece a la señal de acompañamiento musical.

En una primera instancia se realiza un etiquetado individual de las unidades T-F. Las unidades en las que la voz predomina sobre el acompañamiento son etiquetadas como voz dominante, mientras que las otras se etiquetan como acompañamiento dominante.

Luego se etiqueta cada segmento como voz dominante o acompañamiento dominante, teniendo en cuenta si la suma de la energía de sus unidades T-F etiquetadas como voz dominante es mayor a la energía total de todo el segmento. Los segmentos etiquetados como voz dominante son agrupados al *stream* final. Finalmente con el objetivo de refinar el resultado se remueven regiones significativas de unidades T-F etiquetadas como acompañamiento dominante, y se agregan unidades T-F vecinas etiquetadas como voz dominante que no pertenecen a ningún segmento.

7.1. Etiquetado de unidades

En esta etapa se etiqueta cada unidad T-F, ya sea como voz dominante o acompañamiento dominante. Se procede de distinta manera según los segmentos hayan sido formados teniendo en cuenta si sus armónicos eran resueltos o no. En las bajas frecuencias, para etiquetar una unidad se compara su periocidad con el período del *pitch* predominante en esa trama. Por otro lado, en las altas frecuencias, las respuestas son moduladas en amplitud y sus envolventes fluctúan a la frecuencia fundamental f_0, por lo que se compara la periocidad de su envolvente con el período del *pitch* predominante.

Las unidades T-F $u_{c,m}$ que pertenecen a segmentos tipo-1 son etiquetadas como voz dominante, si la autocorrelación en el canal c y la trama m[1] evaluada en el *pitch* estimado $\tau_S(m)$, es comparable con el máximo valor de la autocorrelación dentro del posible rango de *pitch* Γ.[2] La siguiente ecuación realiza lo anterior:

[1] Elemento (c,m) del correlograma

[2] Entre 2 ms y 12.5 ms (32 y 200 muestras respectivamente, con una frecuencia de muestreo de 16 kHz)

$$\frac{A_H(c, m, \tau_S(m))}{\text{máx}_{\tau \in \Gamma} A_H(c, m, \tau)} > \Theta_T \tag{7.1}$$

Donde $\Theta_T = 0{,}688$.[3] En el caso de no cumplir esta condición, la unidad T-F se etiqueta como acompañamiento dominante.

El resto de las unidades T-F se etiquetan como voz dominante si el valor del correlograma de la envolvente, evaluado en el *pitch* estimado, es comparable con su valor máximo dentro del posible rango de *pitch*:

$$\frac{A_E(c, m, \tau_S(m))}{\text{máx}_{\tau \in \Gamma} A_E(c, m, \tau)} > \Theta_A \tag{7.2}$$

Donde $\Theta_A = 0.688$.[4]

En la figura 7.1 se muestra un ejemplo del etiquetado de unidades.

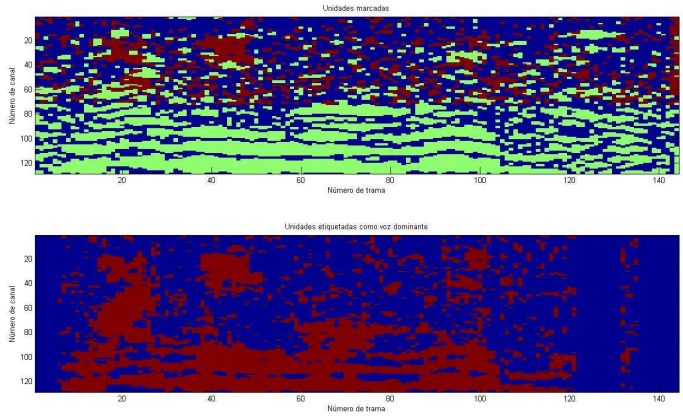

Figura 7.1: En la figura superior se pueden ver las unidades marcadas, en verde son las tipo 1 y en rojo las tipo 2. En la figura inferior se observan en rojo las unidades etiquetadas como voz dominante, y las restantes son la etiquetadas como acompañamiento dominante. El ejemplo es el de voz hablada más *cocktail party*

7.2. Agrupación de segmentos

El objetivo de esta etapa es hallar las unidades T-F en las cuales la voz predomina sobre el acompañamiento, con el fin de formar el *stream* final. Para indicar qué unidades T-F cumplen lo anterior, se utiliza una matriz M de C filas, y T columnas, siendo C la cantidad de canales y T la cantidad de tramas. Los valores posibles para cada elemento de esta matriz son 1 o 0: el elemento $M_{i,j}$ es 1 si en la unidad T-F, del canal i, y trama j, predomina la voz sobre el acompañamiento, o 0 en el caso contrario. A esta matriz se le denomina *stream* o "máscara binaria", y es lo que se pretende obtener en esta etapa. El proceso de agrupación se divide en tres etapas que se describen a continuación.

[3] Sobre el valor de θ_T ver el capítulo 9.
[4] Sobre el valor de θ_E ver el capítulo 9.

La primera etapa es la más importante, en la cual se realiza una agrupación inicial de los segmentos formados en la etapa de segmentación. Un segmento es considerado como voz dominante y agrupado en el *stream* final, si se cumple que la suma de la energía correspondiente a sus unidades etiquetadas como voz dominante, es mayor a la mitad de la energía contenida en todo el segmento. Esto implica, que dado un segmento s que cumpla con lo anterior, todas las unidades T-F que forman el segmento s, tendrán sus correspondientes elementos en la máscara binaria con el valor 1. Se identifican de esta manera cuáles segmentos son voz dominante y cuáles no, para estos últimos se ponen todas sus unidades en la máscara binaria en -1, de forma tal de poder identificarlos. Se crea en esta etapa un *stream* inicial.

En la figura 7.2 se muestran las gráficas que ilustran el proceso. En la figura inferior se puede ver el *stream* inicial formado, mientras que en las figuras superiores se muestran los segmentos formados, las unidades etiquetadas y el cocleagrama respectivamente. El ejemplo es de voz hablada.

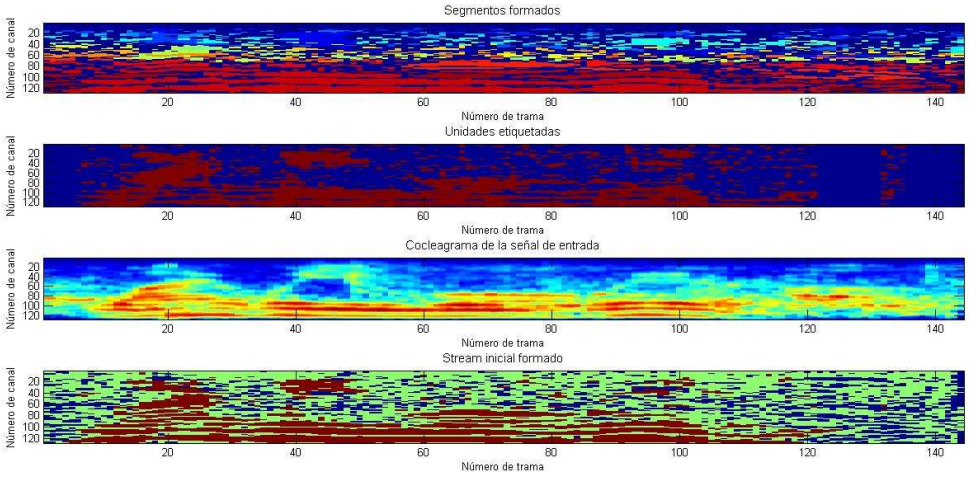

Figura 7.2: La figura inferior muestra el *stream* inicial formado. En rojo están las unidades pertenecientes a segmentos etiquetados como voz dominante, tienen valor 1. En azul se observan las unidades pertenecientes a segmentos que no fueron agrupados en esta etapa, y por lo tanto son considerados como acompañamiento dominante, tienen valor -1. En verde se ven las unidades T-F que no pertenecen a ningún segmento, tienen valor 0. En las figuras superiores se muestran los segmentos formados, las unidades etiquetadas y el cocleagrama respectivamente. El ejemplo es voz hablada más *cocktail party*.

Las próximas dos etapas son de refinamiento del proceso descrito anteriormente. La segunda etapa consiste en eliminar del *stream* anterior, las regiones de unidades T-F etiquetadas como acompañamiento dominante mayores a $40ms$. Estas regiones son aquellas formadas por unidades T-F que fueron agrupadas al *stream* inicial por pertenecer a un segmento etiquetado como voz dominante, pero que individualmente fueron etiquetadas como acompañamiento dominante. Por tanto, se buscan regiones de unidades T-F

que cumplan con lo anterior, y si están formadas por más de 3 tramas[5] se eliminan del *stream*, o sea, a las unidades de dichas regiones se les asigna el valor de -1. En la figura 7.3 se muestra el *stream* inicial formado, las regiones de unidades T-F etiquetadas como acompañamiento dominante pertenecientes a dicho *stream*, y por último el *stream* resultante luego de eliminar las regiones mayores a 40 ms.

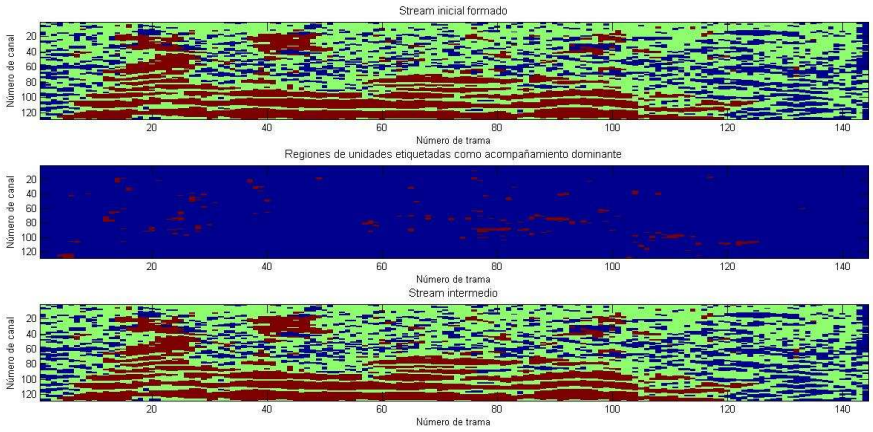

Figura 7.3: La figura superior muestra el *stream* inicial formado. La figura central muestra sus regiones de unidades T-F etiquetadas como acompañamiento dominante. Finalmente en la figura inferior se muestra el *stream* obtenido después de eliminar las regiones mayores a 40 ms. El ejemplo es voz hablada más *cocktail party*.

La tercera y última etapa, tiene como objetivo reunir más unidades T-F etiquetadas como voz dominante para agregarlas al *stream*. Para ello se buscan unidades etiquetadas como voz dominante en la vecindad de los segmentos etiquetados como voz dominante, y que no pertenezcan a ningún segmento. Esto quiere decir que se buscan unidades etiquetadas como voz dominante que cumplan que, actualmente tengan valor 0 en la máscara binaria, y que sean vecinas a unidades que actualmente valen 1. Dichas unidades toman luego el valor 1 en la máscara.

Finalmente para hallar el *stream* final hay que quedarse con aquellas unidades T-F que tienen el valor 1 en la máscara, y las demás deben ser puestas en 0. De esta forma, se obtiene la máscara binaria, en donde las unidades T-F con un 1 son aquellas en las cuales la voz predomina por sobre el acompañamiento, por lo que se tendrán en cuenta a la hora de la resíntesis. En la figura 7.4 se puede ver el *stream* de la etapa anterior, las unidades etiquetadas y finalmente el *stream* final obtenido por el algoritmo.

[5]Cabe señalar que cada trama está separada cada 10 ms de la siguiente, por lo que si un segmento tiene más de 3 tramas, significa que el segmento es igual o mayor a 40 ms.

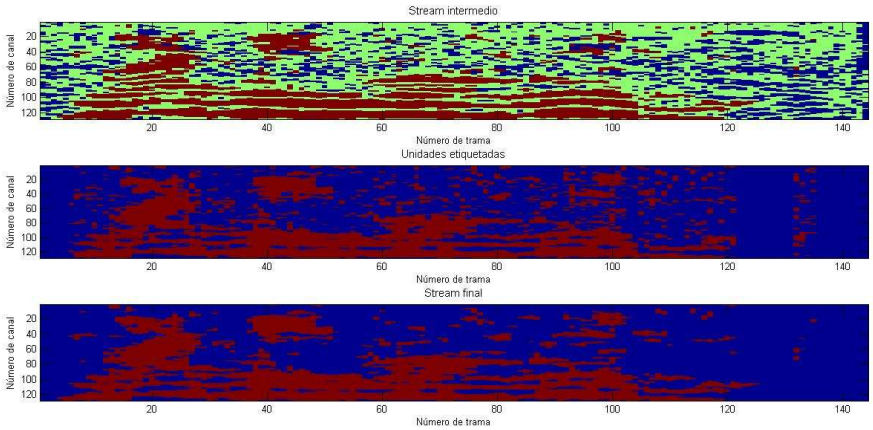

Figura 7.4: En la figura superior se muestra el *stream* intermedio obtenido en la etapa anterior, mientras que en la figura central se puede observar a las unidades etiquetadas. En la figura de inferior se observa el *stream* final obtenido, en rojo se muestran las unidades T-F que se identificaron como voz dominante. El ejemplo es voz hablada más *cocktail party*.

7.3. Resultados y conclusiones

A modo de comparación, en la figura 7.5 se puede observar el *stream* final obtenido, junto con la máscara ideal binaria para el ejemplo de voz hablada. Se observa que hay similitud entre ambas.

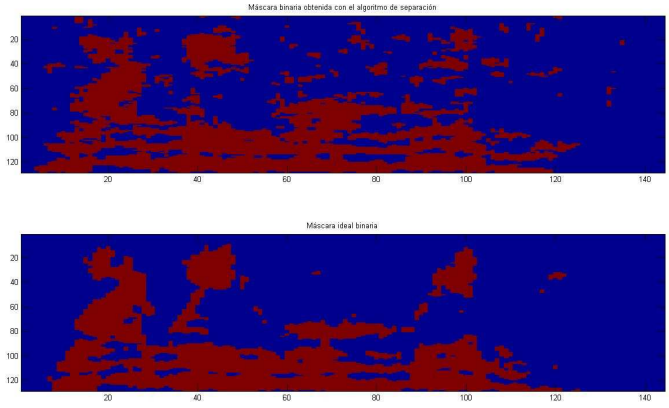

Figura 7.5: En la figura superior se observa la máscara binaria obtenida con el código implementado. En la figura inferior se ve la IBM. El ejemplo es voz hablada más *cocktail party* (v3n3.wav).

En la figura 7.6 se aprecia un ejemplo para un ejemplo de voz cantada. Las máscaras son similares, aunque se notan diferencias en las zonas de alta frecuencia, en donde se

realiza la segmentación y el agrupamiento en forma diferente. En esta zona se encuentran más problemas para realizar la separación, ya que como se explicó anteriormente, allí los armónicos son generalmente no resueltos y las envolventes de las respuestas fluctúan a f_0. Además en dicha zona, el acompañamiento es más predominante que en las bajas frecuencias.

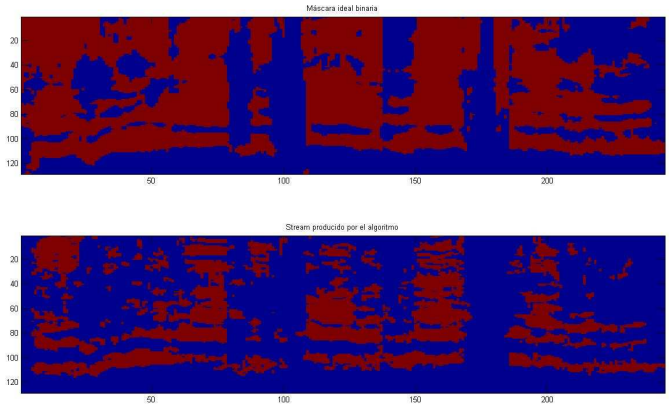

Figura 7.6: En la figura superior se puede ver la máscara binaria obtenida con el código implementado. En la figura inferior se ve la IBM. El ejemplo es de voz cantada más acompañamiento musical (todo61.wav).

La señal de entrada está compuesta por dos fuentes acústicas, la señal de voz y el acompañamiento. En la figura 7.7 se pueden ver los cocleagramas del ejemplo de voz hablada. Se muestra el correspondiente al de la señal de voz, al de la interferencia y finalmente al de la mezcla. Se puede observar que el cocleagrama de la señal de mezcla es una superposición de los otros dos.

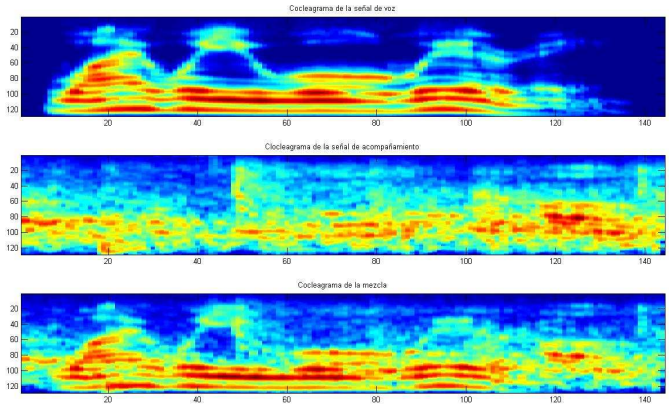

Figura 7.7: En la figura superior se puede ver el cocleagrama de la señal de voz, en la central el de la señal de interferencia, y en la inferior el de la mezcla. El ejemplo es voz hablada más *cocktail party*, archivos v3n3.wav, v3.wav y n3.wav.

El objetivo del sistema es identificar cuáles regiones del cocleagrama de la mezcla pertenecen a la señal de voz. El *stream* es justamente un indicador de en qué zonas la voz predomina por sobre el acompañamiento. En la figura 7.8 se ilustra lo anterior. En ella se puede observar el cocleagrama de la señal de voz original, junto con el cocleagrama de la mezcla enmascarado con la máscara ideal binaria y con el *stream* obtenido con el algoritmo. Se observa la similitud entre los cocleagramas enmascarados con el cocleagrama de la voz original. Además se puede ver que la gráfica perteneciente a la IBM es muy parecida a la de la señal de voz, lo que muestra la validez del concepto de la máscara ideal binaria.

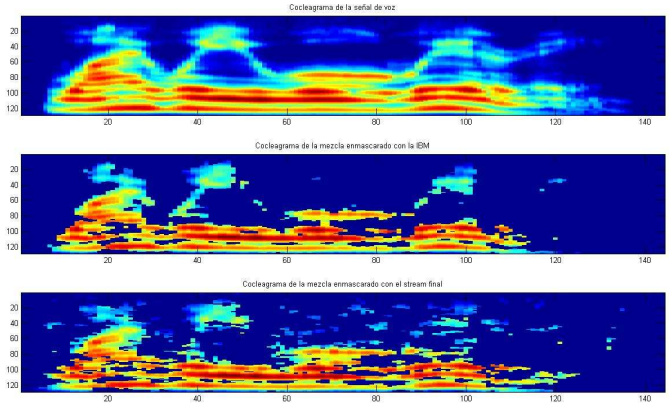

Figura 7.8: En la figura superior se encuentra el cocleagrama de la señal de voz original, en la central el cocleagrama de la mezcla enmascarado con la máscara ideal binaria, y en la inferior el enmascarado con la máscara producida por el algoritmo. El ejemplo es el mismo que el anterior.

A modo de ejemplo se presenta lo mismo pero para el caso de voz cantada más acompañamiento musical, lo cual está en las gráficas 7.9 y 7.10.

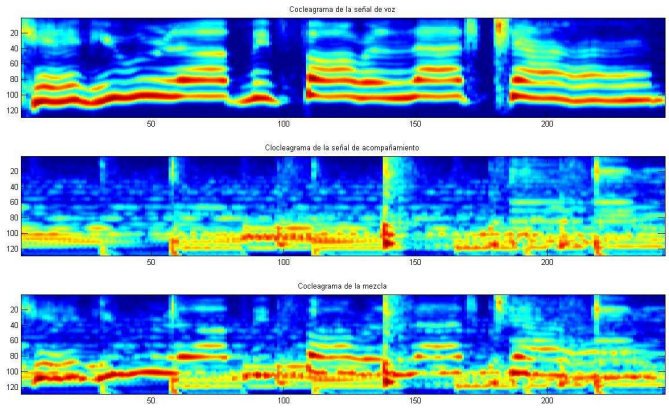

Figura 7.9: En la figura superior se puede ver el cocleagrama de la señal de voz, en la central el de la señal de acompañamiento musical, y en la inferior el de la mezcla. El ejemplo es voz cantada más acompañamiento musical, los archivos utilizados son todo61.wav, voz61.wav y acom61.wav

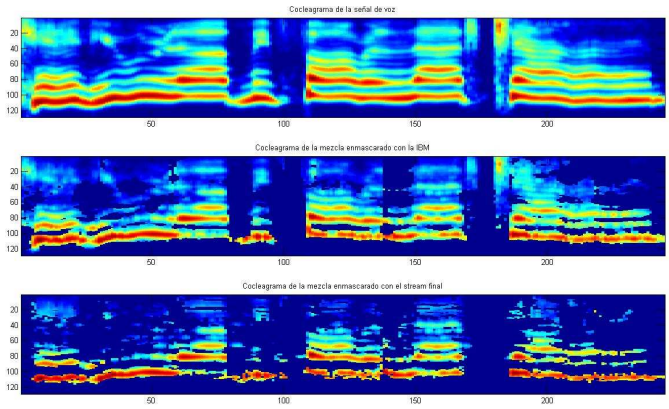

Figura 7.10: En la figura superior se encuentra el cocleagrama de la señal de voz, en la central el cocleagrama de la mezcla enmascarado con la máscara ideal binaria, y en la inferior el enmascarado con la máscara producida por el algoritmo. El ejemplo es el mismo que el anterior.

Se puede observar que nuevamente se arriba a buenos resultados.

Capítulo 8

Resíntesis

La función de resíntesis constituye el bloque final del algoritmo de separación, la cual permite obtener la forma de onda en el dominio del tiempo de la señal de voz cantada. Para ello, en esta etapa se utilizan como entradas el *stream* final obtenido en la etapa de agrupamiento y la salida del banco de filtros auditivos.

En principio, lo primero que se espera al recuperar la señal de voz cantada, es poder reproducirla y escucharla, con el fin de observar si la salida obtenida es inteligible, es decir, si puede distinguirse lo que se está escuchando. Este es el método de evaluación más sencillo para el algoritmo implementado. En el capítulo 9 se desarrollaran métodos más formales para la evaluación del algoritmo.

El algoritmo de resíntesis, realiza la reconstrucción de la señal utilizando las unidades T-F a la salida de los filtros gammatone en donde el *stream* final vale 1, o sea, se utilizan las unidades en las cuales la voz predomina sobre el acompañamiento. Por lo tanto, se hace un enmascaramiento entre el mapa de unidades T-F obtenido por los filtros auditivos y el *stream* final.

A diferencia de etapas anteriores, en esta etapa es muy importante que el desfasaje entre muestras de la señal sea lo menor posible, ya que se reconstruye la señal muestra a muestra. Por tanto, como se mencionó en el capítulo 3 sobre la corrección de fase en los filtros gammatone, en este bloque se utiliza la salida del banco de filtros con corrección de fase mediante doble filtrado,[1] ya que aquí lo que interesa es la alineación entre diferentes canales de las muestras.

8.1. Implementación

La implementación del algoritmo de resíntesis se basó en la función utilizada en [1], la cual consiste en 3 pasos:

1 Quedarse con aquellas unidades T-F que valen 1 en el *stream* final, o lo que es lo mismo, con las unidades T-F en las cuales la voz predomina sobre el acompañamiento. Para ello se realiza un enmascaramiento entre el mapa de unidades T-F y el *stream* final.

[1]Se utiliza un filtrado hacia "adelante" y luego hacia "atrás", haciendo que la fase sea cero.

2 Utilizar una ventana para ponderar cada muestra dentro de las unidades T-F resultantes del paso anterior.

3 Sumar todas las señales que se obtienen del paso 2 para reconstruir la señal de voz cantada en el dominio del tiempo.

Para el cálculo de los pesos se utiliza una ventana con forma de coseno elevado:

$$coswin(i) = \frac{(1 + cos(\frac{2\pi(i-1)}{N} - \pi))}{2} \tag{8.1}$$

Donde i indica el número de muestra y varía entre 1 y el tamaño de la ventana utilizada (256 en el caso del algoritmo implementado).

Una vez obtenida la señal en el dominio del tiempo, se normaliza y luego se filtra con un filtro Butterworth de orden 3 y frecuencia de corte en 800 Hz. Esto permite eliminar ruido e interferencia en la señal obtenida.

8.2. Resultados y conclusiones

La etapa de resíntesis depende fuertemente de los resultados obtenidos en la etapa de separación. Cabe destacar, que si no se obtiene una buena separación, por muy bueno que sea el algoritmo de resíntesis, la reconstrucción de la señal no será buena.

Las siguientes figuras muestran algunos resultados obtenidos, aplicando el algoritmo de resíntesis tanto a voz hablada como a voz cantada.

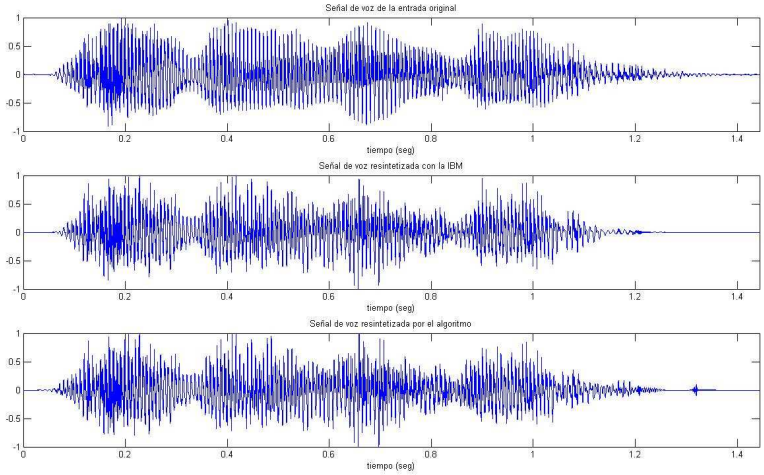

Figura 8.1: Formas de ondas obtenidas con el algoritmo de resíntesis para el archivo de voz hablada v3.n3.wav. En la figura superior se muestra la forma de onda original de la voz hablada, sin interferencia. En la figura central está la forma de onda obtenida al resintetizar con la IBM, mientras que en la figura inferior está la forma de onda obtenida al resintetizar con el *stream* final.

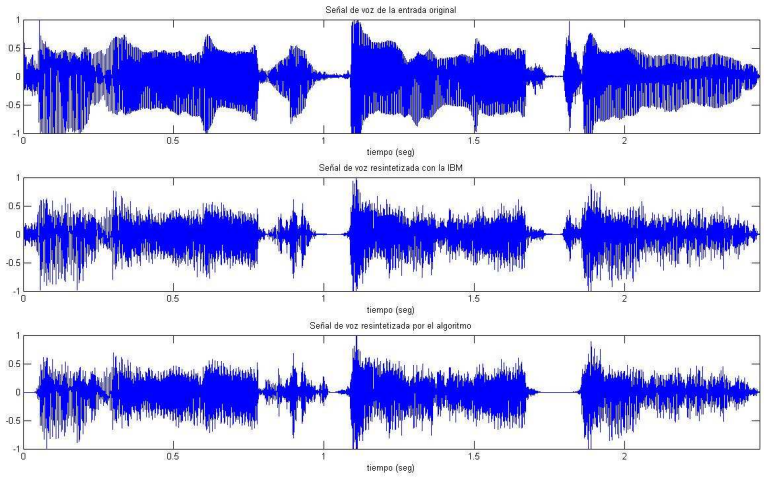

Figura 8.2: Formas de ondas obtenidas con el algoritmo de resíntesis para el archivo de voz cantada todo61.wav. En la figura superior se muestra la forma de onda original de la voz hablada, sin interferencia. En la figura central está la forma de onda obtenida al resintetizar con la IBM, mientras que en la figura inferior está la forma de onda obtenida al resintetizar con el *stream* final.

El objetivo computacional del sistema es obtener un *stream* final lo más parecido a la máscara ideal binaria. Por lo tanto, la reconstrucción de la señal utilizando la máscara ideal brinda la mejor forma de onda que puede obtenerse mediante este algoritmo.

En los ejemplos anteriores se puede ver que las formas de onda que se obtienen al resintetizar con la máscara ideal, son muy parecidas a las formas de onda de la voz originales. Asimismo, se puede observar que las formas de onda obtenidas con el algoritmo implementado son similares a las formas de onda obtenidas con la máscara ideal.

Capítulo 9

Evaluación y Resultados

En este capítulo se presenta el método de evaluación utilizado. Asimismo se explica el proceso realizado para la estimación de los parámetros que fueron utilizados en los capítulos 6 y 7.

Por otro lado, se presentan los resultados obtenidos en la evaluación del sistema, y finalmente se muestra un ejemplo de punta a punta realizando una discusión sobre los resultados obtenidos en cada etapa del algoritmo de separación de voz cantada.

9.1. Método de evaluación

Los conceptos principales para realizar la evaluación son la máscara binaria obtenida (*stream* final) y la máscara ideal binaria. Cabe recordar que el objetivo computacional del sistema es que el *stream* final sea lo más parecido posible a la IBM.

La máscara ideal puede obtenerse fácilmente si se tiene por separado la señal de voz cantada y la señal del acompañamiento musical. Primero se calcula la energía de cada unidad T-F para ambas señales, y luego se comparan estos resultados. Si la energía de la unidad perteneciente a la voz cantada es mayor o igual a la del acompañamiento musical, se asigna el valor 1 en la máscara ideal, de lo contrario toma el valor 0. Los fragmentos de señales necesarios para la obtención de la máscara ideal se obtuvieron a partir de discos compactos de *karaoke*.

Teniendo en cuenta lo anterior, la forma de onda que se obtiene al resintetizar la mezcla a partir de la máscara binaria ideal, es tomada como la señal de voz cantada de referencia. Si bien a priori, se podría pensar en utilizar la señal de voz cantada original para realizar las evaluaciones, esto no sería del todo correcto. En este caso no se estaría tomando en cuenta el verdadero objetivo del sistema, el cual es obtener la máscara binaria, así como tampoco se tendrían en cuenta el enmascaramiento auditivo y las distorsiones introducidas en la representación de la señal y en la resíntesis.

Según estudios realizados, la inteligibilidad de la señal obtenida con la máscara ideal binaria para el caso de voz hablada es muy buena [46], [47], [48].

En lo que sigue se presenta un criterio para medir la performance del sistema. Para cuantificar la mejora obtenida por el sistema, lo que se hace es calcular la relación señal a ruido (SNR por sus siglas en inglés *Signal to Noise Ratio*) antes y después de la

separación.

Luego, se toma como medida de performance la ganancia de la SNR, es decir, la diferencia entre la SNR antes y después de la separación. La SNR es una comparación entre la potencia de la señal portadora de información, y la potencia del ruido que obstaculiza la percepción de la información, como es habitual se mide en decibeles (dB). En el sistema planteado la SNR se define como [1]:

$$SNR = 10\log_{10}[\frac{\sum_n I^2(n)}{\sum_n (I(n) - O(n))^2}] \tag{9.1}$$

Cuando se considera el cálculo de la SNR antes de la separación, $I(n)$ es la señal de voz cantada obtenida al resintetizar la mezcla con la máscara ideal binaria, y $O(n)$ es la mezcla resintetizada con una máscara cuyos valores son todos 1, lo cual compensa la distorsión introducida en la resíntesis. Por otro lado, cuando se considera el cálculo después de la separación, $I(n)$ es la misma señal que la anterior, pero $O(n)$ es la salida del sistema.

Si se considera a la señal de entrada como la suma de $I(n)$ más una señal de error $e(n)$, tal que $I(n) + e(n) = O(n)$, la ganancia da una medida de cuánto se redujo la señal de error que acompaña a la señal de voz $I(n)$ al pasar por el sistema. Previo a la separación, se puede considerar que la señal $e(n)$ está principalmente compuesta por el acompañamiento musical. Luego de la separación, se puede considerar que $e(n)$ está compuesta por la señal de acompañamiento que no fue correctamente separada. Por lo tanto, la ganancia del sistema es una medida de cuánto se pudo disminuir la señal de acompañamiento musical.

Con el fin de realizar una evaluación intermedia al sistema, se implementa una comparación entre la máscara binaria obtenida (*stream* final) y la máscara ideal. De esta manera, se puede tomar un indicador que es independiente de la etapa de resíntesis. Para llevar a cabo la evaluación del *stream* obtenido, se compara la cantidad de unidades distintas entre ambas máscaras en relación a las unidades totales. Sea L la cantidad de canales y sea M la cantidad de tramas, se define entonces la siguiente medida de performance:

$$\Delta M = \frac{|IBM - stream|}{L \times M} \tag{9.2}$$

Por otra parte, debido a que el bloque de detección de la frecuencia fundamental es de vital importancia y en el mismo se introducen errores considerables, se decidió realizar la evaluación del sistema tomando en cuenta dos escenarios diferentes:

1. Utilizando el *pitch* hallado con *WaveSurfer* sobre la voz cantada solamente. De esta manera puede evaluarse al sistema sin considerar el bloque de detección de f_0, lo que permite evaluar toda la parte de separación sin considerar los errores de ese bloque.

2. Utilizando el *pitch* hallado por el algoritmo implementado. Aquí se realiza la evaluación de punta a punta del sistema, obteniéndose las medidas de performance del algoritmo implementado.

9.2. Estimación de parámetros

Se decidió realizar una estimación automática de los principales parámetros del sistema, los cuales son: θ_H, θ_E, θ_T y θ_A. Los mismos son muy importantes y tienen una clara influencia por sobre los resultados del sistema. Se recuerda que los parámetros θ_H y θ_E son usados en la etapa de segmentación para comparar la correlación cruzada entre canales adyacentes y la correlación cruzada entre las envolventes de canales adyacentes, respectivamente. Si las correlaciones toman valores más altos que θ_H o θ_E, se juntan ambas unidades adyacentes para formar parte del mismo segmento. Por otro lado, los parámetros θ_T y θ_A son utilizados en la etapa de agrupamiento para comparar el período de una unidad T-F o el período de la envolvente de una unidad T-F, con el período del *pitch* predominante en dicha unidad, respectivamente. Para ello se compara el valor máximo de la autocorrelación de la unidad o de la envolvente de la misma, contra el valor de la autocorrelación en el período del *pitch*, si dicho valor es mayor a θ_T o θ_A se etiqueta la unidad como voz predominante.

Con el fin de obtener los mejores valores para los parámetros mencionados, se realizó una variación automática de ellos observando los diferentes valores de la SNR a la salida del sistema. Finalmente se eligió el juego de parámetros que hicieran máxima la SNR.

Cabe destacar que se entendió conveniente realizar la estimación eliminando el bloque de detección de la frecuencia fundamental. De esta forma, se pone énfasis sólo en la separación, eliminando los errores producidos en la etapa de detección de f_0. La frecuencia fundamental puede hallarse con *WaveSurfer* sobre la señal de voz cantada, como se explica en el apéndice F.4.

En una primera instancia se procedió a variar los parámetros en forma manual, observando la manera en que cada uno afectaba al resultado final. Analizando los resultados obtenidos en los valores de las SNR, se estableció que los parámetros θ_T y θ_A podían variar de igual forma, ya que sus valores generalmente eran muy parecidos. De esta manera quedan tres parámetros a variar, lo cual permite disminuir los tiempos de procesamiento.

Como resultado de las pruebas anteriores, se obtuvieron rangos de variación para los tres parámetros, según se muestra en la tabla 9.1, de forma tal de poder realizar medidas de la SNR en una grilla fina de valores.

	valor inicial	**salto**	**valor final**
θ_H	0.979	0.003	0.991
θ_E	0.97	0.003	0.982
θ_{TA}	0.65	0.03	0.77

Tabla 9.1: Variación de los parámetros estimados.

Por lo tanto, para cada fragmento se obtiene un cubo de $5 \times 5 \times 5$ cuyos valores son las SNR a la salida del sistema.

Se utilizaron 11 fragmentos de canciones para realizar la estimación, y para cada uno de ellos se obtuvo el cubo mencionado anteriormente. Luego se buscó el juego de

parámetros $(\theta_H, \theta_E, \theta_{TA})$ que maximice la SNR en cada fragmento. Una vez obtenidos los valores máximos, se decidió promediar los valores de los parámetros que provocaban la máximas SNR, obteniéndose finalmente los valores de los parámetros que se utilizan en el algoritmo. En la figura 9.1 se observa la variación de los parámetros máximos, correspondiendo cada uno a un fragmento distinto.

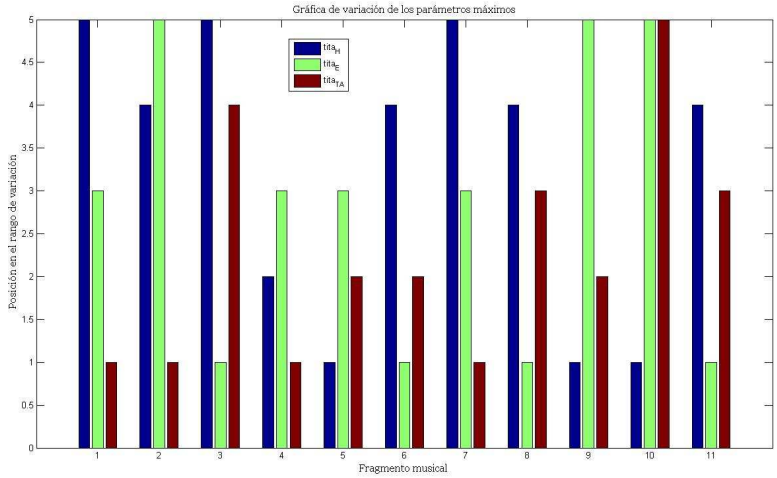

Figura 9.1: Variación de los parámetros máximos de cada fragmento. En el eje de las abscisas está el número de fragmento, mientras que en el eje de las ordenadas está la posición en el rango de variación de cada parámetro.

Los valores finales de los parámetros son los siguientes: $\theta_H = 0{,}986$, $\theta_E = 0{,}975$ y $\theta_{TA} = 0{,}688$. Cabe destacar que como punto de partida se tomaron los valores del artículo [16], los cuales son: $\theta_H = 0{,}99$, $\theta_E = 0{,}99$, $\theta_T = 0{,}85$ y $\theta_A = 0{,}7$.

9.3. Resultados

Para realizar la evaluación del sistema se construyó una base de datos compuesta por 18 fragmentos de grabaciones musicales de corta duración. Para dichos fragmentos se tiene disponible de antemano la voz cantada y el acompañamiento musical en forma separada. Se trató de encontrar ejemplos que contemplen distintos casos, como por ejemplo fragmentos que contengan principalmente sonidos sonoros u otros con muchos sonidos sordos, también que hayan tanto cantantes masculinos como femeninos o que se encuentren distintos tipos de acompañamientos. Todos ellos son fragmentos en idioma inglés de música rock, pop y country.[1]

La tabla 9.2 presenta las medidas de performance realizadas para el caso 1, es decir, tomando como frecuencia fundamental en cada trama la hallada con *WaveSurfer*.

[1]En la base de datos también hay disponibles unos pocos fragmentos de canciones en idioma español. Esto se debe a que no se disponía de discos de *karaoke* con canciones en español para realizar la separación. Los resultados obtenidos para estos ejemplos fueron muy buenos.

Archivo	SNR antes (dB)	SNR después (dB)	Ganancia (dB)	ΔM (%)
todo51	2.39	10.53	8.14	21
todo52	2.08	6.59	4.51	25
todo53	3.67	7.37	3.7	28
todo11c	8.56	11.46	2.9	25
todo12c	8.7	14.15	5.45	29
todo13c	2.68	6.09	3.41	30
todo61	3.35	11.68	8.33	31
todo62	1.49	5.94	4.45	23
todo63	2.07	9.11	7.04	22
todo71	0.29	3.99	3.7	35
todo72	1.75	6.52	4.77	23
todo73	1.82	5.38	3.56	40
todo91	2.64	5.76	3.12	44
todo92	2.59	8.89	6.3	39
todo93	2.17	6.73	4.56	42
todo101	-0.32	4.31	4.63	27
todo102	0.3	2.55	2.25	35
todo103	-2.12	-1.37	0.84	18

Tabla 9.2: Medidas de performance con el *pitch* calculado con *WaveSurfer*.

Tomando promedios entre los valores de las ganancias, se llega a un valor de 4,54dB. También promediando la comparación entre las máscaras se obtiene al valor de $\Delta M =$ 29,8. Los valores obtenidos para las ganancias en la tabla 9.2 son comparables con los valores presentados en [1], en dicho artículo los mismos están entre 0 dB y 12 dB. Cabe destacar que en [1] se realiza la medida de la performance sobre distintos fragmentos mezclados a diferentes SNR.

La tabla 9.3 muestra las medidas de performance realizadas para el caso 2, o sea, tomando como frecuencia fundamental en cada trama la hallada con el algoritmo de detección de f_0 presentado en este documento. Se obtienen para este caso las medidas de performance del sistema de punta a punta.

Archivo	SNR antes (dB)	SNR después (dB)	Ganancia (dB)	ΔM (%)
todo51	2.39	8.93	6.54	23
todo52	2.08	4.93	2.85	28
todo53	3.67	5.49	1.82	35
todo11c	8.56	11.95	3.39	24
todo12c	8.7	13.96	5.26	30
todo13c	2.68	2.47	-0.21	34
todo61	3.35	7.81	4.46	32
todo62	1.49	3.10	1.61	23
todo63	2.07	7.52	5.45	26
todo71	0.29	2.86	2.57	36
todo72	1.75	5.58	3.83	25
todo73	1.82	4.71	2.89	41
todo91	2.64	5.02	2.38	47
todo92	2.59	9.07	6.48	36
todo93	2.17	0.6	-1.57	54
todo101	-0.32	0.18	0.5	36
todo102	0.3	4.53	4.23	33
todo103	-2.12	-1.23	0.89	24

Tabla 9.3: Medidas de performance para el sistema de punta a punta.

En este caso la ganancia llega al valor promedio de $2{,}97dB$, y la comparación entre las máscaras da un porcentaje de error promedio de $\Delta M = 32{,}6\,\%$. Si bien el valor de la ganancia promedio es algo menor al obtenido en el caso 1, es comparable con los valores presentados en [1]. Una observación importantes es que en casi todos los casos se consigue una mejora en la SNR.

Con respecto a los resultados de inteligibilidad, vale la pena destacar que si bien se obtienen señales de audio que se escuchan correctamente, en todos los casos queda un ruido de fondo y distorsión. Otra observación importante es que las señales obtenidas utilizando el *pitch* obtenido con *WaveSurfer* se escuchan mejor. Esto era de esperarse, ya que el error al calcular el *pitch*, es menor al que se puede obtener con el algoritmo de detección de f_0.

A continuación se presenta una gráfica comparativa para las ganancias obtenidas en ambos casos, se muestran para los 18 fragmentos utilizados.

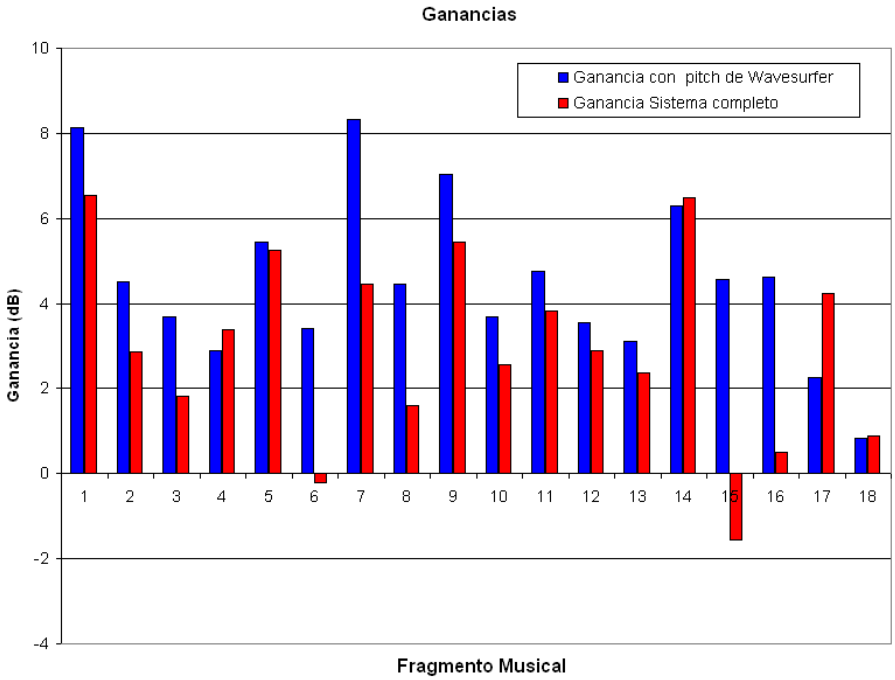

Figura 9.2: Ganancias obtenidas.

Puede observarse que si bien se obtienen mejores resultados con el *pitch* calculado con *WaveSurfer*, los resultados obtenidos al utilizar el algoritmo de detección de f_0 implementado también son buenos y son similares a los anteriores.

9.4. Ejemplo de punta a punta

En esta sección se presenta un ejemplo de principio a fin para un archivo en particular, mostrando en cada etapa los resultados obtenidos. Cabe destacar que se muestra un ejemplo para el caso de voz hablada con acompañamiento del tipo *cocktail party*, ya que algunos detalles importantes de las figuras que se presentan se pueden apreciar de mejor manera.

El archivo que se utiliza en este ejemplo puede encontrarse en la base de datos bajo el nombre de "v3n3.wav". También se utilizan para realizar comparaciones y evaluaciones los archivos que contienen las señales de voz y acompañamiento originales, los cuales se encuentra con los nombres "v3.wav" y "n3.wav" respectivamente.

El objetivo de este ejemplo es mostrar paso a paso los resultados del algoritmo, con el fin de integrar todos los resultados de cada bloque del sistema implementado.

a- Señal de entrada La figura 9.4 muestra la forma de onda de la señal de mezcla (voz hablada más interferencia del tipo *cocktail party*) y la forma de onda de la señal de sólo voz hablada (gráfica inferior).

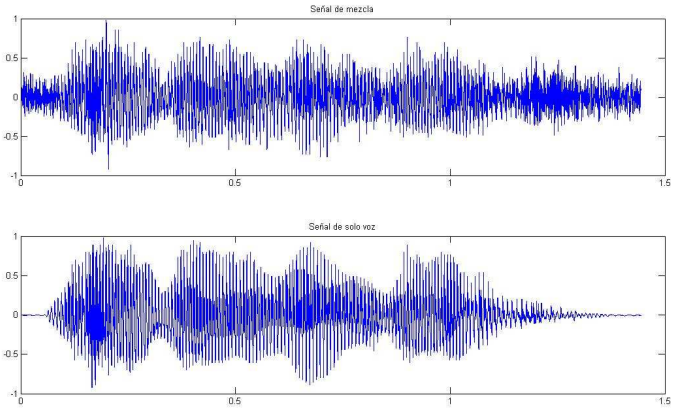

Figura 9.3: Formas de onda de la mezcla (superior) y de la señal de sólo voz hablada (inferior).

b- Filtros gammatone y Modelo de Meddis

Al estar trabajando con un archivo del que ya se posee todo por separado, es decir, se tiene por un lado la señal de mezcla, la señal de voz y la señal de interferencia, se puede construir fácilmente la máscara ideal binaria (IBM). La siguiente figura muestra el cocleagrama de la señal de mezcla (i), el cocleagrama de la señal de interferencia (ii) y el cocleagrama de la señal de sólo voz (iii). También se muestra la correspondiente máscara ideal binaria (iv).

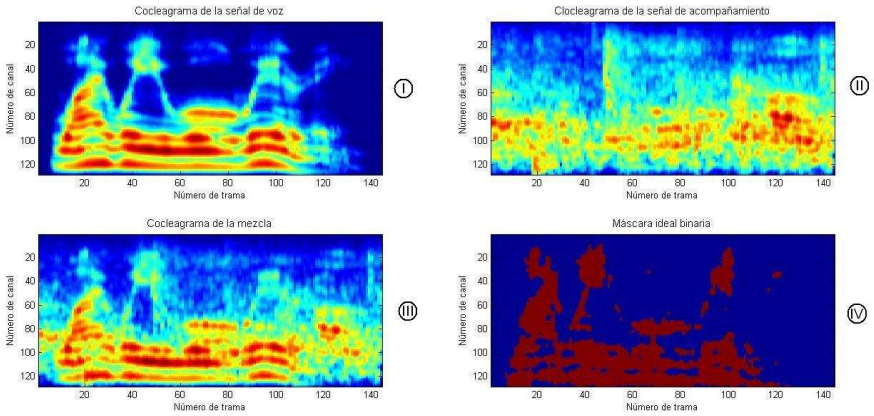

Figura 9.4: Cocleagramas y máscara ideal binaria.

Como se mencionó anteriormente, el objetivo computacional del sistema es obtener una máscara binaria lo más similar a la IBM. En la figura 9.4 puede observarse que el cocleagrama de la señal mezcla, es la superposición del cocleagrama de la señal de voz y el cocleagrama de la señal de interferencia. El objetivo del sistema es identificar aquellas unidades T-F del cocleagrama de la señal de mezcla, en donde la voz predomine por sobre la interferencia. Puede observarse que si se enmascara

al cocleagrama de la señal de mezcla con la IBM se obtiene el cocleagrama de la señal de voz.

c- Detección del *pitch* predominante
A continuación se muestra el *pitch* obtenido con el algoritmo de detección de *pitch* implementado.

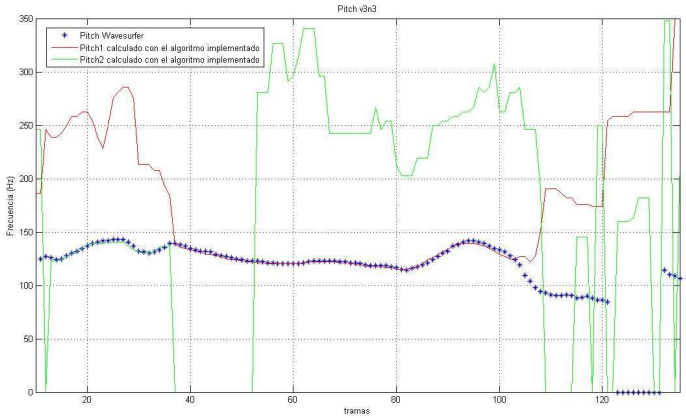

Figura 9.5: *Pitch* obtenido.

En la figura 9.4 se muestra en magenta el *pitch* obtenido con el algoritmo implementado, y con cruces el *pitch* obtenido con *WaveSurfer*. El resultado en este caso es muy bueno en la parte central, pero tanto en el principio como al final del fragmento se aprecian algunos errores importantes. Para este ejemplo particular, el error en la detección del *pitch* fue de un 50 % comparando contra *WaveSurfer*.

d- Segmentación
En esta parte se presenta el proceso de marcado de unidades y la formación de los segmentos.

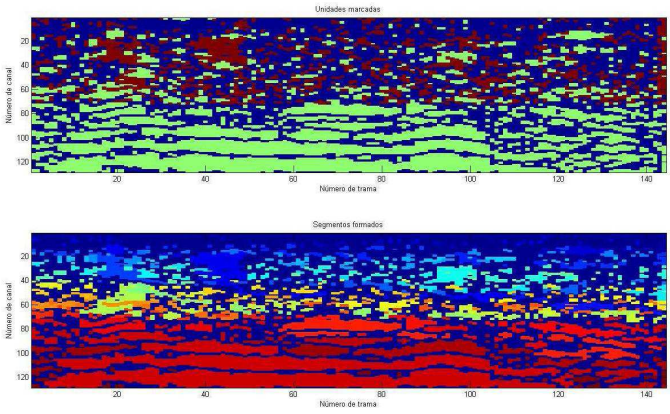

Figura 9.6: Marcado de unidades y formación de segmentos.

La figura 9.4 muestra en la parte superior el marcado de unidades previo a la formación de segmentos. En verde las unidades marcadas con 1 que son aquellas que tienen alta correlación cruzada, y en rojo las unidades marcadas con 2 que son las unidades que tienen alta correlación cruzada entre las envolvente de los canales. En la parte inferior se muestran en diferentes colores los segmentos que se formaron.

e- Agrupamiento

La siguiente figura muestra las unidades marcadas, y luego en la parte inferior se pueden ver en rojo las unidades que fueron etiquetadas como voz dominante.

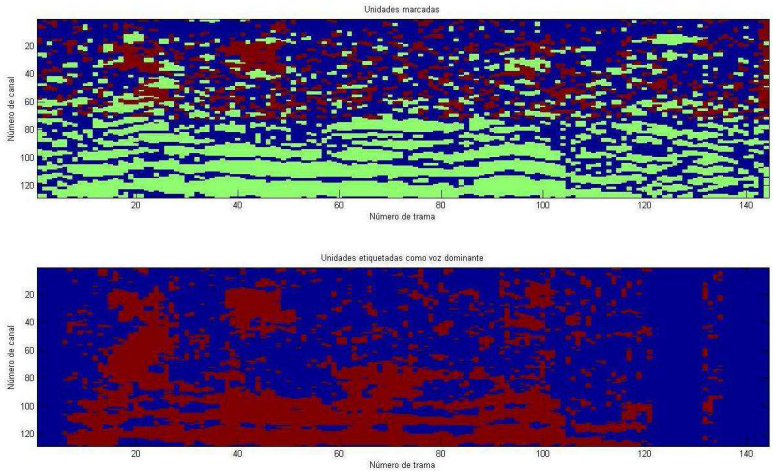

Figura 9.7: Unidades marcadas y unidades etiquetadas.

En la figura 9.4 se muestra el *stream* final y la máscara ideal binaria.

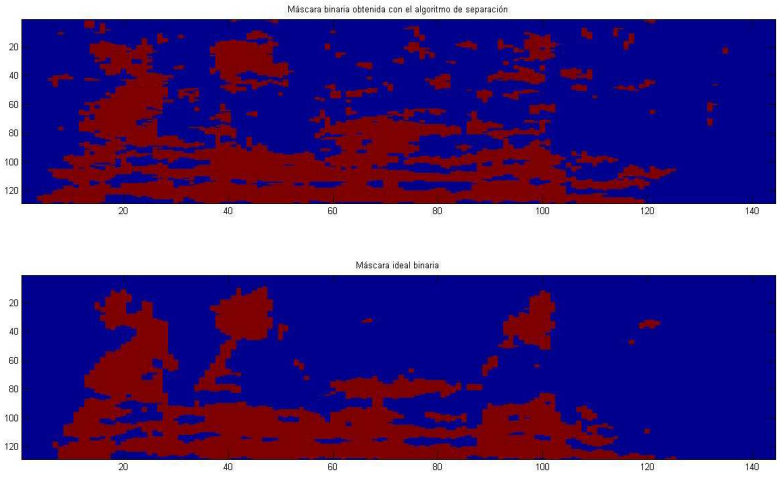

Figura 9.8: *Stream* final obtenido y máscara ideal binaria.

En la figura 9.4 puede observarse la similitud entre el *stream* final obtenido y la IBM.

f- Resíntesis
Por último, se llega a la etapa final del algoritmo, en la cual se reconstruye la señal de voz en el dominio del tiempo.

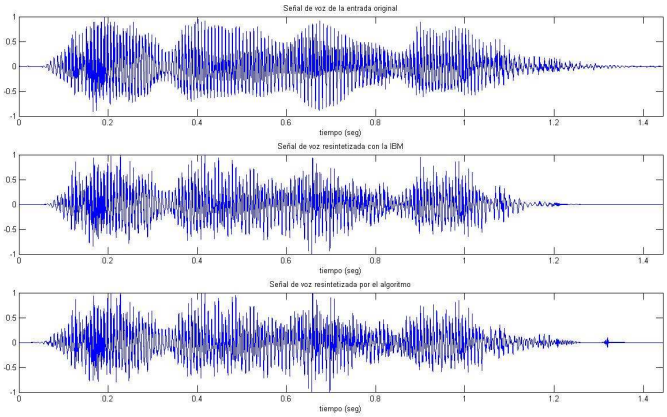

Figura 9.9: Formas de onda obtenidas.

En la figura 9.4 se observan las diferentes formas de onda de la señal de voz. En la parte superior se observa la forma de onda de la señal de voz hablada original. En la parte central está la forma de onda obtenida al resintetizar con la máscara ideal, y en la parte inferior la forma de onda obtenida al resintetizar con la máscara obtenida por el algoritmo. Como se mencionó anteriormente, el objetivo computacional del sistema es obtener una máscara binaria (*stream*) lo más similar posible a la máscara ideal binaria. En la figura 9.4 se puede ver que la forma de onda original y la obtenida con la IBM son similares. Asimismo puede notarse que las formas de onda obtenidas con la IBM y con algoritmo implementado también son muy parecidas, por lo que el resultado obtenido en este ejemplo es muy bueno.[2].

9.5. Conclusiones

Los resultados obtenidos mediante el código implementado son exitosos, estando a la par de los algoritmos de separación de voz cantada existentes, [1] y [49] por ejemplo.

Como se mencionó anteriormente el bloque de detección de *pitch* es desde el punto de vista de desarrollo equivalente al resto del algoritmo, siendo computacionalmente la etapa más costosa. La evaluación del algoritmo de separación de voz cantada aquí presentado se realiza en dos escenarios diferentes, teniendo en cuenta este hecho. Se realiza una evaluación utilizando el *pitch* calculado con *WaveSurfer* por un lado y luego utilizando el bloque de detección de f_0 implementado, observándose que el resultado final

[2]En la base de datos se encuentran disponibles todos los archivos para este ejemplo

del sistema depende fuertemente de la detección de *pitch*. Asimismo, también se realiza por separado la evaluación del método de detección de *pitch*. De los resultados de las evaluaciones anteriores, se puede afirmar que, si se impone que el *pitch* sea una entrada al sistema implementado los resultados obtenidos son muy buenos, concluyendo por lo tanto, que el método de separación funciona de manera aceptable. Cuando se utiliza la función de detección de f_0 implementada los resultados obtenidos también son exitosos, pero existe una mayor dependencia de las características de la señal de entrada, incurriendo en mayores errores en general. Cuando la señal de entrada no presenta un porcentaje elevado de sonidos sordos y no presenta largos silencios, el *pitch* obtenido es muy bueno. Es importante destacar nuevamente, que el *pitch* detectado por el algoritmo presentado en este documento, detecta el *pitch* para la voz cantada a partir de la señal de mezcla, voz cantada y acompañamiento musical, a diferencia de *WaveSurfer* que lo calcula sólo sobre la señal de voz cantada.

Finalmente al comparar los resultados obtenidos contra la IBM, la cual es el objetivo computacional, también se obtienen resultados aceptables. En el caso de evaluar al sistema utilizando el *pitch* como una entrada más al sistema, se comete un error de 29,8 % en promedio, mientras que si se evalúa al sistema utilizando el método de detección de *pitch* implementado, se comete un error de 32,6 % en promedio. Cabe destacar que como se mencionó en el capítulo 7, el error que se comete en el método de detección de *pitch* es de 30 % en promedio. Si bien estos valores pueden parecer elevados, son comparables con los obtenidos en [1].

Capítulo 10

Conclusiones

Este capítulo final, expone conclusiones generales sobre el sistema de separación de voz cantada implementado, así como también una evaluación de la experiencia de realizar este proyecto de fin de carrera.

Respecto al método seleccionado para la resolución del problema planteado [1], se obtuvieron resultados exitosos. En el capítulo 9 se presentaron dos métodos para evaluar el desempeño del sistema implementado. El primero consiste en evaluar al sistema utilizando el *pitch* como una entrada más al sistema. Por otro lado el segundo consiste en evaluar al sistema utilizando la función de detección de f_0 implementada. De este modo, luego de realizar ambas evaluaciones, se desprende que el resultado final del proceso de separación de voz cantada depende fundamentalmente, como se explicó en el capítulo 5, de la detección del *pitch*. Cabe recordar, que el problema de obtener el *pitch* de la voz cantada, a partir de una canción, es en sí, desde el punto de vista conceptual y de implementación un problema de proporciones similares al problema abordado. Esto último, justifica la evaluación del sistema de dos maneras diferentes.

De la primera evaluación se deduce que las etapas del modelo auditivo, extracción de características, segmentación, agrupamiento y resíntesis, capítulos 3, 4, 6, 7 y 8 respectivamente, funcionan correctamente, dando muy buenos resultados. Es importante aclarar, que trabajar de este modo, reduce considerablemente los tiempos de ejecución del algoritmo de separación de voz cantada. Ya que un 60 % del tiempo de ejecución se dedica a la función de detección de *pitch*.

Como se mencionó desde un comienzo, todo el algoritmo implementado está pensado para sonidos sonoros. De todos modos, como los mismos aparecen en un porcentaje mucho mayor a los sonidos sordos, y teniendo en cuenta que los fragmentos utilizados fueron de corta duración (2,5 segundos), la presencia de sonidos sordos en los fragmentos musicales no tuvo gran influencia en los resultados finales.

En el capítulo 5 se presentó un método de evaluación para la función de detección de f_0, el mismo se basó en comparar el *pitch* calculado con el algoritmo implementado contra el *pitch* obtenido con *WaveSurfer*. En dicha comparación se obtuvo un valor de error relativo de 30 %, llegando a errores menores al 10 % en algunos casos.

Si se considera la inteligibilidad de la señal separada como criterio de éxito, es decir, si al escuchar la señal separada se entiende la letra de la canción, se obtuvo aproximadamente un 80 % de resultados exitosos.

A pesar de lo mencionado anteriormente, actualmente el método implementado dista mucho de ser un método recomendado para la mayoría de las posibles utilidades mencionadas en el capítulo 1, debido a los altos costos computacionales. Actualmente, procesar un segmento de 2,5 segundos insume aproximadamente 40 minutos de procesamiento. Cabe destacar, que si bien los tiempos de procesamiento son elevados, el hecho de trabajar con fragmentos de corta duración era de esperarse, ya que al utilizar una frecuencia de muestreo de 16 kHz implica 16000 muestras por segundo, además luego de utilizar el banco de filtros auditivos, se obtienen matrices de 128 filas por la cantidad de muestras. Esta es la principal limitante para trabajar con fragmentos de mayor tamaño.

En el código realizado por Hu-Wang, que implementa el algoritmo presentado en el artículo [16], se pone una limitante sobre el número máximo de muestras de 200000. Este valor corresponde a una duración maxima de 12,5 segundos trabajando a 16 kHz para las señales de audio de entrada. En particular en los ejemplos que se incluyen junto con el código de [16] los fragmentos son de un segundo de duración.

Quedó pendiente el pasaje a otra plataforma de trabajo. En un principio se pensó en pasar el sistema implementado en Matlab a $C++$, luego de estudiar diferentes alternativas, finalmente se comenzó a implementar la solución en *Java* y *C*. Desafortunadamente, los tiempos no alcanzaron para culminar esta tarea. El pasaje a otra plataforma hubiera sido importante, para obtener menores tiempos de procesamiento y para tener una aplicación multiplataforma (en caso de haber escrito todo en *C* por ejemplo).

Finalmente se cita una frase de Cherry escrita hace mucho tiempo atrás en [8] y citada en [11] :

One of our most important faculties is our ability to listen to, and follow, one speaker in the presence of others ... we may call it "the cocktail party problem". No machine has yet been constructed to do just that. ([8] p. 280)

Si bien la frase es referida al problema de *cocktail party* es perfectamente ajustable al problema abordado en este proyecto. Actualmente se puede afirmar que se está muy cerca de construir una máquina o sistema, que sea capaz de seguir a una fuente (voz cantada), en presencia de otras fuentes interferentes (acompañamiento musical). Y este proyecto de fin de carrera, es un ejemplo de lo anterior.

10.1. Trabajo futuro

Como trabajo futuro quedan pendientes varios puntos a mejorar del algoritmo implementado.

- Optimización del algoritmo
  - Optimizar la función de autocorrelación. Ésta puede implementarse utilizando la transformada discreta de Fourier (*fft* en Matlab), de este modo se podrían disminuir los tiempos de ejecución, ya que dicha función implica un gran costo computacional y es una de las limitantes para introducir fragmentos de mayor duración al sistema implementado.

  - Optimizar el algoritmo de detección de *pitch*. Éste es el que insume mayor tiempo y costo computacional, podría estudiarse la posibilidad de implementar algunas de sus partes de manera más óptima, por ejemplo el *Pitch Tracking* que es la parte más larga del algoritmo.

- Mejoras al algoritmo

  - Una mejora importante sería incluir las funciones necesarias para manejar los sonidos sordos. Se recuerda que si bien son un porcentaje menor que los sonoros (10 % frente a un 90 % de sonidos sonoros), el algoritmo implementado, trata a ambos sonidos por igual, cometiendo errores al encontrase frente a un sonido sordo. Recientemente se publicó un artículo que trata sobre la separación de los sonidos sordas [50].
  - Mejora de la función de detección de *Pitch*. Esta función es de vital importancia para obtener buenos resultados al final del sistema. Se podría por ejemplo reestimar los valores de las probabilidades de transición que fueron tomados de [1], utilizando para ello una nueva base de datos. Otra mejora a esta función sería agregar un algoritmo de corrección para corregir el problema de error en la detección de f_0, calculándose erráticamente $2f_0$ o $f_0/2$. Otra posibilidad sería encontrar o desarrollar algún otro método de obtención de *pitch* con un mejor desempeño y que contemple los puntos anteriormente descritos.
  - En la etapa final del algoritmo se podría implementar algún método de mejora para la señal obtenida luego de la resíntesis. Si bien luego de normalizarla y filtrarla se obtiene una señal entendible, en general suena "poco natural".
  - Incluir el bloque de detección vocal/no-vocal. Este primer bloque es de vital importancia para el correcto funcionamiento del sistema implementado. Debido a la limitante de poder usar solamente fragmentos de corta duración se optó por descartar la implementación del mismo, ya que al utilizar fragmentos cortos se pueden elegir fácilmente aquellos que incluyan voz cantada y acompañamiento musical todo el tiempo.

- Pasaje del algoritmo a otra plataforma

  - En un principio se estudió la posibilidad de realizar el pasaje del sistema implementado en Matlab a *C* o *C++*. Luego se estudió la posibilidad de realizar el pasaje a *Java* combinado con *C*. Si bien no se pudo llevar a cabo esta tarea en tiempo y forma, se presentan en el apéndice I algunas ideas adquiridas para un futuro trabajo.

10.2. ¿Qué nos dejo este proyecto?

Si bien se aprendió mucho sobre el área del tratamiento de señales, toda la primera etapa de planeamiento, organización, trabajo en equipo y división de tareas dejó un gran aporte a cada uno de los integrantes del grupo. Se aprendió a planificar con más

detalle las tareas a realizar y lo importante que es tomar en cuenta la mayor cantidad de variables en el momento de la planificación (por ejemplo viajes, licencias, enfermedad).

La etapa de documentación del proyecto fue una ardua tarea. En dicha tarea, se percibió una gran falta de entrenamiento para la formulación, desarrollo y estructuramiento de documentos técnicos. Cabe destacar que la misma se facilitó considerablemente al conocer el editor de textos LaTeX, el cual permitió elaborar el documento de una manera diferente. Para su aprendizaje se consultó el manual [51].

Apéndice A

Voz y Sistema Auditivo

En este capítulo se introducen conceptos relacionados con la voz y el sistema auditivo humano. Es importante tener una breve descripción de ellos ya que son la base teórica de muchos de los temas tratados en este documento.

A.1. Generación de la voz

En esta sección se da una introducción al tema de la generación de la voz humana. Se presenta el mecanismo de producción de voz y se muestra una clasificación de los sonidos en sonoros y sordos. Finalmente se introduce el modelo lineal con el cual se intenta aproximar el mecanismo de producción de voz.

A.1.1. Mecanismo de producción de voz

La voz humana se produce voluntariamente por medio del aparato fonatorio [52]. Éste está formado por los pulmones como fuente de energía en la forma de un flujo de aire, la laringe, que contiene las cuerdas vocales, la faringe, las cavidades oral (o bucal) y nasal y una serie de elementos articulatorios: los labios, los dientes, el alvéolo, el paladar, el velo del paladar y la lengua. En la figura A.1 se muestra un diagrama del aparato fonatorio.

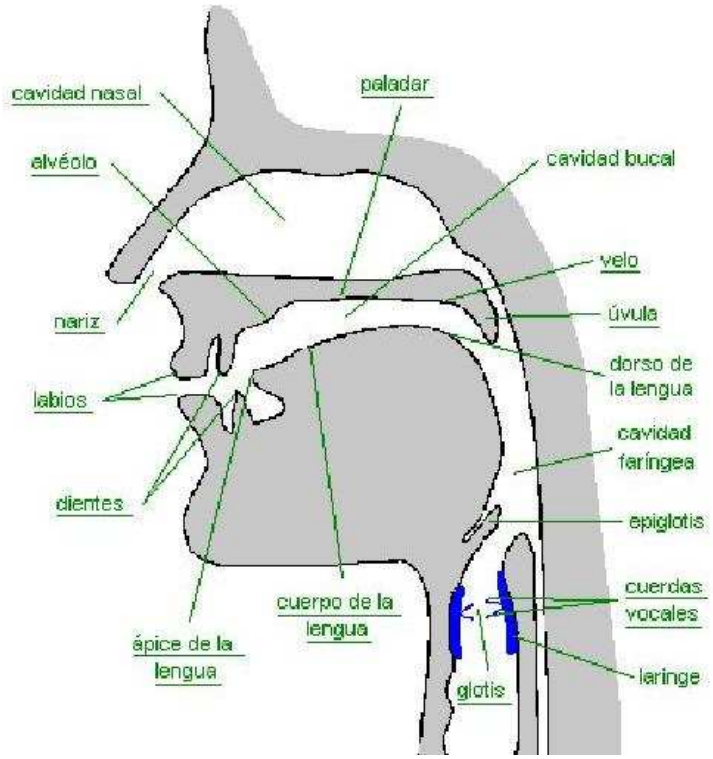

Figura A.1: Corte esquemático del aparato fonatorio

Las cuerdas vocales son dos membranas dentro de la laringe orientadas de adelante hacia atrás. Por adelante se unen en el cartílago tiroides. Por detrás, cada una está sujeta a uno de los dos cartílagos aritenoides, los cuales pueden separarse voluntariamente por medio de músculos. La abertura entre ambas cuerdas se denomina glotis. En la figura A.2 se puede ver un diagrama de la laringe.

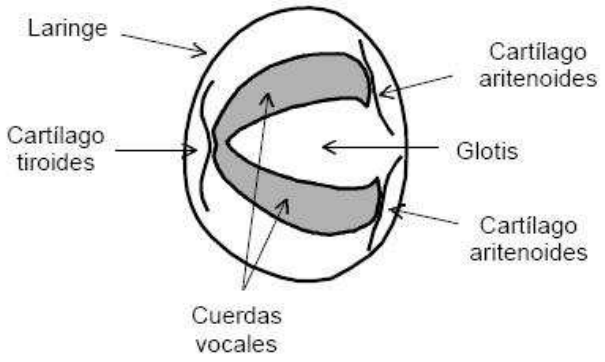

Figura A.2: Corte esquemático de la laringe

Cuando las cuerdas vocales se encuentran separadas, la glotis adopta una forma triangular. El aire pasa libremente y prácticamente no se produce sonido. Es el caso de la

respiración. Cuando la glotis comienza a cerrarse, el aire que la atraviesa proveniente de los pulmones experimenta una turbulencia, emitiéndose un ruido de origen aerodinámico conocido como aspiración. Al cerrarse más, las cuerdas vocales comienzan a vibrar a modo de lengüetas, produciéndose un sonido tonal, es decir periódico. La frecuencia de este sonido depende de varios factores, entre otros del tamaño y la masa de las cuerdas vocales, de la tensión que se les aplique y de la velocidad del flujo del aire proveniente de los pulmones.

Luego de que el aire atraviesa las cuerdas vocales se produce una modulación en el tracto vocal. Se puede considerar al tracto vocal como tubo de sección variable, controlado según la articulación determinada por los músculos de la faringe, el cuerpo y punta de la lengua, el velo del paladar y los labios. Al igual que en un tubo simple, el fenómeno de resonancias en el tracto aumentan la amplitud de un grupo de frecuencias alrededor de una determinada banda de frecuencia. A cada resonancia se le denomina formante.

A.1.2. Clasificación de los sonidos

Los sonidos emitidos por el aparato fonatorio pueden clasificarse de distintas maneras, según se tengan en cuenta los diferentes aspectos del fenómeno de producción. Una clasificación importante y que es necesaria conocer para la comprensión de este trabajo, es la de considerar dos grupos o categorías, los sonidos sonoros (*voiced*) y los sonidos sordos (*unvoiced*).

Los sonidos sonoros se producen por la vibración de las cuerdas vocales. Por otro lado, los sonidos sordos se generan como turbulencias de aire en alguna constricción del tracto vocal. A continuación se listan algunas características sobresalientes de estos sonidos.

Sonidos sonoros:

- Forma de onda similar a la salida de un filtro excitado con un tren de pulsos. Es aproximadamente periódica.
- Su espectro tiene componentes discretos que están en relación armónica, es decir, son múltiplos enteros de una frecuencia fundamental f_0 ($f_k = kf_0$).
- Tiene alta energía.
- La mayoría de los sonidos caen en esta categoría, por ejemplo las vocales y n,m y $\tilde{n}$.

La figura A.3 muestra un sonido de este tipo.

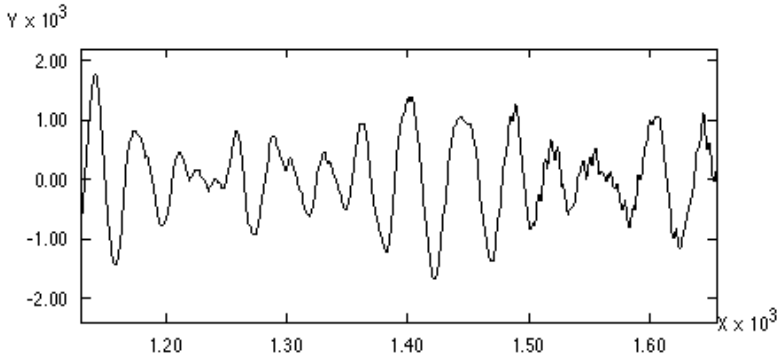

Figura A.3: Forma de onda para un sonido sonoro (*voiced*)

En la figura A.3 puede verse que hay una fuerte tendencia de forma de onda periódica.

Sonidos sordos:

- Forma de onda aleatoria pero de espectro acotado. Tiene aspecto ruidoso, es no periódica.
- Tiene baja energía.
- A este grupo pertenecen por ejemplo *sh*, *f* y *s*.

En la figura A.4 se muestra la forma de onda de uno de estos sonidos.

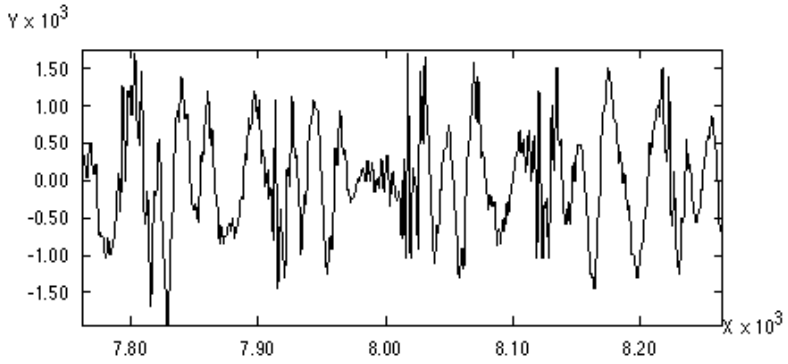

Figura A.4: Forma de onda para un sonido sordo (*unvoiced*)

La figura A.4 muestra claramente que la forma de onda para los sonidos sordos tiende a ser ruidosa.

Las diferencias entre estos dos tipos de sonidos lleva a identificarlos y tratarlos a cada uno de manera diferente.

A.1.3. Modelo de producción de voz

El mecanismo de producción del habla puede ser descrito como una fuente de energía periódica o aleatoria excitando un filtro no lineal [9]. En la práctica se aproxima a un modelo lineal.

El modelo lineal consiste en una fuente generadora de señales, ya sea un tren periódico de pulsos para los sonidos sonoros, o una fuente de ruido para los sonidos sordos. La salida de las fuentes es seguida por un filtro lineal que modela el tracto vocal, en su forma más general es un conjunto de ceros y polos. La forma de onda de la voz es por tanto considerada como una convolución entre una fuente de excitación y la respuesta de un filtro lineal. La figura A.5 muestra este modelo lineal.

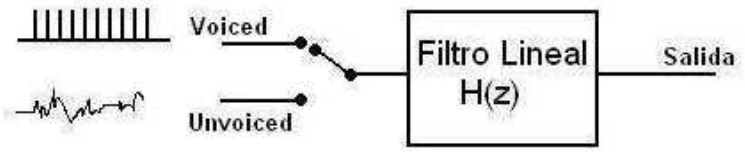

Figura A.5: Modelo lineal simplificado de generación del habla.

La ecuación del filtro viene dada por:

$$H(z) = \frac{\sum_{i=0}^{p} a_i z^{-i}}{1 + \sum_{j=1}^{q} b_j z^{-j}} \tag{A.1}$$

A partir de este modelo, se pueden estudiar o abordar diferentes tipos de problemas. Además de la separación de voz, hay dos problemas que se estudian comúnmente. Uno es el reconocimiento del habla, el cual implica obtener la representación textual de una señal de voz. El otro es la conversión de texto a voz. La figura A.6 muestra un esquema de ambos.

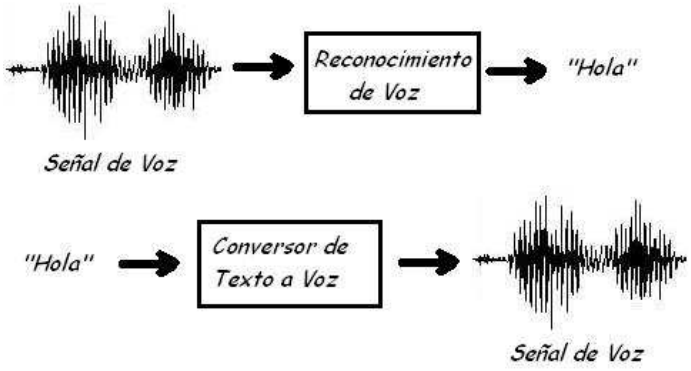

Figura A.6: Esquema de dos de los problemas que se estudian a partir del modelo de generación del habla.

A.2. Sistema auditivo humano

En esta sección se presentan algunas características del sistema auditivo humano, con el fin de entender mejor la habilidad de los seres humanos para diferenciar sonidos provenientes de diferentes fuentes. Es importante conocer el modelo de percepción del sonido, ya que el algoritmo implementado se basa en una representación del sistema auditivo.

La función de nuestro sistema auditivo es, esencialmente, transformar las variaciones de presión originadas por la propagación de las ondas sonoras en el aire en impulsos eléctricos (variaciones de potencial), información que los nervios acústicos transmiten a nuestro cerebro para la asignación de significados [53].

El principal órgano es el oído. Está compuesto por tres estructuras: el oído externo, el oído medio y el oído interno. En la figura A.7 se muestra un diagrama del sistema auditivo.

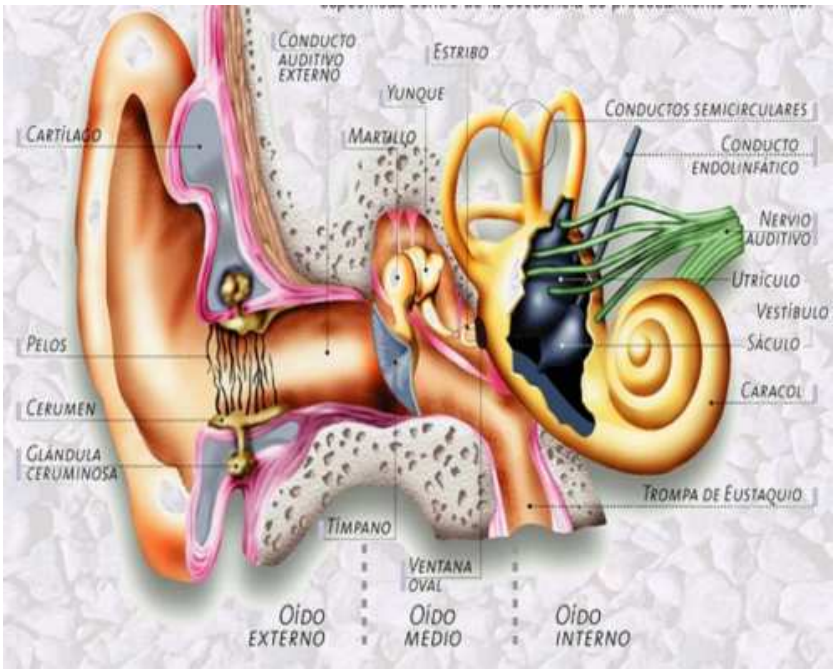

Figura A.7: Sistema auditivo [54].

El oído externo está compuesto por la oreja y el conducto auditivo externo. La oreja concentra las ondas sonoras en el conducto, el cual desemboca en el tímpano. La función del oído externo es la de recolectar las ondas sonoras y encauzarlas hacia el oído medio. Asimismo, el conducto auditivo tiene dos propósitos adicionales: proteger las delicadas estructuras del oído medio contra daños y minimizar la distancia del oído interno al cerebro, reduciendo el tiempo de propagación de los impulsos nerviosos.

El oído medio está lleno de aire y está compuesto por el tímpano (que separa el oído externo del oído medio), los osículos (martillo, yunque y estribo, una cadena ósea denominada así a partir de sus formas) y la trompa de Eustaquio. El tímpano es una membrana que es puesta en movimiento por la onda que la alcanza. Sólo una parte de la onda que llega al tímpano es absorbida, la otra es reflejada. Se llama impedancia acústica a esa tendencia del sistema auditivo a oponerse al pasaje del sonido. Su magnitud depende de la masa y elasticidad del tímpano y de los osículos y la resistencia friccional que ofrecen. Los osículos tienen como función transmitir el movimiento del tímpano al oído interno a través de la membrana conocida como ventana oval.

Si en el oído externo se canaliza la energía acústica y en el oído medio se la transforma en energía mecánica transmitiéndola -y amplificándola- hasta el oído interno, es en éste en donde se realiza la definitiva transformación en impulsos eléctricos. El laberinto óseo es una cavidad en el hueso temporal que contiene el vestíbulo, los canales semicirculares y la cóclea (o caracol). Dentro del laberinto óseo se encuentra el laberinto membranoso, compuesto por el sáculo y el utrículo (dentro del vestíbulo), los ductos semicirculares y el ducto coclear. Este último es el único que cumple una función en la audición, mientras que los otros se desempeñan en nuestro sentido del equilibrio. La cóclea es un conducto

casi circular enrollado en espiral (de ahí su nombre) unas 2,75 veces sobre sí mismo, de unos 35 mm de largo y unos 1,5 mm de diámetro como promedio. La cóclea se comporta como un banco de filtros pasabanda, donde cada uno de ellos está centrado en una frecuencia particular. Se divide entonces el espectro auditivo en porciones que indican el rango para que un sonido active una determinada porción de la membrana basilar, la cual está en comunicación con las neuronas auditivas que conducen los estímulos al cerebro. Es importante saber que la resolución en frecuencia del oído humano no es uniforme y se lo describe usualmente en base a las llamadas bandas críticas.[1]

A.3. Voz hablada y voz cantada

En esta sección se muestran algunas diferencias y similitudes entre la voz hablada y la voz cantada. Las similitudes entre ambas llevan en principio a explorar el tema de separación de voz hablada, con el fin de tomar ideas y enfoques para resolver el problema de separación de voz cantada. Sin embargo, existen una serie de diferencias significativas que hacen que el problema de separación de voz cantada sea diferente del problema de separación de voz hablada.

La voz cantada y la voz hablada comparten características comunes debido a que son generadas por el mismo mecanismo, además ambas están compuestas por sonidos sordos y sonoros. A continuación se listan algunas diferencias a tener en cuenta.

- En la voz cantada, para un cantante de ópera, aparece un formante adicional ubicado en el rango de frecuencia de 2000-3000 Hz. Este formante es el que da a la voz del cantante la ayuda necesaria para sobresalir del acompañamiento. Sin embargo, no existe en todos los géneros musicales, por ejemplo en los estilos musicales de Pop, Rock y Country este formante no aparece.

- En general en la voz cantada el cantante esfuerza su voz para que se ajuste a la melodía del acompañamiento, esto tiene dos consecuencias directas:

  - Altera el porcentaje de sonidos sonoros/sordos. Para la voz hablada, los sonidos sonoros están presentes alrededor del 60 % del tiempo, los sonidos sordos el 25 % y el silencio el 15 %. En la voz cantada, el porcentaje de sonidos sonoros llega al 95 % (en el caso de la ópera), ya que es a través de la tonalidad que se transmite la mayor parte del contenido musical de una partitura de voz [55]. En los demás estilos musicales el porcentaje de sonidos sonoros llega al 90 %.
  - Varía la dinámica de la frecuencia fundamental de la voz. En el caso de la voz cantada, tiende a ser constante con cambios abruptos en el medio. Además los rangos de variación para la voz cantada son mayores que en el caso de la voz hablada. Por ejemplo, para la voz hablada el rango de variación es de 80 a 400 Hz, mientras que para la voz cantada puede ser de hasta 1400 Hz para un soprano [1].

[1] Por más información ver el apéndice C.

A.4. Conclusiones

En este capítulo se ha hecho una breve descripción sobre la generación de la voz, se mostró el mecanismo de producción, la clasificación de los sonidos en sonoros y sordos y el modelo de producción. Tales temas son de gran importancia en la mayoría de los problemas relacionados con el procesamiento de señales de audio.

Asimismo se dio también una introducción al sistema auditivo humano, cuyo conocimiento ayuda a comprender la base teórica del algoritmo implementado.

Finalmente se mostraron algunas similitudes y diferencias entre la voz hablada y la voz cantada.

Apéndice B

Estado del Arte del Problema

Este capítulo introduce los distintos enfoques existentes para resolver el problema de la separación de la voz cantada en una grabación musical.

B.1. Introducción

Para atacar el problema de la separación de la voz cantada del acompañamiento musical, básicamente se encontraron dos enfoques distintos: uno basado en la representación computacional del sistema auditivo humano (CASA) y el otro basado en métodos estadísticos.

Dentro del primer enfoque, CASA, se encuentra el artículo de Li-Wang [1].

Con respecto a los métodos estadísticos existen 3 artículos que plantean posibles soluciones para el problema de la extracción de la voz cantada en una canción. Estos artículos son [56], [57] y [58]. También existe el artículo [49], de reciente publicación. Este último artículo es de los mismos autores de [58] y es justamente su continuación. En [49] se profundizan los conceptos presentados inicialmente en [58] y se ofrece una mejor explicación del contenido.

Luego de la etapa de búsqueda, se eligieron dos artículos representativos de cada enfoque para estudiarlos con mayor profundidad, con el fin de decidir que solución implementar. Dentro del enfoque estadístico se optó por el artículo [49], y por el enfoque CASA se escogió el artículo [1].

Finalmente, luego de estudiar ambos enfoques y evaluar la viabilidad de la implementación de cada uno, se decidió implementar el artículo [1], ya que a priori, presentaba mejores resultados y mayor cantidad de información disponible.

En las próximas secciones de este capítulo, se comentan brevemente los conceptos principales de los artículos [56] y [57], además se desarrolla con más detalle [49].

B.2. Separation of vocals from polyphonic audio recordings

En este artículo se utilizan técnicas estadísticas como PCA (*Principal Component Analysis*), junto con ICA (*Independent Component Analysis*) o NMF (*Non-negative Matrix Factorization*).
La siguiente figura muestra el diagrama de bloques propuesto en [56].

Figura B.1: Diagrama de bloques de [56].

El primer paso es segmentar la entrada en partes vocales y no-vocales.[1] Luego, a las partes vocales se les calcula la STFT (*Short Time Fourier Transform*) y se obtiene una matriz, la cual está conformada por filas que contienen los números de las tramas temporales, y por columnas que contienen los canales de frecuencia. Esta matriz se usa como entrada en el bloque de separación de las fuentes. Allí primero se utiliza PCA para reducir las redundancia, luego se utiliza ICA o NMF para estudiar las características de la señal.

Lo que se logra como salida es una serie de señales, las cuales representan fuentes estacionarias e independientes entre sí. Como la voz es no estacionaria, está conformada por una combinación de alguna de éstas señales, mientras que las restantes pertenecen al acompañamiento musical.

Lo que se intenta resolver en este artículo es la tarea final de reconstruir la voz a partir de las señales anteriores. Se propone para ello dos métodos, uno es reutilizar el bloque de segmentación vocal y no-vocal para identificar el grupo de señales pertenecientes a la voz, y el otro utilizando extracción de características como MFCC (*Mel Frequency Cepstral Coefficients*), PLP (*Perceptual Linear Prediction*) y LFPC (*Log Frequency Power Coefficients*).

A través de un estudio experimental, se concluye (en el mismo artículo) que los resultados obtenidos no fueron buenos, ya que la tarea del último bloque no pudo ser llevada a cabo de buena manera. De todas formas se propone como estudio a futuro utilizar un método llamado ICA no estacionario.

Respecto a este artículo, se concluye que el método está en fase de desarrollo, por lo que fue descartada su implementación.

[1] Vocal: parte de la canción que contiene voz cantada, no-vocal: parte de la canción donde sólo hay acompañamiento musical.

B.3. Separating a foreground singer from background music

Este artículo se basa en un modelo estadístico de variable latente. La idea principal es modelar el par tiempo-frecuencia (t, f) como una variable aleatoria, y el valor del espectro en (t, f) es igual al número de veces en las que el par (t, f) fue elegido desde una distribución bivariante $P(t, f)$. La distribución $P(t, f)$ representa la distribución conjunta de la variable aleatoria t y de la variable aleatoria f. La descomposición de estas variables se hace a través del nombrado modelo de variable latente:

$$P(x) = \sum_z P(z)P(t|z)P(f|z) \tag{B.1}$$

donde z representa una variable latente o no vista discreta. $P(f|z)$ representa una distribución marginal en frecuencia, donde sus elementos son distintos estados sonoros. $P(t|z)$ es una distribución marginal temporal, cuyos elementos representan los pesos de la distribución anterior, es decir, nos muestra cuando están presentes los distintos estados sonoros. En el caso de tener una mezcla de señales de voz y acompañamiento musical, el modelo queda:

$$P_{total}(t, f) = P(m) \sum_z P_m(z)P_m(t|z)P_m(f|z) + P(v) \sum_z P_v(z)P_v(t|z)P_v(f|z) \tag{B.2}$$

donde $P(m)$ es la probabilidad de que la porción de espectro pertenezca al acompañamiento musical, y $P(v)$ a la voz.

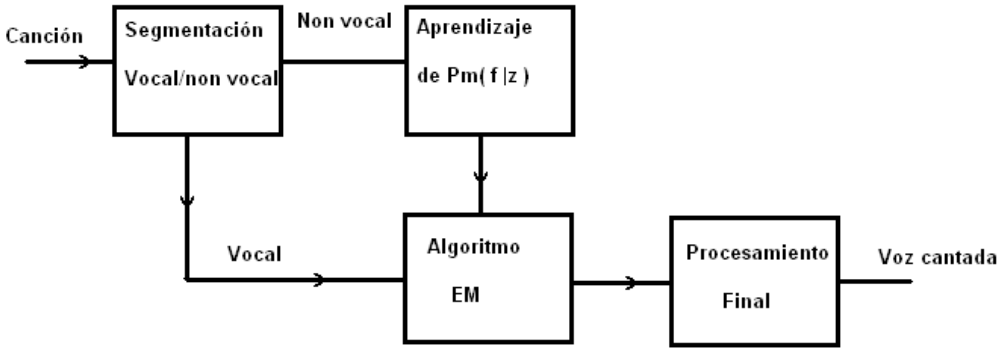

Figura B.2: Diagrama de bloques de [57]

Nuevamente aquí el primer paso es segmentar la entrada en partes vocales, donde esté presente la voz cantada y partes no-vocales donde esté sólo el acompañamiento musical. Luego a partir de las partes no-vocales se aprende el modelo $P_m(f|z)$. Todos los demás términos son aprendidos a través del algoritmo EM (*Expectation–Maximization*),

donde a través de ecuaciones iterativas se aprenden los modelos. Esto se hace en las partes donde está presente la voz cantada.

Una vez que están aprendidos todos los términos del modelo, se realiza una estimación del espectro de cada una de las señales, para luego antitransformar y obtenerlas en el dominio temporal. Un punto negativo en este artículo es que no se habla de la forma de la distribución $P(f|z)$, sin embargo en el artículo [59], se presenta el mismo modelo y se dice que dicha distribución es multinomial. Otro punto en contra es que tampoco se comenta sobre la elección del valor de la variable latente z, solamente hay una mención práctica en donde se elige el valor de 100, pero sin ningún fundamento. Haciendo una comparación con [49], sobre el cual se hablará en la siguiente sección, en este último se aprende un modelo general de la voz utilizando una base de datos genérica, y luego éste se adapta a las características de la grabación en cuestión. En [57] se aprende el modelo de voz desde la misma grabación, pero sin ninguna consideración previa. Además este artículo apunta a la personalización o modificación de la canción más que a la separación de la voz cantada. Debido a todo lo anterior, es que se decidió no implementar este artículo.

B.4. Adaptation of bayesian models

B.4.1. Introducción

En este artículo se busca estimar la señal de voz y la señal de acompañamiento musical usando modelos estadísticos. Dada una señal de audio $x(n) = s_v(n) + s_m(n)$, siendo $s_v(n)$ la señal de voz y $s_m(n)$ la señal del acompañamiento musical, el problema consiste en estimar la contribución de la voz $\hat{s}_v(n)$ y la música $\hat{s}_m(n)$. Para resolver lo anterior, se plantea el problema en el dominio de la frecuencia tomando la STFT (*short-time Fourier transform*) en ambos lados de la ecuación:

$$X(t, f) = S_v(t, f) + S_m(t, f) \tag{B.3}$$

donde: $t = 1, 2..T$ y $f = 1, 2..F$.

Dichas señales en el dominio de la frecuencia son modeladas usando GMM (*Gaussian Mixture Models*). La idea es representar a cada fuente como una variable aleatoria, con un conjunto de características espectrales, vistas como densidades espectrales de potencia "locales" (PSDs por sus siglas en inglés *Power Spectral Densities*), cada una representando un evento sonoro. La salida de un GMM es una suma ponderada de Q_k distribuciones gaussianas. Por lo tanto el modelo de cada fuente Λ_k está compuesto de Q_k estados correspondientes a Q_k PSDs locales $r^2_{k,i}(f) \leq f \leq F$, con $i = 1, 2..Q_k$.

Condicionado al estado i, el espectro de tiempo corto (*short-term spectrum*) $S_k(t)$ es visto como una realización de un vector complejo aleatorio Gaussiano, el cual tiene media cero y una matriz de covarianza diagonal $R_{k,i}$, la cual corresponde a la PSD local, por lo que:

$$R_{k,i} = diag[r^2_{k,i}(f)] \tag{B.4}$$

El modelo de cada fuente queda por tanto $C_k = \{$ u(k,i)$R_{k,i}$ $\}$, donde $u_{k,i} \geq 0$ son los pesos de cada densidad Gaussiana, y cumplen que $\sum_i u_{k,i} = 1$

La función de probabilidad (*pdf*) del espectro de tiempo corto S_k(t) es, por lo tanto:

$$p(S_k(t)|k) = [u(k,i)N_C(S_k(t) : 0, R(k,i))] \tag{B.5}$$

donde $N_C(V : u, R)$ es la *pdf* de un vector aleatorio gaussiano complejo V ϵ C^f, con el vector de medias: $u =$ $\{$u(f)$\}_f$ ϵ C^fy la matriz de covarianza :

$$R = diag[\{r^2(f)\}_f]\epsilon R^{FxF} \tag{B.6}$$

Y la *pdf* anterior es definida como:

$$N_C(V : u, R) = \prod_f \frac{1}{\pi r^2(f)} e^{-\frac{|V(f)-u(f)|^2}{r^2(f)}} \tag{B.7}$$

En otras palabras, lo que se busca es modelar cada fragmento del espectrograma como la suma ponderada de distintas PSD, cada una representando a una forma espectral diferente.

En el artículo se da un modelo de resolución del problema en una forma genérica, y luego a través de consideraciones experimentales se concluye que se pueden evitar modelos más complejos obteniendo los mismos resultados prácticos. Nosotros nos vamos a concentrar en el modelo más simplificado.

Conceptualmente se puede dividir el artículo en dos grandes etapas, la primera dedicada a obtener los modelos de voz y acompañamiento musical, y la segunda, dedicada a estimar la contribución de cada modelo en la grabación musical.

B.4.2. Obtención de los modelos de voz y acompañamiento musical

En la primera parte se obtienen los modelos de adaptación de música y voz, λ_m y λ_v respectivamente. El modelo de adaptación de música se obtiene a partir de las partes no-vocales de la grabación musical, y el modelo de voz se obtiene a partir de un modelo general de voz C_v y de la grabación musical.

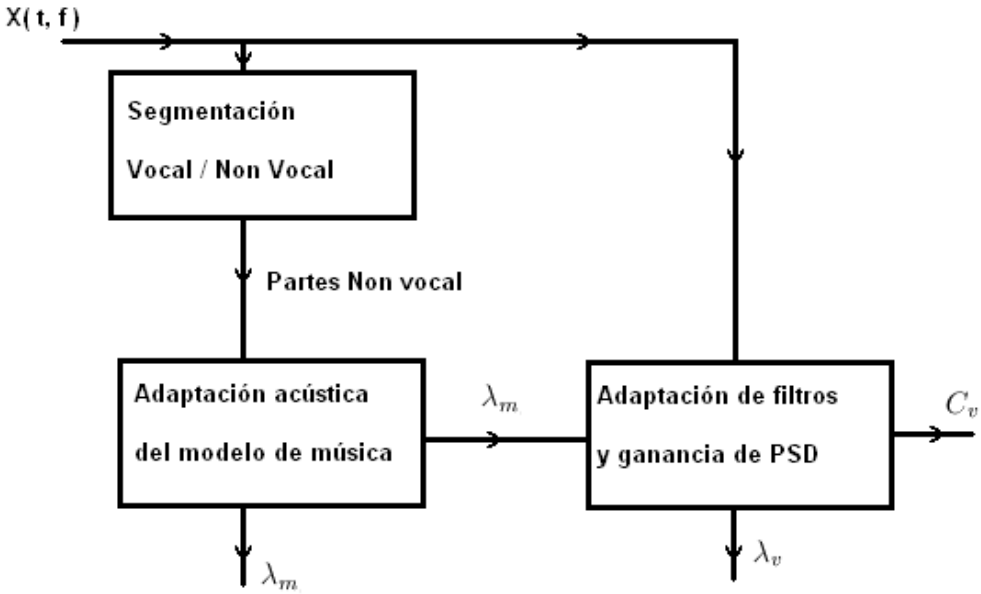

Figura B.3: diagrama de bloques

La idea de trabajar con modelos adaptados es que se puedan ajustar los modelos a los estados internos de las grabaciones musicales, ya que si sólo se entrenaran los GMM externamente habría que utilizar muchos más estados, debido a que las condiciones de las distintas grabaciones de entrenamiento son muy diferentes, por ejemplo, cambian las condiciones ambientes en las grabaciones, cambian los sensores, cambian las notas y ritmos, etc. De esta manera se reduce la complejidad computacional, haciendo el problema más sencillo y más exacto. El modelo general de voz C_V se obtiene aprendiendo los datos desde una base de datos (Y_v) con fragmentos de voz sin acompañamiento musical. Se usa para ello el criterio de maxima verosimilitud (*maximum–likelihood*):

$$C_v = ||C'_v p(Y_v|C'_v)||_{max} \tag{B.8}$$

Dicha ecuación se resuelve usando el algoritmo EM (*Expectation-Maximization*), inicializado por el algoritmo *k-means*. El modelo de adaptación de acompañamiento musical se halla en forma análoga, pero los datos de entrenamiento son los fragmentos no-vocales de la grabación musical.
Se obtienen entonces el modelo general de voz $C_v = \{u(v,i), R(v,i)\}$ y el modelo de adaptación del acompañamiento musical $\lambda_m = \{w_{m,j}, \sum_{m,j}\}_j$ siendo $w_{m,j}$ los pesos de las gaussianas y $\sum_{m,j} = diag[\{\sigma^2_{m,j}(f)\}_f]$ las matrices de covarianza diagonales.

Cabe aclarar que la cantidad de estados para la voz y para el acompañamiento musical, i y j respectivamente, son datos que se ingresan manualmente. En el artículo a través de pruebas experimentales se llega a que una medida justa es de 32 estados para cada uno.

Se pasa ahora al bloque de adaptación del modelo de voz, el cual consta de dos partes. La primera se denomina adaptación de filtrado (*filter adaptation*), y la segunda adaptación de ganancia (*gain adaptation*). En la primera parte, se busca encontrar un filtro lineal que adapte el modelo general a las características de la grabación, haciéndolo invariante a distintas acústicas, distintos micrófonos, etc. Por lo tanto el modelo adaptado λ_v será el resultado de un filtro lineal h_v aplicado al modelo general C_v. El objetivo de esta parte es por tanto obtener dicho filtro.

Sea $H_v = \{H_v(f)\}_f$ la transformada de Fourier de la respuesta al impulso de el filtro h_v. La relación de las PSD entre el modelo adaptado y el general es la siguiente:

$$\sigma^2_{v,i}(f) = \parallel H_v(f) \parallel^2 r^2_{v,i}(f) conf = 1, 2..F \tag{B.9}$$

Sea la matriz diagonal $\aleph_v = diag[\{\aleph_v(f)\}_f]$ con $\aleph_v = |H_v(f)|^2$, entonces el modelo adaptado de voz queda relacionado al general de la siguiente manera:

$$\lambda_v = \aleph_v C_v = \{u(v,i), \aleph_v R(v,i)\} \tag{B.10}$$

El filtro se halla usando:

$$\aleph_v = Arg(\aleph'_v p(X|(\aleph_v C_v)')\lambda_m)_{max} \tag{B.11}$$

Dicha ecuación se resuelve usando EM, lo anterior es una clase de lo llamado MLLR (*Maximum Likelihood Linear Regresion*). Como se puede observar, lo que se busca en esta adaptación es buscar el mejor filtro lineal que modifique las varianzas en cada frecuencia, adaptando cada banda de frecuencias a la grabación musical bajo estudio.

En la segunda parte, el objetivo es adaptar la energía de cada PSD i. Cada estado GMM es descrito por una forma espectral característica o PSD, la cual corresponde por ejemplo a una nota musical, la misma puede estar en promedio con más energía en una grabación que en otra, el objetivo de esta parte es tomar en consideración lo anterior. Una ganancia positiva $g_{v,i} > 0$ es asociada a cada PSD $\{r^2_{k,i}(\mathrm{f})\}_f$ del modelo de C_v, esta ganancia corresponde a la energía media del evento sonoro representado por esa PSD. Finalmente se adapta el modelo de voz de la siguiente manera:

$$\lambda_v = g_v C_v = \{u(v,i), [g(v,i)R](v,i)\} \tag{B.12}$$

donde g_v=$\{g_{v,i}\}_i$ es un vector de ganancias PSD, y el símbolo se usa para distinguir esta multiplicación de la multiplicación de la primera parte. Por lo tanto se busca estimar g_v de la siguiente manera:

$$g_v = Arg(g'_v p(X|g_v)' C_v \lambda_m)_{max} \tag{B.13}$$

Esta ecuación también se resuelve usando EM. Para obtener la adaptación hay que reunir ambas partes. Se busca resolver entonces:

$$(\aleph_v, g_v) = Arg(\aleph'_v . g'_v p(X|g'_v . \aleph'_v C_v \lambda_m))_{max} \tag{B.14}$$

El algoritmo EM no se puede aplicar para resolver este problema, una posible solución es aplicar el algoritmo SAGE (*space–alternating EM*), en donde se estima cada

parámetro en pasos separados. El artículo hace hincapié sobre el alto costo computacional que implica utilizar ese método, por lo que propone realizar una variante del algoritmo EM haciendo un paso de E (*Expectation*) y muchos pasos alternados de M (*Maximization*).

Cabe destacar que las ecuaciones para implementar este bloque están en el apéndice del artículo, por lo que ahí se tiene un gran punto de partida para la implementación.

B.4.3. Separación de los modelos

Resta ahora la parte final, donde a partir de los modelos hallados anteriormente se obtiene una estimación de las señales de voz y acompañamiento musical en el dominio de la frecuencia, y luego se antitransforman las señales para obtenerlas en el dominio del tiempo.

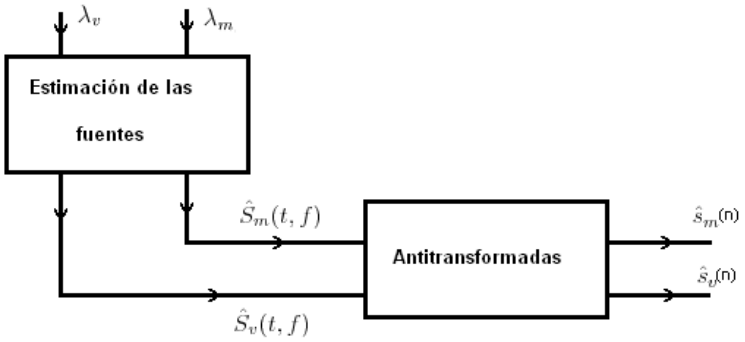

Figura B.4: diagrama de bloques para separación de modelos

En la estimación de las fuentes, lo que se busca es la estimación del siguiente criterio:

$$d(\hat{s}_v, \hat{s}_m; s_v, s_m) = \| S_v - \hat{S}_v \|^2 + \| S_m - \hat{S}_m \|^2 \tag{B.15}$$

Esto conduce a lo que se conoce como filtros de Wiener, en donde:

$$\hat{S}_v(t,f) = \sum_{i,j} \gamma_{i,j}(t) \frac{\sigma^2_{v,i}(f)}{\sigma^2_{v,i}(f) + \sigma^2_{m,i}(f)} \tag{B.16}$$

$$\hat{S}_m(t,f) = \sum_{i,j} \gamma_{i,j}(t) \frac{\sigma^2_{m,i}(f)}{\sigma^2_{m,i}(f) + \sigma^2_{v,i}(f)} \tag{B.17}$$

Siendo $\gamma_{i,j}$(t) la probabilidad a posteriori de que el par de estado (i,j) haya emitido la trama t, y cumple que $\sum_{i,j} \gamma_{i,j}(t) = 1$

$$\gamma_{i,j}(t) = P(q_1 = i, q_2 = j | X, \lambda_v, \lambda_m) \propto w_{v,i} w_{m,j} N_C(X(t); 0, \Sigma_{v,j} + \Sigma_{m,j}) \tag{B.18}$$

Donde $X(t)$ es el espectro de tiempo corto (*short-term spectrum*), y $q_1(t)$, $q_2(t)$ son los estados de los modelos. Recordando que los modelos de acompañamiento musical y de voz son de la forma:

$$\lambda_m = \{w_{m,j}, \Sigma_{m,j}\}_j \tag{B.19}$$

siendo $w_{m,j}$ los pesos de las Gaussianas y $\Sigma_{m,j} = diag[\{(\Sigma_{m,j})^2(f)\}_f]$ las matrices de covarianza diagonales, y

$$\lambda_v = \{w_{v,j}, \Sigma_{v,j}\}_i \tag{B.20}$$

siendo $w_{v,i}$ los pesos de las Gaussianas y $\Sigma_{v,i} = diag[\{(\Sigma_{v,i})^2(f)\}_f]$ las matrices de covarianza diagonales.

En este punto ya tenemos la estimación en el dominio de la frecuencia de ambas señales, queda por tanto el último bloque para antitransformar. La idea es usar una técnica llamada OLA (*Overlap and Add*), esta técnica es de amplio uso en la recuperación de este tipo de señales.

Apéndice C

Bandas críticas y su relación con el sistema auditivo

Puede considerarse la banda crítica, como la mínima banda de frecuencia alrededor de una determinada frecuencia, que activa la misma zona de la membrana basilar. Esto hace que se pueda considerar al sistema auditivo periférico como un conjunto de filtros pasabanda, con bandas superpuestas. El funcionamiento mismo de la membrana basilar sería el responsable de ello. Aparentemente cada banda crítica correspondería a una distancia fija a lo largo de la membrana basilar, de aproximadamente 1.3 mm de longitud (independientemente de la frecuencia central), abarcando unas 150 células receptoras en el órgano de Corti, de un total de 3600 células capilares que hay en línea entre el helicotrema y la ventana oval bajo la membrana basilar. Respecto a la banda crítica y la discriminación de parciales o armónicos en un sonido complejo, se puede acotar que se pueden detectar entre los primeros 5 hasta 8 parciales de un sonido complejo. A partir de allí los parciales caerían dentro de una misma banda crítica. Se ha establecido también que los músicos -con entrenamiento del oído- son capaces de realizar una detección más fina que quienes no poseen dicho entrenamiento, aún dentro de lo que sería una banda crítica [60].

Apéndice D

Valores del ERB y la Frecuencia central

La siguiente tabla muestra para cada canal del banco de filtros auditivos, su frecuencia central y su ancho de banda correspondiente.

Canal	f_c(Hz)	ERB(Hz)	Canal	f_c(Hz)	ERB(Hz)	Canal	f_c(Hz)	ERB(Hz)
1	5000	564,39	44	1777,5	216,56	87	540,98	83,093
2	4884,8	551,96	45	1733,3	211,79	88	524,02	81,262
3	4772,2	539,8	46	1690	207,12	89	507,43	79,472
4	4662	527,91	47	1647,8	202,56	90	491,21	77,721
5	4554,2	516,28	48	1606,4	198,1	91	475,35	76,009
6	4448,9	504,91	49	1566	193,73	92	459,84	74,334
7	4345,8	493,78	50	1526,4	189,46	93	444,67	72,697
8	4245	482,9	51	1487,8	185,29	94	429,83	71,095
9	4146,5	472,27	52	1450	181,21	95	415,32	69,529
10	4050,1	461,86	53	1413	177,22	96	401,13	67,997
11	3955,8	451,69	54	1376,8	173,31	97	387,25	66,499
12	3863,6	441,74	55	1341,4	169,49	98	373,68	65,034
13	3773,5	432	56	1306,8	165,76	99	360,4	63,601
14	3685,3	422,49	57	1273	162,76	100	347,42	62,2
15	3599,1	413,18	58	1239,9	158,54	101	334,73	60,83
16	3514,7	404,08	59	1207,6	155,04	102	322,31	59,49
17	3432,3	395,17	60	1175,9	151,63	103	310,17	58,179
18	3351,6	386,47	61	1145	148,29	104	298,29	56,898
19	3272,7	377,95	62	1114,7	145,02	105	286,68	55,644
20	3195,6	369,63	63	1085,1	141,83	106	275,32	54,418
21	3120,1	361,48	64	1056,2	138,7	107	264,22	53,219
22	3046,4	353,52	65	1027,9	135,65	108	253,36	52,047
23	2974,2	345,73	66	1000,2	132,66	109	242,73	50,9
24	2903,6	338,12	67	973,09	129,73	110	232,34	49,779
25	2834,6	330,67	68	946,61	126,88	111	222,18	48,682
26	2767,1	323,38	69	920,72	124,08	112	212,25	47,61
27	2701,1	316,26	70	895,39	121,35	113	202,53	46,561
28	2636,6	309,29	71	870,63	118,67	114	193,03	45,535
29	2573,5	302,48	72	846,4	116,06	115	183,73	44,532
30	2511,7	295,81	73	822,72	113,5	116	174,64	43,551
31	2451,4	289,3	74	799,55	111	117	165,76	42,592
32	2392,3	282,92	75	776,89	108,56	118	157,06	41,653
33	2334,6	276,69	76	754,74	106,17	119	148,56	40,736
34	2278,1	270,59	77	733,07	103,83	120	140,25	39,838
35	2222,9	264,63	78	711,88	101,54	121	132,12	38,961
36	2168,9	258,8	79	691,15	99,302	122	124,16	38,102
37	2116	253,1	80	670,89	97,115	123	116,39	37,263
38	2064,4	247,53	81	651,07	94,975	124	108,78	36,442
39	2013,9	242,07	82	631,68	92,883	125	101,34	35,639
40	1964,4	236,74	83	612,72	90,837	126	94,071	34,854
41	1916,1	231,52	84	594,18	88,836	127	86,957	34,086
42	1868,9	226,42	85	576,05	86,878	128	80	33,335
43	1822,7	221,44	86	558,32	84,965			

Apéndice E

Modelos Ocultos de Markov (*Hidden Markov Models*)

El objetivo de esta sección es introducir el tema de modelos ocultos de Markov o HMM. El método de detección de *pitch*[1] implementado se basa principalmente en HMM, por tanto es necesario una breve introducción a esté tema para conocer cuál es la idea principal que hay detrás del algoritmo implementado.

E.1. Definiciones principales

A continuación se presentan algunos conceptos claves a saber en el tema de HMM. Cabe destacar que esta sección no pretende ser una introducción al tema HMM, aquí sólo se dan las ideas principales de HMM, para tener una idea de cómo funciona el algoritmo de detección de *pitch* implementado. Para entrar en detalle y con mayor profundidad al tema de HMM puede consultarse [61], el cual es un tutorial de HMM orientado a la utilización de esta técnica en el procesamiento de voz.

♣ **Cadena de Markov:** Es una serie de eventos $w(t)$, en los cuales la probabilidad de que ocurra un evento determinado depende de eventos anteriores (se dice que tienen memoria)

♠ **Cadenas de Markov discretas de primer orden:** Una secuencia particular de estados, en T períodos de tiempo, es un determinado conjunto $W(T)$ de estados, $W(T) = \{w(1), w(2), w(3), ...w(T)\}$, en los cuales los elementos del mismo no son IID (Independientes e Idénticamente Distribuidos). Se dice que es de primer orden, porque sólo depende del estado anterior (depende solamente del estado $n-1$).

Para determinar el modelo de producción de secuencias, se define la probabilidad de transición:

$$P(w_j(t+1)|w_i(t)) = a_{i,j} \tag{E.1}$$

La ecuación E.1 nos da la probabilidad de estar en el estado i en un tiempo t, y pasar al estado j en el siguiente instante $t+1$.

[1] Ver el capítulo 5.

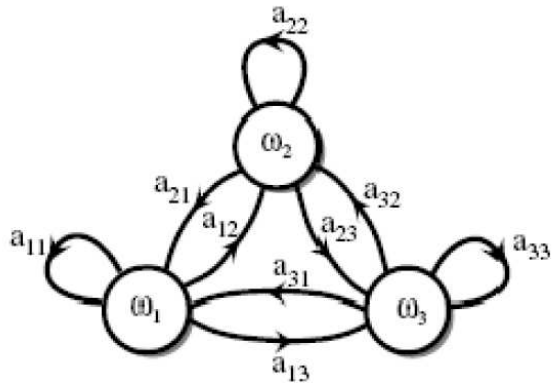

Figura E.1: Estados y probabilidad de transición

Las probabilidades de transición no tienen porqué ser simétricas, es decir, a_{ij} no tiene porqué ser igual a a_{ji}. Los estados son representados como nodos como muestra la figura E.1.

Dado el modelo de transición de probabilidades a_{ij} , la probabilidad de que ocurra una secuencia determinada de estados, es la multiplicación de las probabilidades de transición entre estados sucesivos, y también la probabilidad de empezar en un determinado estado inicial.

♢ **Modelos de Markov ocultos discretos de primer orden:** En cada instante t, estamos en un estado $w(t)$ oculto (no sabemos cuál es), y se emite un símbolo o estado visible $v(t)$. Se puede decir que se tienen los estados visibles $v_k(t)$, pero no los estados $w_i(t)$. Para modelar lo anterior, se considera cuál es la probabilidad de ver el estado $v_k(t)$ dado un estado $w_i(t)$.

$$P(v_k(t+1)|w_j(t)) = b_{j,k} \tag{E.2}$$

Se define como "estado absorbente" al estado en el cual una vez que se cae en el mismo, la probabilidad de ir a otro estado es 0 (ó lo que es lo mismo, la probabilidad de pasar al mismo estado en el siguiente instante de tiempo, es 1).

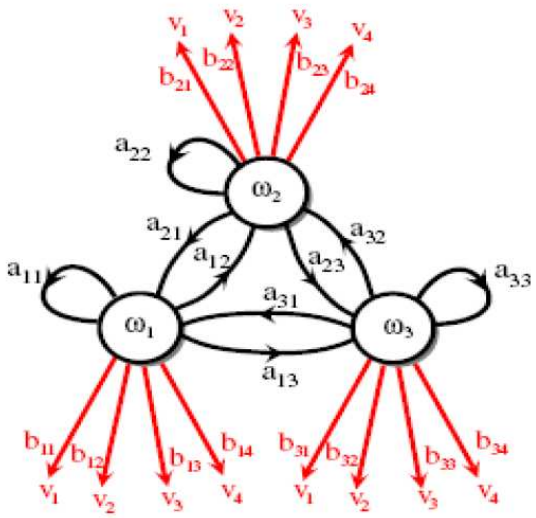

Figura E.2: Modelo oculto de Markov

Las condiciones que deben cumplir, tanto las probabilidades de transición de estados, como las de observación son:

$$\sum_j a_{ij} = 1 \forall i \tag{E.3}$$

$$\sum_k b_{jk} = 1 \forall j \tag{E.4}$$

Considerando un modelo oculto de Markov, hay tres tipos de problemas:
• Dados los parámetros del modelo, compútese la probabilidad de un secuencia de salida en particular.

• Dada una secuencia de salida o un conjunto de tales secuencias, encuéntrese el conjunto de estados de transición y probabilidades de salida más probables. En otras palabras, entrénense a los parámetro del HMM dada una secuencia de datos.

• Dados los parámetros del modelo, encuéntrese la secuencia más probable de estados ocultos que puedan haber generado una secuencia de salida dada. Este problema se resuelve mediante el algoritmo de Viterbi.

Este último caso, es el que se considera en la parte de detección del *pitch*.

♡ **Algoritmo de Viterbi:** El algoritmo de Viterbi es una técnica muy utilizada en el procesamiento de voz y se utiliza conjuntamente con los modelos ocultos de Markov (HMM).

El algoritmo de Viterbi permite encontrar las secuencia de estados más probable en un Modelo oculto de Markov, $S = (q_1, q_2, ..., q_T)$, a partir de una observación $O = (o_1, o_2, ..., o_T)$, es decir, obtiene la secuencia óptima que mejor explica la secuencia de observaciones.

Se considera la variable $\delta_t(i)$ que se define como:

$$\delta_t(i) = \max_{q_1, q_2, \cdots q_{t-1}} P(q_1, q_2, ...q_t = i, o_1, o_2, ...o_t|\mu) \tag{E.5}$$

$\delta_t(i)$ es la probabilidad del mejor camino hasta el estado i habiendo visto las t primeras observaciones. Esta función se calcula para todos los estados e instantes de tiempo.

$$\delta_{t+1}(i) = [\max_{1<i<N} \delta_i(a_{ij})]b_j(o_{t+1}) \tag{E.6}$$

Puesto que el objetivo es obtener la secuencia de estados más probable, será necesario almacenar el argumento que hace máxima la ecuación anterior en cada instante de tiempo t y para cada estado j, para ello se utiliza la variable $\varphi_t(j)$, siendo
$\varphi_{t+1}(j) = \arg\max_{1<i<N} \delta_t(i)a_{ij}$ [62].
Una vez evaluada las anteriores ecuaciones se obtiene la secuencia optima que mejor explica la secuencia de observaciones.

Apéndice F

Manual de Usuario

F.1. Requerimientos

El algoritmo implementado exige grandes costos computacionales. Por ejemplo para un archivo de 2.5 segundos de duración, se trabaja con matrices de 128 filas (número de canales) y 40000 columnas.[1] Se decidió por tanto cortar los archivos de entrada en un máximo de 40000 muestras.

Es necesario contar con una máquina potente, buen procesador y una buena cantidad de memoria RAM. El código fue probado en una PC Intel® Dual Core 2.8 GHz con 512 MB de RAM.[2] En esta máquina para un fragmento de 2.5 segundos se obtiene un tiempo total de procesamiento de aproximadamente 30 minutos. Este tiempo puede disminuir bastante si se disminuye el número de canales, por ejemplo a 64 canales,[3] y puede bajarse más aún si se posee de antemano un archivo de texto con el *pitch* ya calculado de alguna otra manera.[4]

Otro requerimiento es contar con los paquetes de procesamiento de señales utilizados por Matlab®. La versión de Matlab® utilizada fue la versión 7.0.0.199.20 Release 14.

[1] 2.5 segundos equivalen a 40000 muestras trabajando a 16 kHz.

[2] El algoritmo se testeó en varias PCs, siendo esta la menos potente en cuanto a memoria RAM se refiere.

[3] Puede cambiarse manualmente en el código, en el archivo `main_pitch.m` donde dice *numChannel*.

[4] Por ejemplo si se tiene de antemano la voz cantada aislada, puede calcularse el *pitch* utilizando *WaveSurfer*. En la sección F.4 se explica esto último.

F.2. ¿Cómo usar el código?

A continuación se explica cómo utilizar el código implementado.

F.2.1. Interfaz Gráfica

Al código implementado en Matlab® se le agregó una interfaz gráfica también diseñada en Matlab®.

La interfaz gráfica se muestra en la siguiente figura.[5]

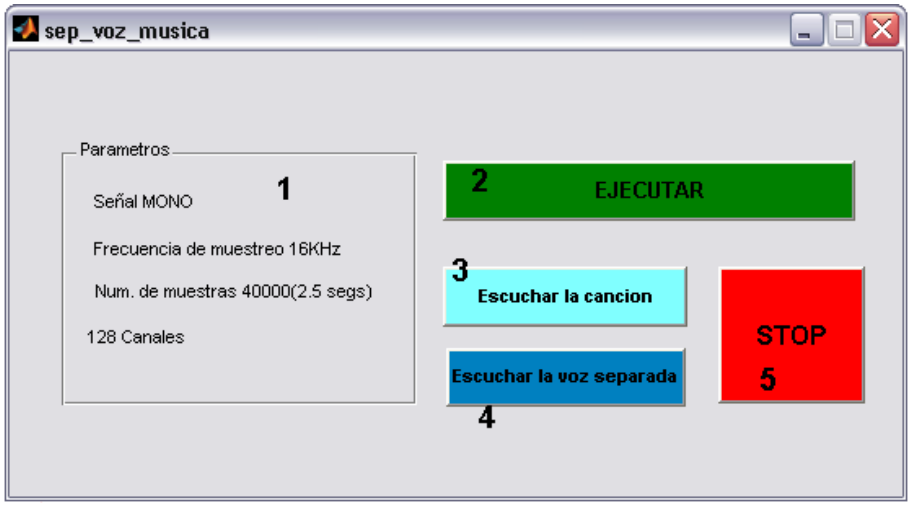

Figura F.1: Interfaz Gráfica del Programa Implementado

A continuación se explica el funcionamiento de la misma:

1 Cuadro *Parámetros*. Muestra los parámetros utilizados en el código.

2 Botón *EJECUTAR*. Corre el programa.

3 Botón *escuchar canción*. Permite escuchar la canción seleccionada para separar.

4 Botón *escuchar voz separada*. Permite escuchar la voz cantada separada mediante el algoritmo implementado.

5 Boton *STOP*. Detiene la reproducción de la canción o la voz separada.

F.2.2. Ejecución del programa

La ejecución del programa es relativamente sencilla. A continuación se muestra paso a paso cómo ejecutar el código y se explican los detalles de cada menú que aparece.

[5]El aspecto visual de la interfaz puede variar de acuerdo al estilo de visualización que tenga el sistema operativo.

En principio para poder correr el código lo único que se necesita es una señal en formato `WAV` de tipo `MONO` y frecuencia `16kHz`. El código permite cargar otros archivos pero son opcionales.

1 Primero hay que ejecutar el script de la interfaz gráfica en Matlab®, este archivo se encuentra con el nombre de `sep_voz_musica.m` o `sep_voz_musica.fig`.[6] Una vez ejecutado alguno de estos dos scripts aparece la interfaz gráfica mostrada en la figura F.1.

2 Se ejecuta el código con el botón *EJECUTAR*.

3 Aparece el cuadro de diálogo que se muestra en la figura F.2. El mismo pide que se cargue el archivo con la canción a separar.

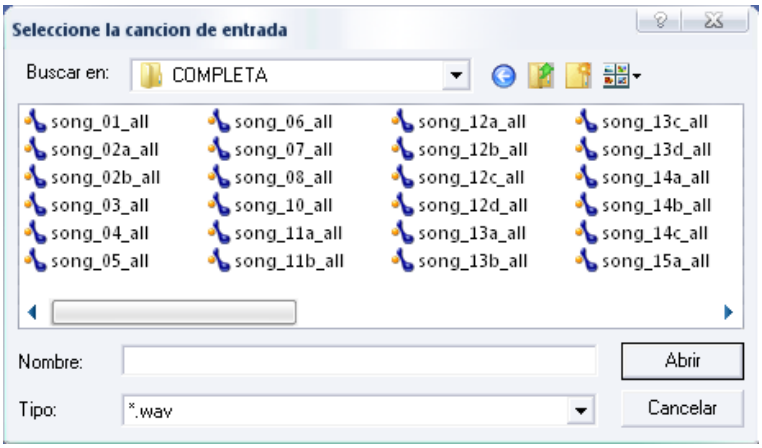

Figura F.2: Cuadro de diálogo para cargar la canción a separar.

4 Al cargar la canción aparece otro cuadro de diálogo como el de la figura F.3. Aquí se pregunta si se tiene algún archivo con la voz cantada ya separada para comparar al final el resultado obtenido.

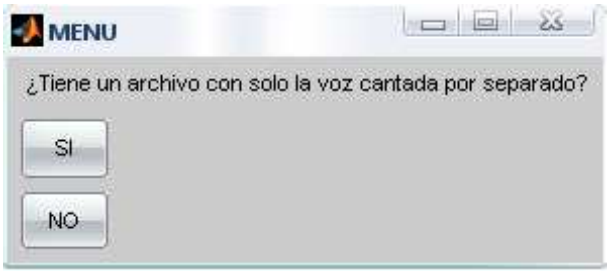

Figura F.3: Cuadro de diálogo preguntando si se tiene un archivo con el cual comparar.

[6] Si se quieren tener todas las variables disponibles, se puede ejecutar el archivo *main_pitch.m*.

En esta parte se puede marcar *SI* o *NO*, en caso de marcar *SI* se abre un cuadro de diálogo similar al de la figura F.2 en el que se pide que la carga del archivo con la señal de sólo la voz cantada.

5 Luego aparece un nuevo cuadro de diálogo como el de la figura F.5. En el mismo se pregunta si se tiene algún archivo con el *pitch* ya calculado.

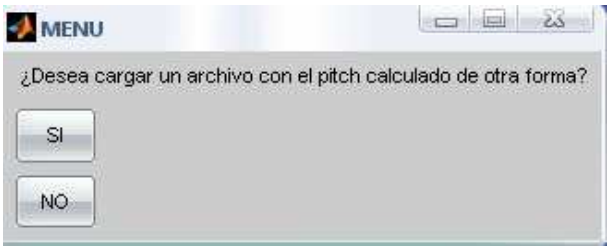

Figura F.4: Cuadro de diálogo preguntando si se tiene algún archivo con el *pitch* ya calculado.

En caso afirmativo se abre un cuadro similar al de la figura F.2 pidiendo que se cargue el archivo con el *pitch* ya calculado. Una forma de obtener el *pitch* se explica en la sección F.4 utilizando *WaveSurfer*. En caso que no se tenga un archivo de *pitch* calculado o se quiera dejar que el programa lo calcule simplemente marcar la opción *NO*.

6 El último cuadro de diálogo que se abre en esta etapa, pregunta si se tiene el acompañamiento musical por separado. En caso afirmativo se puede proceder a cargarlo y al final del código se realiza la evaluación del sistema.

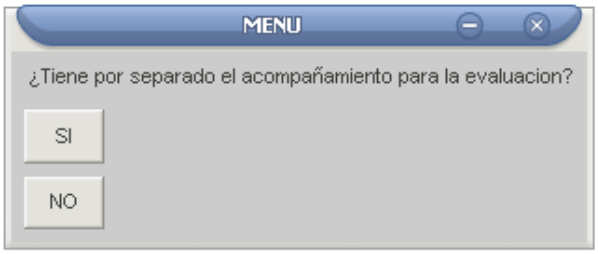

Figura F.5: Cuadro de diálogo preguntando si se tiene el acompañamiento musical ya separado para la evaluación.

7 Luego de los pasos anteriores empieza a correr el algoritmo, el mismo implica muchos cálculos computacionales, y por lo tanto es conveniente dejar que se ejecute hasta que aparezca un nuevo cuadro de diálogo.
Mientras corre el algoritmo va mostrando las etapas en las que va en la pantalla de Matlab como se muestra en la figura F.6

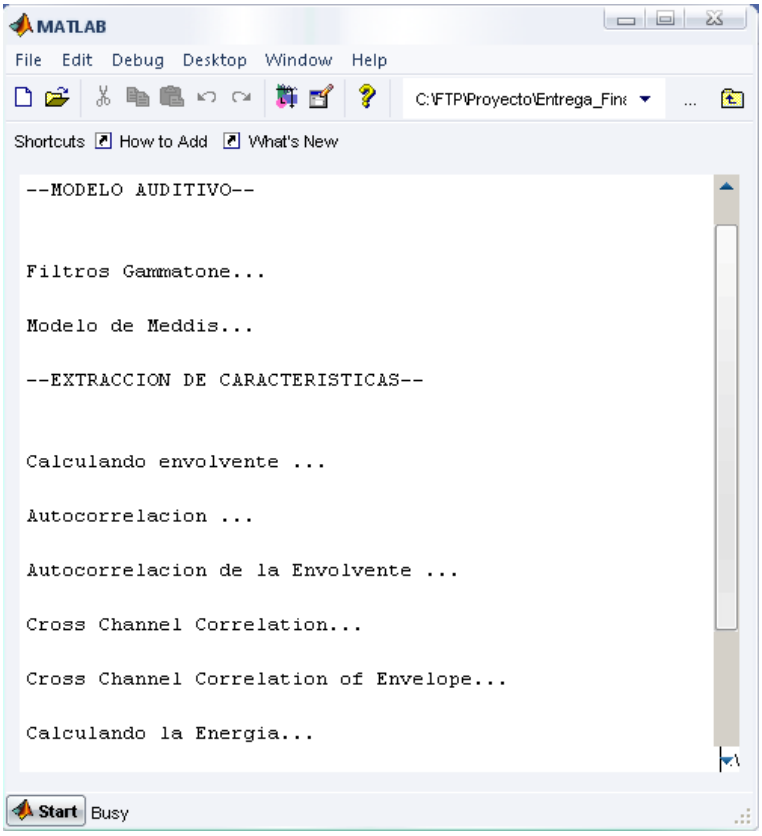

Figura F.6: Etapas que se van marcando en la consola de Matlab.

8 Luego de algunos cálculos aparece un último menú preguntando si se quieren ver algunas gráficas que muestran resultados obtenidos.

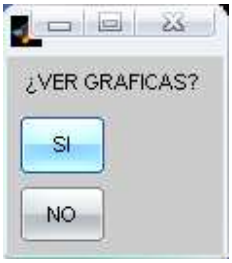

Figura F.7: Cuadro final de diálogo preguntando si se quieren ver gráficas.

9 Por último luego de terminado todos los cálculos se pueden utilizar los botones de la interfaz para escuchar tanto la canción original como la voz cantada separada.

F.3. Problemas

Dado los grandes costos computacionales que implica el algoritmo y al estar trabajando con un prototipo en Matlab, pueden ocurrir diferentes problemas. Se recuerda que la versión de Matlab® utilizada fue la 7.0 release 14.

`1: El PC no responde.`

- Cierre la mayor cantidad de aplicaciones antes de correr el programa.
- Espere, ya que el programa demora bastante, para un fragmento de 2.5 segundos demora casi 30 minutos.

`2: El código da error.`

- Verifique que los archivos de audio cargados sean formato `WAV` y `MONO` (no STEREO)
- Verifique que su versión de Matlab® contenga todo los toolboxes y funciones necesarias.
  Algunas funciones que utiliza el código son:
  - `filter`
  - `filtfilt`
  - `butter`
  - `fft`
  - `ifft`
  - `hilbert`
  - `wavread`
  - `wavwrite`
  - `menu`
  - `uigetfile`
  - `strcat`

F.4. Cálculo del *pitch* utilizando *WaveSurfer*

En esta sección se explica cómo calcular el *pitch* utilizando el *WaveSurfer*.[7]

WaveSurfer es un software pensado para manipular señales de audio, entre muchas de sus funciones se encuentra el cálculo del *pitch*. Para calcular el *pitch* de la voz cantada utilizando *WaveSurfer* es necesario tener de antemano la voz cantada sola por separado, ya que *WaveSurfer* al igual que *Praat*[8] calcula el *pitch* para voz "limpia".

Primero se abre *WaveSurfer*, la interfaz del programa es la siguiente:

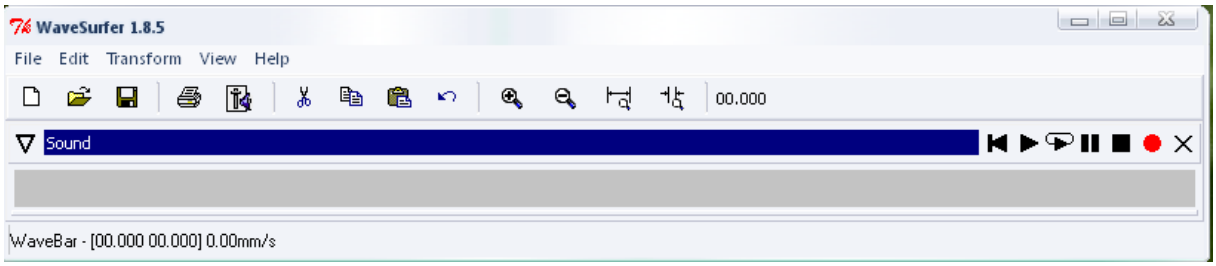

Figura F.8: Interfaz de *WaveSurfer*

Luego hay que ir a `archivo` (`file`) o la carpetita para abrir el archivo de la voz cantada sola por separado. Al abrir tal archivo se presenta el siguiente cuadro:

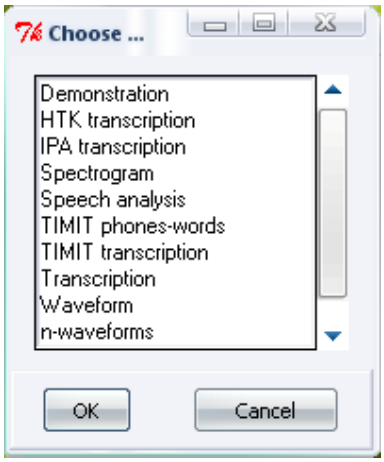

Figura F.9: Opciones de vista de la señal

Aquí hay que elegir la opción `waveform` por ejemplo. Luego aparece la forma de onda de la señal. Sobre la misma se cliquea con el botón derecho del mouse y aparece

[7]Disponible web: http://www.speech.kth.se/WaveSurfer/download.html, también se encuentra disponible en el CD que contiene el prototipo entregable en Matlab.

[8]*Praat* es otro software para manipular señales de audio.

un menú, ahí hay que moverse hacia `Create Pane` para luego elegir la opción `Pitch Contour`, como muestra la siguiente figura.

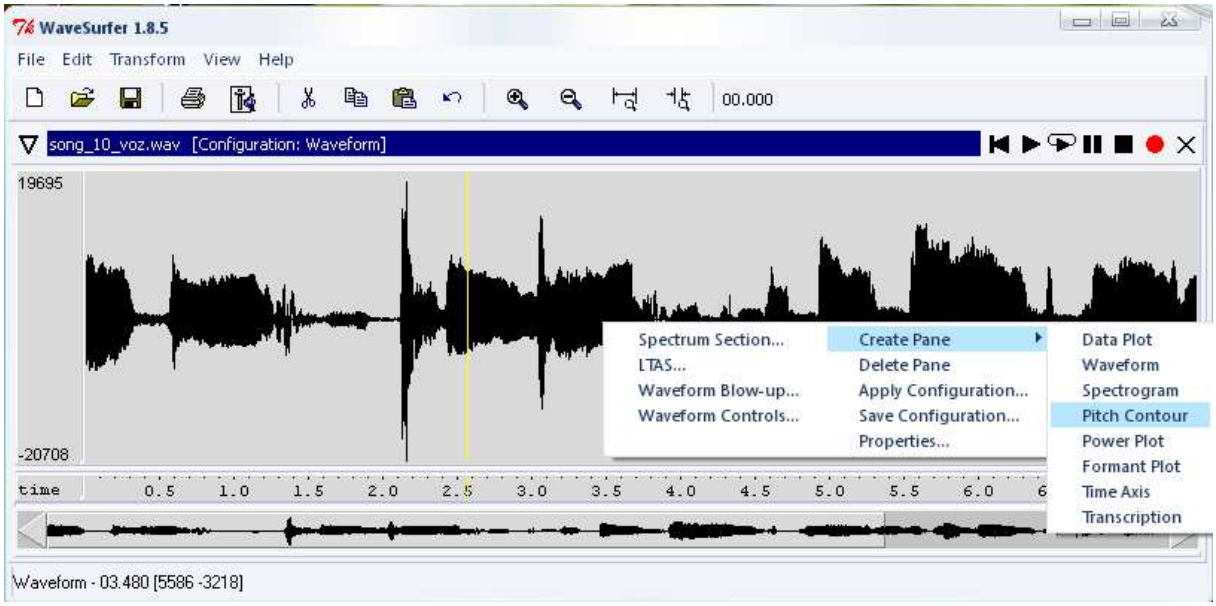

Figura F.10: Creación del panel de *pitch*

Una vez hecho lo anterior aparece un nuevo panel con el *pitch* calculado como se muestra en la figura F.11. Luego de esto se cliquea sobre este nuevo panel con el botón derecho del mouse y se elige la opción propiedades.

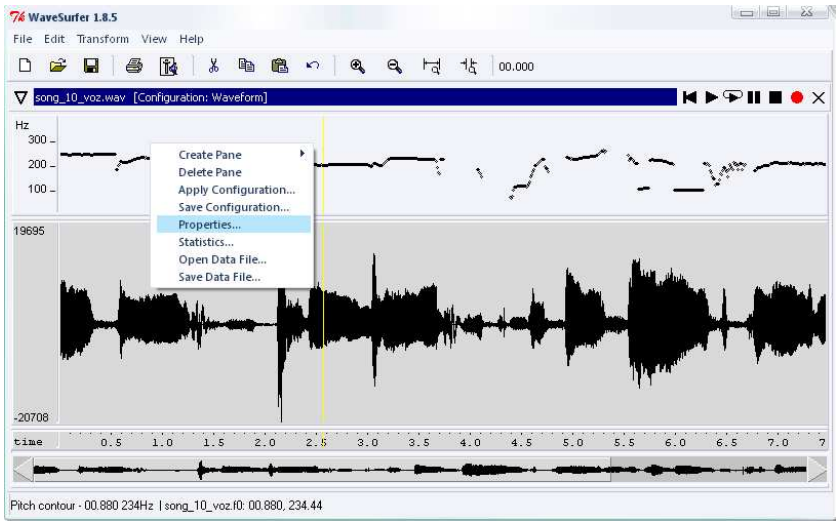

Figura F.11: Panel con el *pitch*

En el cuadro de propiedades hay que ir a la solapa `Pitch Contour` y setear los valores

adecuados que son los que se observan en la figura F.12. Luego de marcar aceptar se recalculará el *pitch* con los nuevos parámetros.

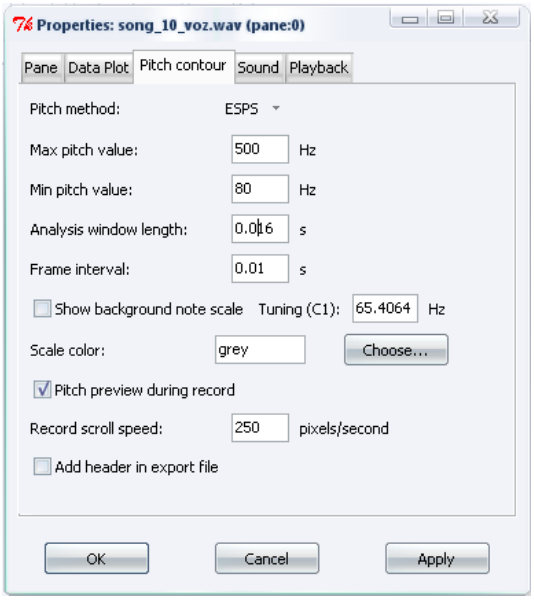

Figura F.12: Seteo de propiedades para el *pitch*

Por último sobre el panel del *pitch* se cliquea nuevamente con el botón derecho del mouse y se elige la opción `Save Data File`. Aquí se guarda como archivo de texto con la extensión `.txt`.

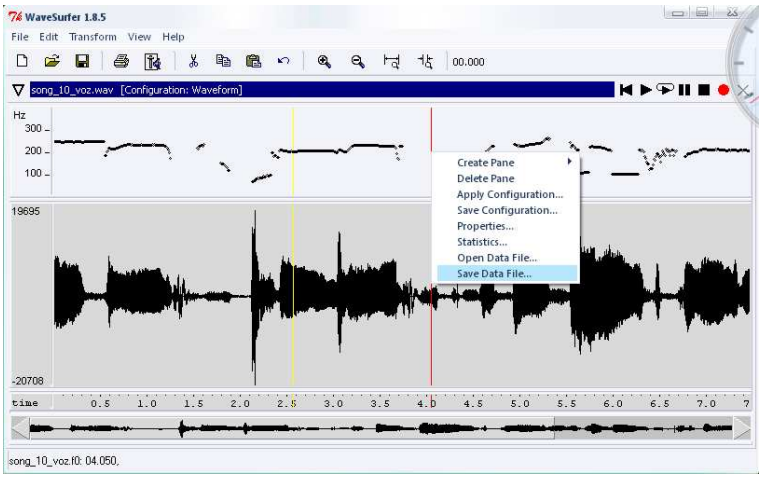

Figura F.13: Guardar el *pitch* calculado.

Apéndice G

Base de datos

A continuación se presenta una tabla con los archivos contenidos en la base de datos utilizada para ajustar y testear el código.

Archivo	Duración	Artista	Título	Nota
song_01_all	10seg	Alicia Keys	Diary	SV
song_02a_all	8seg	Avril Lavigne	My Happy Ending	SS
song_02b_all	10seg	Avril Lavigne	My Happy Ending	SS
song_03_all	10seg	Coldplay	Speed of sound	SV
song_04_all	10seg	Destinys Child	Soldier	SV
song_05_all	10seg	Fuel	Falls on me	SV
song_06_all	9seg	Jennifer López	Baby I Love U	SV
song_07_all	10seg	Natalie	Going Crazy	SV
song_08_all	10seg	Nickelback	Feeling way too damn good	SV
song_10_all	10seg	Ryan Cabrera	True	SV
song_11a_all	9seg	Alan Jackson	Remember When	SS
song_11b_all	9seg	Alan Jackson	Remember When	SS
song_12a_all	9seg	Big and Rich	Save a Horse Ride a Cowboy	SV
song_12b_all	9seg	Big and Rich	Save a Horse Ride a Cowboy	SV
song_12c_all	11seg	Big and Rich	Save a Horse Ride a Cowboy	SV
song_12d_all	4seg	Big and Rich	Save a Horse Ride a Cowboy	SV
song_13a_all	9seg	Brooks and Dunn	That's What it's all about	SV
song_13b_all	4seg	Brooks and Dunn	That's What it's all about	SV
song_13c_all	6seg	Brooks and Dunn	That's What it's all about	SV
song_13d_all	9seg	Brooks and Dunn	That's What it's all about	SS
song_14a_all	13seg	Chely Wright	Back of the Bottom Drawer	SV

Archivo	**Duración**	**Artista**	**Título**	**Nota**
song_14b_all	13seg	Chely Wright	Back of the Bottom Drawer	SV
song_14c_all	9seg	Chely Wright	Back of the Bottom Drawer	SV
song_15a_all	6seg	Lee Ann Womack	I May Hate Myself in the Morning	SV
song_15b_all	8seg	Lee Ann Womack	I May Hate Myself in the Morning	SV
song_15c_all	7seg	Lee Ann Womack	I May Hate Myself in the Morning	SV
song_15d_all	8seg	Lee Ann Womack	I May Hate Myself in the Morning	SV
song_15e_all	9seg	Lee Ann Womack	I May Hate Myself in the Morning	SV
song_16a_all	7seg	Martina McBride	God's Will	SV
song_16b_all	7seg	Martina McBride	God's Will	SV
song_17a_all	13seg	Montgomery Gentry	Gone	SV
song_17b_all	8seg	Montgomery Gentry	Gone	SV
song_17c_all	6seg	Montgomery Gentry	Gone	SV
song_18a_all	11seg	Reba McEntire	Somebody	SV
song_18b_all	8seg	Reba McEntire	Somebody	SV
song_19a_all	8seg	Shania Twain	Don't	SV
song_19b_all	10seg	Shania Twain	Don't	SV
song_20a_all	5seg	Toby Keith	Honky Tonky U	SS
song_20b_all	6seg	Toby Keith	Honky Tonky U	SV
song_20c_all	6seg	Toby Keith	Honky Tonky U	SV
song_20d_all	6seg	Toby Keith	Honky Tonky U	SV
song_21a_all	20seg	Le Petit Comite	Roxanne	SV
song_21b_all	10seg	Le Petit Comite	Roxanne	SV
song_21c_all	12seg	Le Petit Comite	Roxanne	SV
song_22a_all	4seg	Le Petit Comite	Logical Song	SV
song_22b_all	6seg	Le Petit Comite	Logical Song	SV
song_23a_all	15seg	Le Petit Comite	Highway to Hell	SV
song_23b_all	38seg	Le Petit Comite	Highway to Hell	SV
todo5	2seg	Fuel	Falls on me	SVA
todo52	2seg	Fuel	Falls on me	SVA
todo53	2seg	Fuel	Falls on me	SVA
todo61	2seg	Jennifer López	Baby I Love U	SVA
todo62	3seg	Jennifer López	Baby I Love U	SVA
todo63	3seg	Jennifer López	Baby I Love U	SVA
todo71	2seg	Natalie	Going Crazy	SVA
todo72	2seg	Natalie	Going Crazy	SVA
todo73	2seg	Natalie	Going Crazy	SVA
todo91	1seg	Norah Jones	Lonestar	SVA
todo92	2seg	Norah Jones	Lonestar	SVA
todo93	3seg	Norah Jones	Lonestar	SVA
todo101	3seg	Ryan Cabrera	True	SVA
todo102	2seg	Ryan Cabrera	True	SVA
todo103	2seg	Ryan Cabrera	True	SVA
todo11c	3seg	Alan Jackson	Remember When	SVA
todo12c	2seg	Alan Jackson	Remember When	SVA
todo13c	2seg	Alan Jackson	Remember When	SVA
todo19a	1seg	Shania Twain	Don't	SV
todo19b	3seg	Shania Twain	Don't	SV
todo20b	3seg	Toby Keith	Honky Tonky U	SV
todo20c	2seg	Toby Keith	Honky Tonky U	SV
todo20d	2seg	Toby Keith	Honky Tonky U	SV
ritmode	2seg	Puagh	Ritmo de las ideas	SS
panycirco	2seg	La Chancha	Pan y Circo	SS
noctdeluna	3seg	La Chancha	Hotel nocturno de la luna	SS
coca	3seg	Joaquín Sabina	19 dias y 500 noches	SS
hadar	3seg	Hadar	Oculto Ser	SS

♣ En la columna Nota la referencia es la siguiente:

• SS: En la base de datos está sólo disponible el fragmento con la canción.

• SV: En la base de datos se encuentra disponible la voz cantada por separada (en la carpeta SOLO VOZ).

• SVA: En la base de datos se encuentra disponible la voz cantada por separada (en la carpeta SOLO VOZ) y además esta disponible por separado el acompañamiento musical (en la carpeta ACOMPAÑAMIENTO)

Apéndice H

Contenido del CD

Se entrega en forma adjunta con la documentación un cd con material, el cual puede resultar muy útil para comprender mejor el trabajo realizado, así como también para quien quiera profundizar en el tema.

- Base de Datos
  Se entrega la base de datos que se utilizó para ajustar los parámetros y testear el código. Está compuesta por fragmentos de corta duración. Los archivos que contiene fueron descritos en el apéndice B. Se divide en 4 partes:

  - Fragmentos de canciones.
    Estos fragmentos se utilizaron para probar el código implementado de punta a punta.
  - Fragmentos con la voz ya separada.
    Estos fragmentos se utilizaron para calcular el *pitch* con *WaveSurfer*, el cual se utiliza luego como una entrada más al sistema. También se utilizaron para comparar la forma de onda obtenida de la señal separada con la señal original. Los fragmentos se obtuvieron a partir de cds de *Karaoke*.
  - Fragmentos con el acompañamiento por separado.
    Los fragmentos con sólo el acompañamiento musical se utilizaron para realizar la evaluación del sistema implementado. Al igual que los fragmentos de sólo voz cantada se obtuvieron a partir de los cds de *Karaoke*.
  - *Pitches* calculados con *WaveSurfer*.
    Se adjuntan los *pitches* calculados con *WaveSurfer* para algunos fragmentos musicales. El cálculo del *pitch* utilizando *WaveSurfer* se explicó en el apéndice F.4.
  - Fragmentos de voz cantada separados
    Se proporcionan algunos resultados del algoritmo implementado

- Códigos
  Se adjunta el código completo en Matlab.

- Documentación
  Se adjunta la documentación final en formato *PDF*, también se adjuntan las imágenes utilizadas en formato *EPS* y los archivos fuente en LaTeX que generan el *PDF* final.

- Bibliografía
  Adjuntamos toda la bibliografía utilizada y también otros artículos y libros que se consultaron durante el proyecto.

- Artículo
  La versión *PDF* del articulo del proyecto en formato *IEEE*.

- Poster
  Poster del proyecto

- Software
  Se incluyen algunos programas utilizados en el transcurso del proyecto.

  - WaveSurfer
    Se incluye *WaveSurfer*, un editor de audio de código abierto. Se utilizó para calcular el *pitch* para señales de sólo voz cantada, también para convertir los archivos al formato adecuado (`MONO` $16KHz$) y cortar los fragmentos de las canciones originales. Se incluyen las versiones para Windows y para Linux.

  - Praat
    El software *Praat* permite trabajar con archivos de audio, entre muchas de sus funciones permite calcular el *pitch* para una señal de voz "limpia", sin interferencia ni acompañamiento musical.

  - Programa de Hu Wang [16]
    Se incluye el código y el ejecutable realizado por Hu-Wang que implementa lo explicado en [16]. El mismo separa voz hablada de interferencia.

  - Dev C++
    Programa bajo licencia GNU para programar en C y $C++$. El mismo comenzó a utilizarse durante el pasaje del código de Matlab a $C++$

Apéndice I

Comentarios sobre posible implementación

En este apéndice se presentan algunos detalles que se consideraron sobre una posible implementación.

I.1. Sobre el lenguaje de programación

Respecto al lenguaje de programación se considera lo siguiente:

• Java es un lenguaje orientado a objetos, portable, con manejo propio de recursos del sistema, y con un muy buen manejo de objetos, en particular Arrays y Arrays de tamaño dinámico (Arraylist), que pueden resultar muy útiles para el manejo de señales. Pero tiene una gran desventaja, la cual hace que no sea utilizado en tratamiento de señales: los archivos ejecutables, se encuentran "pre-compilados", lo que implica que a la hora de ejecución, deban ser traducidos al lenguaje de máquina. Este hecho, permite que las aplicaciones en Java sean multi-plataforma, a costas de una menor performance.

• C es un lenguaje de medio nivel [63], que permite manejar directamente la memoria y periféricos, orientado a procedimientos, el cual es ampliamente utilizado en aplicaciones que consumen un gran costo computacional. Por ejemplo es muy utilizado en el tratamiento de señales.

Luego de analizar las ventajas y desventajas de cada uno, se optó por una implementación en Java. Asimismo se decidió que los métodos que consumían grandes costos computacionales se iban a implementar en C. El lenguaje Java, contiene de la JNI (Java Native Interface), que es un framework de programación que permite llamar a métodos implementados en otros lenguajes, como C o C++, desde código Java corriendo en la Java Virtual Machine (JVM)[64].

El código realizado de esta manera resulta ser no portable, ya que a partir del código en C, es necesario generar unas librerías de enlace dinámico (dll) en Windows, o una librería compartida si es implementado en Linux.

I.2. Un posible framework de trabajo con Java y C

Un posible framework de trabajo, completamente parametrizable podría basarse en el siguiente diagrama:

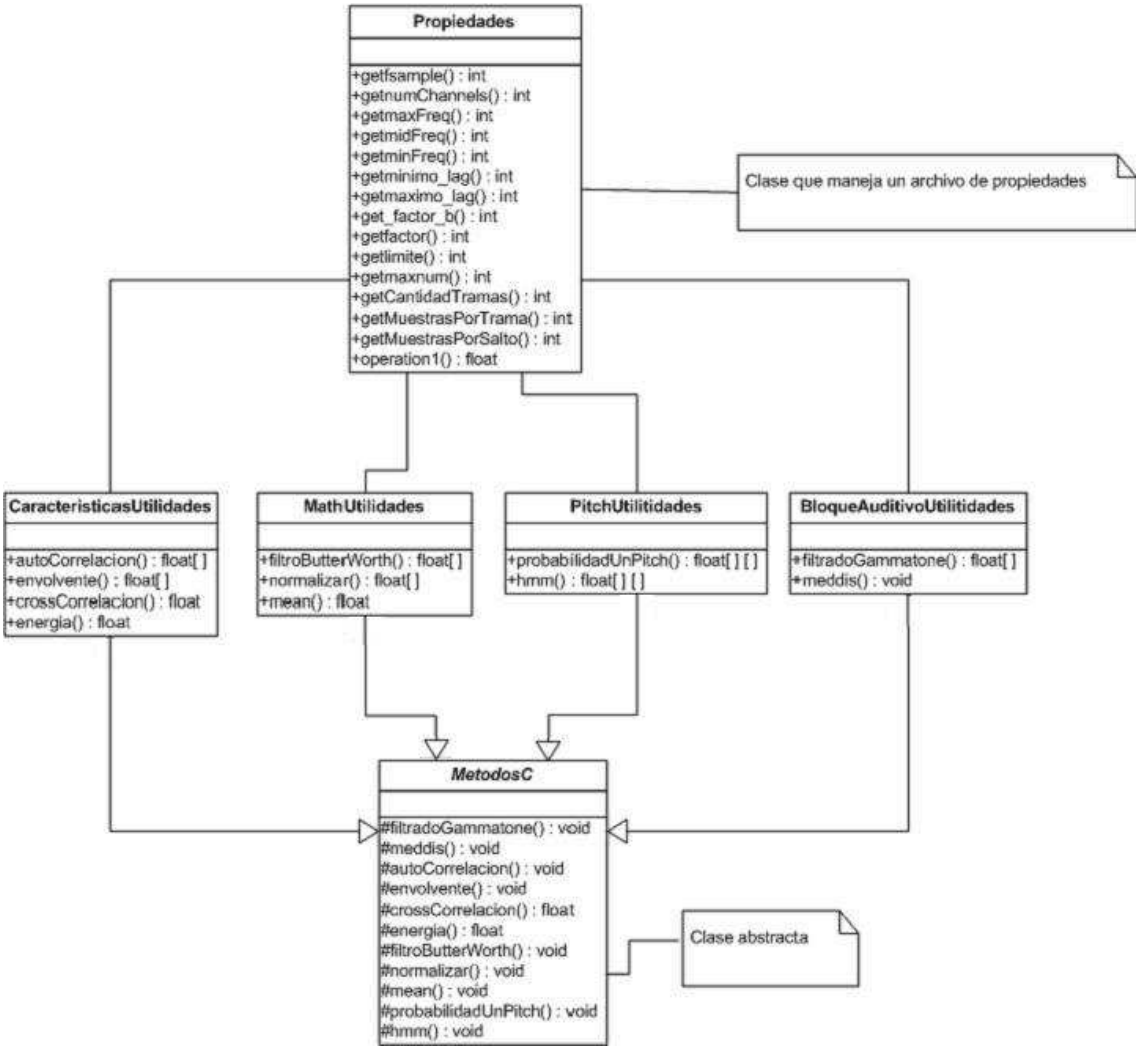

Figura I.1: En esta figura se puede ver que existe una clase *Metodos* abstracta, donde todos sus métodos son nativos y de accesibilidad protegida. También aparecen las clases *CaracteristicasUtilidades*, *MathUtilidades*, *PitchUtilidades*, *BloqueAuditivoUtilidades*, que heredan de *MetodosC*, y una clase, *propiedades*, la cual lee el archivo audio.properties, que contiene los parámetros definidos, haciendo que la aplicación sea parametrizable.

A partir de la clase *MetodosC* se genera el encabezado MetodosC.h, el cual se im-

plementa en C. Por más detalles se recomienda ver [65].

Cuando se definen los métodos nativos en *MetodosC*, se trata de definir la mayor cantidad de parámetros formales, de forma de hacer los métodos en C los más paramétricos posibles. Para trabajar en forma menos engorrosa con estos métodos, se utilizan las 4 clases *CaracteristicasUtilidades*, *MathUtilidades*, *PitchUtilidades*, *BloqueAuditivoUtilidades*, las cuales pretenden ser unas clases que agrupan lógicamente los métodos definidos en *MetodosC*, y hacen transparente la complejidad de la invocación. A continuación se muestran segmentos de código, que sirven a modo de ejemplo:

• En la clase *MetodosC* se define el método:

protected native void filtradoGammatone(float[] salida, float[] entrada,int fSample,float frecuenciaCentral);

• Como el filtrado Gammatone corresponde a la etapa de modelado auditivo, se define en *BloqueAuditivoUtilidades* el método:

public float[] getFiltradoGammatone(float[] entrada){

float[] salida = new float[entrada.length];

filtradoGammatone(salida,entrada,Propiedades.getfsample());

return salida;

}

De esta forma, se ve que se pueden invocar a los métodos necesarios implementados en C, de forma más fácil, considerando la cantidad de parámetros que se definen en los métodos de *MetodosC*[1]. Pero en lo que más facilita, considerando que los métodos en C no devuelven Arrays, es que de esta forma se simula una devolución de Arrays: se crea el Array en Java, se pasa la referencia al método implementado en C, en el cual, se le cargan los valores de salida y luego se retorna el Array creado.

I.3. Detalles del Manejo de Archivos de Audio

La JDK 1.4, cuenta con el paquete *javax.sound.sampled*, en donde están definidas varias clases que sirven para el manejo de archivos de audio. En particular se resaltan dos[2]:

• *javax.sound.sampled.AudioFormat*: Esta clase permite obtener la información necesaria para poder procesar correctamente el archivo de audio, por ejemplo, a partir de ésta podemos obtener el formato, codificación, canales, frecuencia de muestreo, bits por

[1]Esto permite que el código sea altamente parametrizable y reutilizable.

[2]Por más información, recurrir a la documentación proporcionada por SUN.

muestras, si es BigEndian o no, etc.

• *javax.sound.sampled.AudioInputStream*: Esta clase, permite el manejo de las muestras de audio. Contiene el método *read* en particular, que recibe un array de bytes, y devuelve el numero de bytes que se le cargan al array de bytes de entrada, así, contiene información del audio.

Un problema que se presenta aquí, es que la información de audio que se obtiene se devuelve como byte, por lo que sabiendo si es BigEndian o no, como está codificado, y bits por muestra (considerando que un byte tiene 8 bits), debemos obtener la representación en float, para poder realizar todas las operaciones presentadas en este documento.

Bibliografía

[1] Y. Li and D. Wang, "Separation of singing voice from music accompaniment for monaural recordings," *disponible WWW, http://www.cse.ohio-state.edu/ dwang/papers/Li-Wang.taslp07.pdf*, 2007.

[2] M. Rocamora, E. López, and G. Sosa, "Búsqueda de música por tarareo," *IIE, Facultad de Ingeniería, Universidad de la Republica Oriental del Uruguay*, 2004.

[3] R. L. C.K. Wang and Y. Chiang, "An automatic singing transcription system with multilingual singing lyric recognizer and robust melody tracker," *Proceedings of EUROSPEECH*, 2003.

[4] Y. Wang, M. Kan, T.Ñwe, A. Shenoy, and J. Yin, "Lyrically: automatic synchronization of acoustic musical signals and textual lyrics," *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pp. 212–219, 2004.

[5] A. Berenzweig, D. Ellis, and S. Lawrence, "Using voice segment to improve artist classifiction of music," *AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.

[6] Y. Kim and B. Whitman, "Singer identification in popular music recording using voice coding features," *Proceeding of International Conference on Music Information Retrieval*, 2002.

[7] E. Cherry, "Some experiments of recognition of speech, with one and with two cars.," *Journal of the acoustical society of america*, vol. 25, no. 5, pp. 975–979, 1953.

[8] E. Cherry, *On Human Communication*. MIT Press. Cambridge, MA., 1957.

[9] D. E. Chilton, "Speech analysis." University of Surrey, 1999.

[10] A. Bregman, *Auditory Scene Analysis*. MIT Press, 1990.

[11] D. Wang and G. Brown, *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*. Wiley-IEEE Press, 2006.

[12] G. Brown and D. Wang, "Separation of speech by computational auditory scene analysis," *Speech Enhancement*, pp. 371–402, 2005.

[13] P. Divenyi, *Speech Separation by Humans and Machines.* Norwell, MA: Kluver Academic, 2005.

[14] D. Rosenthal and H. Okuno, *Computational Auditory Scene Analysis.* Mahwah, New Jersey: Lawrence Erlbaum, 1998.

[15] A. Hu and D. Wang, "Monoaural speech segregation based on pitch tracking and amplitude modulation," *IEEE Transactions on Neural*, vol. 15, pp. 1135–1150, 2004.

[16] A. Hu and D. Wang, "An auditory scene analysis approach to monaural speech segregation," *Hansler E. and Schmidt G. (ed.)*, pp. 485–515, 2006.

[17] R. Patterson, Nimmo-Smith, J. Holdsworth, and P. Rice, "An efficient auditory filterbank based on the gammatone function," tech. rep., MRC Applied Psychology Unit, Cambridge, 1987.

[18] R. Meddis, "Simulation of auditory–neural transduction: Further studies," *Journal of the Acoustical Society of America*, vol. 83, pp. 1056–1063, 1988.

[19] R. Carlyon and T. Shackleton, "Comparing the fundamental frequencies of resolved and unresolved harmonics: Evidence for two pitch mechanisms?," *Acoustical Society of America Journal*, vol. 95, pp. 3541–3554, 1994.

[20] R. Plomp and M. Mimpen, "The ear as a frequency analyzer," *Acoustical Society of America Journal*, vol. 43, pp. 764–767, 1964.

[21] D. Wang, "On ideal binary mask as the computational goal of auditory scene analysis," *Speech Separation by Humans and Machines*, pp. 181–197, 2005.

[22] B. Moore, *An Introduction to the Psychology of Hearing.* San Diego, CA, USA: Academic Press, 5th ed., 2003.

[23] H. Helmholtz, *On the Sensation of Tone.* Braunschweig, Germany: Vieweg and Son, 1863.

[24] J. Bird and C. Darwin, "Effects of a difference in fundamental frequency in separating two sentences," *Psychophysical and Physiological Advances in Hearing*, 1997.

[25] Y. Li and D. Wang, "Detecting pitch of singing voice in polyphonic audio," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 17–20, 2005.

[26] M. Wu, D. Wang, and G. Brown, "A multipich tracking algorithm for noisy speech," *IEEE Transactions on Speech and Audio Processing*, vol. 11, pp. 229–241, 2003.

[27] G. Brown and M. Cooke, "Computational auditory scene analysis," *Comput.Speech Language*, vol. 8, pp. 297–336, 1994.

[28] D. Wang and G. Brown, "Separation of speech from interfering sounds based on oscillatory correlation," *IEEE Trans. Neural Networks*, vol. 10, pp. 684–697, 1999.

[29] B. Glasberg and B. Moore, "Derivation of auditory filter shapes from notched–noise data," *Hearing Research*, vol. 47, pp. 103–138, 1990.

[30] R. Meddis, M. Hewitt, and T. Shackleton, "Implementation details of a computational model of the inner hair-cell/auditory-nerve synapse," *Journal of the Acoustical Society of America*, vol. 87, pp. 1813–1816, 1990.

[31] D. Wang, "Matlab toolbox for cochleagram analysis and synthesis," *http://www.cse.ohio-state.edu/pnl/shareware/cochleagram/*, 2007.

[32] J. Holdsworth, Nimmo-Smith, R. Patterson, and P.Rice, "Implementing a gammatone filter bank," tech. rep., MRC Applied Psychologi Unit, Cambridge, 1988.

[33] C. Urrutia, "algoritmo para la deteccion de pitch en polifonia en tiempo real," *Departamento de Ingeniería Eléctrica,Universidad catolica de chile*, vol. 5, 2004.

[34] A. Oppenheim, R. Shafer, and J. Buck, *Tratamiento de Señales en Tiempo Discreto.* Prentice Hall, 2000.

[35] J. Kaiser, "On a simple algorithm to calculate the 'energy' of a signal," *Proc. IEEE Internat. Conf. Acoust. Speech Signal Process*, pp. 381–384, 1990.

[36] P. Maragos, J. Kaiser, and T. Quatieri, "On amplitude and frequency demodulation using energy operators," *IEEE Trans. Signal Processing*, vol. 41, pp. 1532–1550, 1993.

[37] K. Sjöl and J. Beskow, "Wavesurfer- an open source speech tool," *http://www.speech.kth.se/wavesurfer.*

[38] P. Boersma and D. Weenink, "Praat: doing phonetics by computer," *http://www.fon.hum.uva.nl/praat.*

[39] "The scientist and engineer´s guide to digital.,"

[40] "Wavelets in real time audio processing," *Master´s thesis, Maihem University*, 2000.

[41] "Discrete time processing of speech signals," *IEEE Press*, 2000.

[42] D. Hand and K. Yu, "Idiot´s bayes - not so stupid after all?," *International Statistics Review*, vol. 69, no. 3, pp. 385–398, 2001.

[43] F. Jelinek, *Staticals Methods for Speech Recognition.* MIT Press, 1997.

[44] A. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Transactions on Speech and Audio Processing*, 2003.

[45] M. Goto, "A predominant-f0 estimation method for polyphonic musical audio signals," *Proceedings of the 18th International Congress on Acoustics (ICA 2004)*, 2004.

[46] D. Brungart, P. Chang, B. Simpson, and D. Wang, "Isolating the energetic component of speech-on-speech masking with ideal time-frequency segregation," *Journal of the Acoustical Society of America*, vol. 120, pp. 4007–4018, 2006.

[47] P. Chang, "Exploration of behavioral, physiological, and compuetational approaches to auditory scene analysis," Master's thesis, The Ohio State University, Deparment of Computer Science and Engineering, 2004.

[48] N. Roman, D. Wang, and G. Brown, "Speech segregation based on sound localization," *Journal of the Acoustical Society of America*, vol. 114, no. 4, pp. 2236–2252, 2003.

[49] A. Ozerov, P. Philippe, R. Gribonval, and F. Bimbot, "Adaptation of bayesian models for single-channel source separation and its application to voice/music separation in popular songs," *IEE Workshop on Application of Signal Processing to Audio an Acoustics*, 2007.

[50] G. Hu and D. Wang, "Segregation of unvoiced speech from nonspeech interference," *Journal of the Acoustical Society of America*, 2008.

[51] I. H. Tobias Oetiker, Hubert Partl and E. Schlegl, "The not so short introduction to latex 2," tech. rep., Junio 2007.

[52] F. Miyara, *La Voz Humana*. Universidad Nacional de Rosario.

[53] D. Maggiolo, "Sistema auditivo," *http://www.eumus.edu.uy/docentes/maggiolo/acuapu/sau.html*.

[54] J. Arribas, "Sistema auditivo," *http://www.lpi.tel.uva.es*.

[55] Y. Kim, "Singing voice analysis/synthesis," 2003.

[56] S. Vembu and S. Baumann, "Separation of vocals from polyphonic audio recordings," 2005.

[57] B. Raj, P. Smaragdis, M. Shashanka, and R. Singh, "Separating a foreground singer from background music," 2006.

[58] A. Ozerov, P. Philippe, R. Gribonval, and F. Bimbot, "One microphone singing voice separation using source-adapted models," *IEE Workshop on Application of Signal Processing to Audio an Acoustics*, 2005.

[59] B. Raj and P. Smaragdis, "Latent variable decompotition of spectrogrms for single channel speaker separation,"

[60] D. Maggiolo, "apuntes de acústica musical - bandas crÍticas," *disponible WWW, http://www.eumus.edu.uy/docentes/maggiolo/acuapu/bcr.html*, 2007.

[61] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition.," *Proceedings of IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[62] Wikipedia.Org, "Algoritmo de viterbi,"

[63] Wikipedia.Org, "Lenguaje de programación c,"

[64] Wikipedia.Org, "Java native interface,"

[65] Wikipedia.Org, "Utilizar el java native interface,"