Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México

Presentado por:

Jorge Alberto TREVIÑO LÓPEZ
Patricia Isabel ORTAL VITE

Reconocimiento de voz esofágica empleando redes neuronales artificiales

---

Presentado el día 23 de Noviembre del 2005

Asesor:
M. en C. Alfredo Mantilla Caeiros *Profesor ITESM-CCM*

Sinodales:
Dr. Ricardo Fernández del Busto y Ezeta *Profesor ITESM-CCM*
Dr. Jorge Brieva Rico *Profesor ITESM-CCM*

Profesor:
Dr. Francisco Javier Cuevas Ordáz *Profesor ITESM-CCM*

Trabajo efectuado en el departamento de Ingeniería Eléctrica y Electrónica del Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Ciudad de México

Índice

Resumen

Este proyecto busca contribuir a la mejora en la calidad de vida de las personas que han sufrido una laringectomía. Como parte de un sistema que pretende aumentar la inteligibilidad de la voz esofágica, se propone un algoritmo para la identificación de los fonemas vocálicos del español. El mismo es diseñado partiendo de los rasgos más generales del oído humano.

Al definir un método para lograr el reconocimiento de voz esofágica, fue necesario recurrir a conceptos provenientes de distintas disciplinas como la anatomía, la fonética, la psicoacústica, el análisis funcional y los métodos para la clasificación de patrones. La convergencia de estas ideas, permitió la aplicación de técnicas con fundamentos sólidos y ampliamente utilizadas, como la transformada wavelet y las redes neuronales artificiales.

Se propusieron modificaciones convenientes a los procedimientos usuales y se demostró la validez de las mismas. Como resultado, se presenta un análisis en resoluciones múltiples que incorpora algunos de los fenómenos psicoacústicos más representativos del oído interno.

Finalmente, se implantó el algoritmo bajo el ambiente de Matlab para evaluar su desempeño al operar sobre datos reales.

Introducción

La comunicación es el proceso por el cual dos o más seres humanos intercambian información, ésta ha sido fundamental en el desarrollo de las sociedades, de la tecnología e incluso importante para la evolución humana. Existen diferentes formas de comunicarse como escrita o gestual, pero la más común y usada por los seres humanos es la oral.

Para que nosotros podamos comunicarnos oralmente, necesitamos de dos factores importantes, al menos una persona que hable y al menos una persona que escuche.

El emisor debe tener entre otras la capacidad de hablar y esto lo logramos normalmente gracias a nuestro aparato respiratorio, centros nerviosos específicos del control del habla situados en la corteza cerebral, centros de control respiratorio en el encéfalo, estructuras de articulación y resonancia dentro de la boca y las cavidades nasales. El habla es producida gracias a dos funciones mecánicas. La fonación que sucede dentro de la laringe y la articulación que sucede dentro de la boca.

Como seres vivientes, nos encontramos expuestos a diferentes trastornos que pueden dañar nuestro organismo con acciones comunes y de la vida diaria. Una de las enfermedades más peligrosas y de las que más se habla hoy en día es el cáncer. El cáncer se presenta como un crecimiento no regulado y anormal de células las cuales al cabo de un tiempo, destruyen tejido y se extienden hasta otras partes del cuerpo. Esta enfermedad no es contagiosa, pero se puede contraer por diversos factores sin importar la edad o el género. Las causas del cáncer son diversas, desde ocupacionales como trabajar en fundadoras, cocinar con carbón o leña, exponerse a los rayos ultravioletas del sol, hasta debidos a algunas adicciones como tabaquismo o alcoholismo.

El cáncer puede presentarse en cualquier lugar del cuerpo, y su foco generalmente se encuentra en órganos importantes. Una parte del cuerpo que se puede ver afectada por la enfermedad del cáncer es la laringe. Este tipo de cáncer se encuentra comúnmente en pacientes de entre 50 y 70 años. Afortunadamente en la actualidad se cuentan con diversos medios para contrarrestar este padecimiento. Algunos de estos tratamientos son por medio de quimioterapia, radiaciones o tratamientos con láser, pero cuando estos métodos no son efectivos y el cáncer en la laringe es avanzado, se debe practicar una laringectomía.

La laringectomía consiste en la extirpación de la laringe y la separación del ducto de aire para respirar por medio de la boca, nariz y el esófago, esto impide la posibilidad de usar el aire proveniente de los pulmones para producir voz. Cuando la laringe es extirpada, se pierden las cuerdas vocales y la capacidad de hablar por medio de ellas, pero no por esto se pierde la capacidad para hablar definitivamente.

Como se comentó anteriormente, el habla en el ser humano es básica para su desarrollo dentro de una sociedad, esto ha provocado que se desarrollen diferentes técnicas que le permitan a un paciente que ha sufrido de una laringectomía, el poder seguir comunicándose oralmente.

En la actualidad se disponen de tres técnicas para conseguir este objetivo. El método tranqueoesofágico, la laringe electrónica y la voz esofágica.

El método traqueoesofágico consiste en realizar una cirugía en la que se coloca una prótesis que le brinda al paciente comunicación entre la traquea y esófago. Esto se logra gracias a una válvula que el paciente cubre y descubre para forzar al aire de los pulmones hacia su boca y así poder hablar. Este método no se puede practicar con todos los pacientes y además tiene la característica de que es difícil de dominar.

El siguiente método es el que hace el uso de una laringe electrónica. La laringe electrónica es un dispositivo que produce voz de forma electromecánica, es uno de los métodos más fáciles de usar, y por lo tanto el más común. Cuando un paciente quiere comunicarse, se coloca en el cuello este dispositivo y comienza a hablar. El dispositivo genera una excitación en el tracto vocal sustituyendo de esta manera a la laringe natural. La transducción de las vibraciones mecánicas en eléctricas es el procedimiento que genera voz con características electrónicas. La desventaja de este método se encuentra principalmente en que la voz que se produce posee la peculiaridad de ser monótona y artificial lo que provoca algunas dificultades al momento de ser entendida.

La última técnica que se utiliza para corregir este padecimiento es la producción de voz esofágica. Esta técnica consiste en el entrenamiento de los pacientes para que puedan generar esta voz. Esto lo logran insuflando aire desde la nariz y boca hasta el esófago ahí lo retienen y luego lo expulsan produciendo vibraciones que originan la voz.

Las ventajas de este método son que no se requiere ninguna intervención quirúrgica ni la necesidad de un dispositivo que se deba portar cada vez que el paciente quiera comunicarse. Su desventaja principal radica en una larga rehabilitación que le permita al paciente comunicarse por medio de voz esofágica ya que no es algo que hacemos de manera natural.

Por otra parte, la voz esofágica presenta características que, en algunos casos, dificultan su entendimiento. Esto se debe en su mayor parte a la falta de intervención por parte de las cuerdas vocales durante su producción. Lo anterior trae como consecuencia que algunos sonidos como los fonemas vocálicos posean una menor claridad en la voz esofágica.

A fin de mejorar la calidad de vida de los pacientes laringectomizados, se ha planteado el desarrollo de sistemas capaces de identificar las regiones de menor inteligibilidad en la voz esofágica y reemplazarlas por voz sintetizada electrónicamente. Lo anterior presenta, a muy grandes rasgos, dos problemas a solucionar. El primero se refiere al reconocimiento

de los fonemas presentes en un fragmento de voz esofágica. El segundo consiste en el desarrollo de un algoritmo para la síntesis de voz.

Este trabajo ofrece una solución para la etapa de reconocimiento basada en el uso de la transformada wavelet y redes neuronales artificiales. Se postula que la voz evolucionó para que ésta fuese fácil de producir y de reconocer por los seres humanos. Tomando esto en consideración, se desarrolló un algoritmo que imitara las características más generales del oído humano.

Al no perder de vista el objetivo principal de contribuir a la mejora de la comunicación oral para pacientes laringectomizados, el alcance de este trabajo se encuentra limitado al reconocimiento de las cinco vocales del español en voz esofágica. Esta decisión se debe no sólo a la importancia de los fonemas vocálicos, sino a que estos últimos son también los más afectados por la pérdida de las cuerdas vocales.

El desarrollo del presente proyecto involucró el uso de técnicas tratadas con frecuencia en otros trabajos sobre el reconocimiento de voz, como las redes neuronales artificiales. Asimismo, se recurre a métodos que no han sido aprovechados por completo en esta área, como es la transformada wavelet. Para la técnica mencionada, se propone una modificación a su versión discreta a fin de obtener resultados de una mayor calidad comparados con los que ofrecerían los algoritmos habituales. Por último, y siguiendo la premisa acerca de la evolución del habla, se define un wavelet basado en el modelado del oído interno.

El algoritmo propuesto es implantado en Matlab para su validación. Durante su programación se tuvo presente la posible necesidad de variar los parámetros más importantes del mismo para que el desempeño global no sea afectado por cambios en el locutor, el ruido o alguna otra variable ambiental.

Finalmente, se evalúa el desempeño del sistema para el reconocimiento de las vocales en hablantes normales y en voz esofágica.

Objetivos

- Contribuir a la mejora de la calidad de vida de aquellas personas que han sufrido una laringectomía.

- A largo plazo, se pretende desarrollar un sistema capaz de aumentar la inteligibilidad de la voz esofágica. Para lograrlo, se ha propuesto reemplazar las secciones vocalizadas del habla de un paciente con laringectomía por voz sintetizada en una computadora.

- Desarrollo e implantación de un algoritmo para la separación y posterior clasificación (basada en el uso de una red neuronal artificial) de las secciones vocalizadas de una señal de voz esofágica.

- Agrupar los segmentos vocalizados de dicha señal dentro de 5 clases definidas a partir de los fonemas vocálicos del español (A, E, I, O y U). Cada una de estas clases debe ser construida de modo que incluya a todos los segmentos que en su mayor parte presenten algún alófono del fonema que la identifica.

- Programación de una interfaz gráfica que facilite el entrenamiento del sistema. Este adiestramiento es requerido debido a que la producción del habla es un proceso estocástico, y como consecuencia del método heurístico empleado para la clasificación de cada segmento de voz.

Justificación

Luego de una laringectomía, los pacientes pierden la capacidad de hablar empleando los medios normales debido a la imposibilidad de hacer fluir aire desde los pulmones, a través del aparato fonador, hasta la boca.

La calidad de vida de estos pacientes y de quienes los rodean se ve deteriorada debido a la dificultad para comunicarse. Existen pocas opciones para superar este tipo de discapacidad como la laringe electrónica y la terapia para la producción de voz esofágica. Estas alternativas le permiten al paciente comunicarse de manera oral, sin embargo su dicción es de una claridad inferior a la voz de una persona normal.

Este trabajo pretende contribuir en el desarrollo de un sistema capaz de mejorar las características acústicas de la voz esofágica, facilitando así la comunicación oral de los pacientes que han sufrido una laringectomía.

Al explorar métodos para la extracción de características significativas de una señal de voz que no han sido estudiados a fondo en trabajos previos, el presente proyecto podrá ser utilizado además como la base para el desarrollo de nuevos algoritmos para el reconocimiento de voz.

Capítulo I

Anatomía de la voz y del sistema auditivo humano

1.1 Introducción: "Anatomía y fisiología de la voz"

Los seres humanos generamos la voz por medio de las cuerdas vocales, el aire que fluye desde los pulmones y un proceso de filtrado producido por las cavidades resonantes del tracto vocal. Es considerada como el instrumento musical más versátil debido a que las cuerdas vocales en combinación con el tracto vocal y los articuladores son capaces de producir prácticamente cualquier tono.

A través de la voz, los seres humanos somos capaces de producir sonidos con frecuencias en un rango desde los 100 Hz hasta los 10 kHz. Sin embargo, en una señal de voz tanto la información verbal como la sección más significativa del espectro de la voz se encuentran contenidas a partir de los 300 Hz hasta los 4 kHz. Tomando en cuenta únicamente la inteligibilidad de la voz, entonces el rango de frecuencias que contendría esta información sería desde los 500 Hz a los 2.5 kHz.

Una de las propiedades importantes de la voz es el tono o pitch. El tono de la voz se refiere al número de veces por segundo que las cuerdas vocales se unen durante la fonación, la fonación se refiere a la acción de producir sonido. Los hombres presentan un tono cerca de 110 ciclos por segundo, mientras que las mujeres manifiestan un tono de 180 a 220 ciclos por segundo.

La intensidad de la voz es otra propiedad que depende de que tan juntas estén las cuerdas vocales entre si, de la cantidad de presión de aire por debajo de la laringe, de la frecuencia fundamental de la voz y la resonancia de la estructura del tracto vocal. El tracto vocal esta formado por la garganta, la cavidad dentro de la boca y los pasajes nasales.

Además del tono y la intensidad, otra propiedad importante del sonido es el timbre. El timbre es lo que nos permite diferenciar a un sonido de otro aunque estos tengan el mismo tono e intensidad. Esta propiedad está determinada principalmente por el contenido harmónico y las dinámicas características del sonido.

Debido a las diferentes dimensiones del tracto vocal ocurren resonancias acústicas llamadas formantes. Las formantes son las frecuencias en donde las harmónicas tienen mayores amplitudes.

La voz de los hombres y las mujeres es distinta debido a factores físicos como el tamaño de la laringe y el espacio que existe entre las cuerdas vocales. Las mujeres presentan en su voz un tono casi una octava más alto comparado con el de los hombres. Esto se debe a que las cuerdas vocales en las mujeres vibran casi dos veces más por segundo. Fisiológicamente, la laringe de las mujeres muestra por lo general una anatomía distinta en el espacio que existe detrás de las cuerdas vocales formado por el glotis cartilaginoso.

1.2 Producción de la voz

En cuanto a la producción de la voz, ésta depende de las vibraciones de las cuerdas vocales y la resonancia.

El sonido más básico producido por la vibración de las cuerdas vocales es el sonido vocalizado. El sonido vocalizado es amplificado y modificado por el tracto vocal y por los articuladores del tracto vocal. El tracto vocal y los articuladores del tracto vocal dependen de las características físicas de cada persona, esa es la razón por la que estos son los que le dan características distinguibles a la voz de cada una de las personas y hacen las palabras inteligibles.

La voz es producida, como ya se dijo, cuando el aire dentro de los pulmones se dirige hacia las cuerdas vocales. Las cuerdas vocales a su vez vibran por la acción de nervios, cartílagos y la laringe. La laringe es una estructura cilíndrica formada por cartílagos que sirve para mantener cerradas las cuerdas vocales hasta que son forzadas a separarse nuevamente. Un ciclo vibratorio de las cuerdas vocales comienza cuando la presión del aire las separa, el aire continúa moviéndose hacia arriba y se crea un espacio de baja presión debajo de la columna de aire, debido al principio de Bernoulli, las cuerdas vocales se cierran. La oclusión de las cuerdas vocales corta el aire liberando un pulso y finalmente el ciclo se repite. Al tiempo que transcurre entre cada uno de esos pulsos se le llama periodo fundamental. Estos pulsos de aire con comportamiento cuasi-periodico son los que producen el sonido vocalizado. Por otra parte, si las cuerdas vocales están por completo abiertas, no se produce voz. Esto ocurre cuando respiramos normalmente y durante la producción de algunos sonidos consonantes. Al sonido resultante, se le llama sonido no vocalizado.

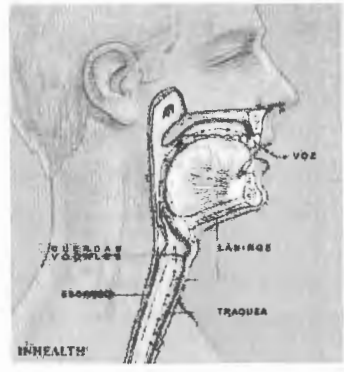


Figura 1.2.1. Anatomía del aparato fonador

1.3 Fonética acústica

El punto de articulación de una vocal o consonante se refiere al lugar de las cavidades supraglóticas donde se produce la articulación de una unidad de sonido o fonema. De acuerdo con los diferentes puntos de articulación se pueden distinguir los siguientes fonemas:

Consonantes

Bilabiales: Contactan los labios superiores e inferiores.
Linguodentales: Contacta el ápice de la lengua con los incisivos superiores.
Labiodentales: Contacta el labio inferior con los incisivos superiores.
Linguointerdentales: Se sitúa el ápice de la lengua entre los incisivos superiores e inferiores.
Linguoalveolares: Contacta el ápice o predorso de la lengua con los alvéolos.
Lingupalatales: Contacta el predorso de la lengua con el paladar duro.
Linguovelares: Se aproxima o toca el postdorso de la lengua con el velo del paladar.

Vocales

Anteriores: La lengua se aproxima a la región delantera o zona del paladar duro.
Centrales: La lengua se encuentra en la parte central del paladar.
Posteriores: La lengua se aproxima a la zona velar.

Los distintos modos de articulación utilizados en el lenguaje español son:

Consonantes

Oclusivas: Se establece un cierre completo de los órganos articulatorios y el aire sale de forma explosiva tras la interrupción.
Fricativas: Existe un estrechamiento de dos órganos articulatorios donde pasa el aire espirado.
Africadas: Se forma por combinación de una oclusiva seguida de una fricativa.
Laterales: Durante su emisión el aire se escapa por un lado o por los dos de la lengua.
Vibrantes: Se produce una o varias vibraciones del ápice de la lengua.

Vocales

Cerradas: La lengua se encuentra muy cerca del paladar.
Medias: La lengua está en una distancia intermedia del paladar.
Abiertas: La lengua se separa totalmente del paladar.

Finalmente, los sonidos también pueden ser clasificados según la intervención que tiene el velo del paladar durante su producción. Estos pueden ser:

Nasales: Cuando el velo del paladar se encuentra separado de la pared faríngea.

Orales: Cuando el velo del paladar se encuentra separado de la pared faríngea y no permite el paso del aire hacia la cavidad nasal.

1.4 Voz esofágica

La voz esofágica es uno de los procedimientos más comunes que se siguen para recuperar la voz después de una laringectomía. En esta operación, los pacientes pierden la capacidad de comunicarse oralmente, además de los sentidos del gusto y el olfato. La laringectomía total, es una técnica quirúrgica que se practica en pacientes que han desarrollado tumores malignos en la laringe y que consiste en la extracción de ésta y la sutura directa de la tráquea a la piel. Esta sutura implica un traqueostoma que permanecerá abierto para permitir la respiración.

La erigmofonación o voz esofágica se produce comprimiendo (con la ayuda de la lengua) y posteriormente deglutiendo el aire contenido en el tracto vocal. La voz se genera cuando la corriente de aire pasa por el segmento faringo-esofágico y provoca la vibración del músculo cricofaríngeo.

El sonido generado por la voz esofágica se caracteriza por ser de un tono promedio bajo, un timbre generalmente ronco, gran perturbación entre ciclos y una intensidad promedio baja. Sin embargo, los patrones de voz de un paciente laringectomizado se consideran similares a los de un hablante normal.

Para los pacientes que decidieron optar por el método de rehabilitación de voz esofágica, el proceso de aprendizaje consiste en hacer consciente a la persona del uso de la parte superior del esófago. Posteriormente se le enseña al paciente a producir las letras t, c, p, k, junto con las vocales a, e, i, o y u, por ser los fonemas más fáciles de trabajar. Estos ejercicios facilitan la articulación de las sílabas en principio, y posteriormente de frases hasta lograr construir las oraciones.

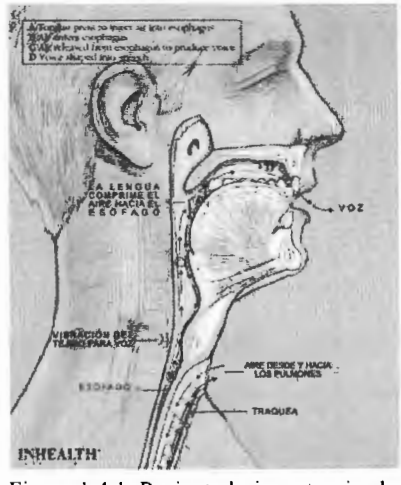


Figura 1.4.1. Paciente laringectomizado.

1.5 Introducción: "Anatomía del sistema auditivo humano"

El problema del reconocimiento de voz puede abordarse desde distintos enfoques. Algunos trabajos buscan identificar el estado del aparato fonador del locutor a partir de las características espectrales de su voz. Con base en esta información, tratan de determinar el contenido fonético de los datos analizados. A diferencia del método anterior, este proyecto busca extraer características de las señales de audio similares a las presentadas por el oído hacia el cerebro en el sistema auditivo humano. En este apartado se presenta una breve descripción de la fisiología del oído. Asimismo, se introduce uno de los principios psicoacústicos de mayor importancia: las bandas críticas.

Como la mayoría de los elementos biológicos, el sistema auditivo humano presenta una gran complejidad. Éste cumple la tarea de proporcionarnos información acerca del sonido, es decir, los cambios en la presión atmosférica debidos a la vibración de un cuerpo. Para llevar a cabo lo anterior, el sistema auditivo se encarga de detectar las oscilaciones que viajan por el aire y convertirlas en impulsos nerviosos que más tarde son interpretados por el cerebro.

De una manera más específica, el oído se encarga de recibir los cambios en la presión y de convertir esta variable física en otra capaz de excitar directamente las terminaciones del nervio auditivo. Esta fibra, parte del sistema nervioso periférico, transporta el estímulo hacia la corteza primaria auditiva. Es en esta región del cerebro, ubicada entre los lóbulos frontal y temporal, donde el impulso nervioso disparado inicialmente en el oído es interpretado como sonido.

Continuando con la división planteada, se observa que el cerebro lleva a cabo las tareas de mayor complejidad como la identificación de la fuente de distintos sonidos o la comprensión de un mensaje oral. Esto lo realiza a partir de la información que el nervio auditivo transporta desde el oído. De modo que este último, al cambiar la representación de una señal audible (comenzando con vibraciones sonoras y terminando con estímulos sensados por el nervio auditivo), preserva las características adecuadas de ésta para realizar los distintos análisis que puede efectuar el sistema nervioso central.

Teniendo en mente lo anterior, y considerando que una de las tareas más importantes del sistema auditivo es permitir la comunicación oral, se estudiará ahora la fisiología del oído para aproximar el mapeo que éste realiza partiendo de las oscilaciones transportadas por el aire.

En general, el oído se puede dividir en tres secciones: el oído externo, medio e interno. Cada una de ellas presenta dinámicas que modifican ciertas características de la señal audible. A continuación se presentan los puntos más relevantes de cada una de ellas.

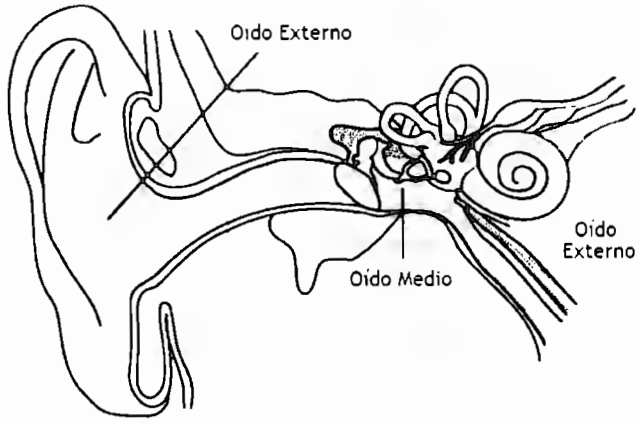


Figura 1.5.1 Anatomía del oído humano

1.6 Oído externo

El oído externo se encuentra formado por la aurícula, el canal auditivo y termina con la membrana externa del tímpano.

La aurícula es la única parte visible del sistema auditivo y se encarga de recibir las vibraciones transportadas por el aire y canalizarlas hacia el interior del oído. Posee una estructura helicoidal y presenta múltiples surcos y depresiones. Ésta permite al oído captar una mayor cantidad de sonidos al reducir las pérdidas por reflexión y difracción. Asimismo, su forma le otorga características acústicas que modifican las vibraciones sonoras. Lo anterior permite al sistema nervioso central identificar si la fuente del sonido se encuentra ubicada al frente o detrás.

El canal auditivo es un conducto que une la aurícula con el tímpano. Tiene una longitud de entre 25 y 35 milímetros. Desde el punto de vista acústico, este canal presenta una frecuencia de resonancia cercana a los cuatro kiloHertz [1]. Lo anterior modifica las características espectrales de las señales audibles, amplificando sus componentes de frecuencias cercanas al valor mencionado.

Al final del canal auditivo se encuentra el tímpano. Éste marca la división entre el oído externo y el oído medio. El tímpano consta de tres capas. La primera es un recubrimiento de piel similar al que presenta el canal auditivo. La segunda es una membrana elástica gracias a la cual el tímpano es capaz de convertir los cambios de presión presentes en el oído externo en vibraciones mecánicas que se transportan hacia el oído medio. Finalmente, su capa interna es una estructura mucosa consistente con las paredes del oído medio.

1.7 Oído medio

Limitando con el tímpano, comienza el oído medio. Éste se encuentra constituido esencialmente por los tres huesos más pequeños del cuerpo humano, a los que comúnmente se les denomina martillo, yunque y estribo.

El martillo se encuentra unido al tímpano, de modo que ambos se mueven simultáneamente como consecuencia de las vibraciones sonoras en el oído externo. El martillo a su vez está conectado al yunque, al cual se une el estribo. Finalmente el estribo está en contacto con la ventana oval, la cual marca el límite con el oído interno.

La función principal del oído medio es la de transportar las vibraciones mecánicas del tímpano hacia el oído interno de manera eficiente. Mientras el oído externo conduce vibraciones presentes en el aire, el oído interno se encuentra lleno de un fluido de mayor viscosidad. El oído medio es entonces en encargado de acoplar ambos elementos evitando pérdidas de energía. Esto se consigue gracias a la diferencia en las longitudes del martillo y el yunque (siendo mas largo el martillo), lo cual permite a ambos huesos trabajar como una palanca. Además, el área de la unión entre el estribo y el oído interno es menor al área del tímpano; esto provoca que la presión ejercida sobre la ventana oval sea mayor a la que los cambios de presión producen al inicio del oído medio.

Una segunda función del oído medio es la de brindar una cierta protección al oído interno para sonidos de amplitudes altas. Lo anterior se logra gracias a dos músculos que se contraen como reflejo a sonidos de intensidades mayores a los 75 decibeles. Estos tejidos se encuentran unidos al tímpano y al estribo y su contracción vuelve más rígidos los enlaces entre los tres huesos produciendo así una atenuación de la vibración transmitida hacia el oído interno.

1.8 Oído interno

El oído interno es la última etapa del sistema auditivo anterior al sistema nervioso. En él se encuentra la cóclea, la cual está unida al oído medio a través de su ventana oval.

La cóclea es un conducto que forma una espiral de aproximadamente 2.75 vueltas. En su interior, ésta se encuentra dividida por dos formaciones: la membrana de Riessner y la membrana basal. La cóclea contiene un líquido que transporta las vibraciones mecánicas producidas por el estribo sobre la ventana oval. Al ser incompresible, este fluido produce el movimiento de una segunda estructura llamada ventana circular. Durante este proceso, las ondas que viajan por el interior de la cóclea afectan a la membrana de Riessner y a la membrana basal.

La membrana basal no es uniforme. Su espesor es más pequeño en un extremo de la cóclea y se va incrementando hasta llegar al otro. Dado que las estructuras pequeñas responden mejor a frecuencias altas, mientras que los cuerpos grandes son afectados en mayor medida por las frecuencias bajas, el movimiento de la membrana basal no será igual a todo su largo. Las posiciones en que ocurra un desplazamiento máximo de la membrana basal estarán entonces directamente relacionadas con las frecuencias presentes en las vibraciones provenientes del oído medio. Esto es análogo a un análisis espectral. Es decir que, en el oído humano, la membrana basal es la responsable de descomponer la señal audible en sus distintas componentes de frecuencia.

Si el oído es estimulado por un tono puro, la respuesta de la membrana basal tendrá una posición dentro de la misma donde el desplazamiento sea máximo. Del mismo modo, si nos concentramos en una sección de la membrana basal y observamos su respuesta ante estímulos de distintas frecuencias, encontraremos que ésta es máxima para una frecuencia particular.

La membrana basal se encuentra cubierta en un lado por el órgano de Corti. Éste consiste en un conjunto de vellos que, al ser doblados, disparan un impulso nervioso transmitido al cerebro por el nervio auditivo. Estas células capilares, al estar unidas a la membrana basal, son afectadas directamente por los movimientos de alguna de las secciones de ésta última. El cerebro entonces, al conocer la posición del vello afectado por un estímulo audible, recibe del oído la información de las componentes espectrales presentes en el sonido percibido.

1.9 Bandas críticas

Como se vio anteriormente, si el oído es estimulado con un tono puro las distintas regiones de la membrana basal responderán en proporciones diferentes. Será posible encontrar una sección donde la membrana sufra un desplazamiento máximo; sin embargo, las regiones cercanas a ésta también se verán afectadas (aunque en menor medida) por el estímulo.

Si ahora se tiene una señal audible que contiene a dos o más frecuencias, la respuesta de la membrana basal será la superposición o suma de los efectos de cada uno de sus componentes. En este caso, si se tienen dos constituyentes espectrales, cada uno afectará a una posición particular de la membrana basal y a sus alrededores. Si ambos se encuentran lo suficientemente cerca, será imposible distinguirlos como tonos independientes.

El fenómeno descrito anteriormente nos indica que el oído no es capaz de identificar con precisión cada una de las frecuencias presentes en un estímulo. Resulta entonces deseable conocer una aproximación para la resolución espectral del oído humano a fin de entender mejor la información que recibe la corteza primaria auditiva. Lo anterior otorga una mayor comprensión sobre como escuchamos los diferentes sonidos. Específicamente para este trabajo, una aproximación de este tipo nos dará uno de los parámetros más importantes al momento de modelar la descomposición que lleva a cabo el oído interno sobre las señales auditivas que lo estimulan.

A fin de conocer la resolución aproximada del oído, se definió el concepto de ancho de banda crítico. Éste puede resumirse como la mínima diferencia necesaria en la frecuencia de dos tonos puros para que estos sean distinguidos como sonidos independientes. Una manera sencilla de entender el concepto anterior consiste en el siguiente experimento: Supongamos que el oído de una persona es estimulado por la suma de dos señales senoidales. Si inicialmente ambas poseen la misma frecuencia, naturalmente serán percibidas como un tono puro. Si una señal se mantiene fija, mientras se varía la frecuencia de la otra, eventualmente el escucha detectará una serie de pulsaciones. A pesar de esto,

será incapaz de distinguir ambos tonos por separado. Conforme la diferencia en frecuencias continúa aumentando, el periodo de las pulsaciones disminuirá. El siguiente cambio que notará el individuo será cuando el estímulo sea escuchado como un tono áspero, es decir, parecerá de una calidad inferior. Si se incrementa aún más la diferencia en las frecuencias de ambas señales, llegará un momento en el que sean percibidas como dos tonos independientes, aunque aún con cierta interferencia entre ambos. Finalmente, la diferencia en frecuencias para la cual el estímulo puede ser distinguido como dos tonos puros completamente separados es el llamado ancho de banda crítico.

En el siguiente capítulo se explicará como fue creado un modelo del oído expuesto por Zhang, et.al. a partir de la información que se ha explorado en este apartado. El modelo del oído contemplará únicamente la sección del oído interno que es la que nos es de gran utilidad para la finalidad de este proyecto.

Capítulo II

Modelo del oído

2.1 Introducción

En el siguiente apartado, se hará una revisión del modelo fenomenológico para las respuestas de las fibras del nervio auditivo propuesto en el documento de Zhang, Xuedong; Heinz, Michael; Bruce, Ian y Carney, Laurel.

El objetivo principal de la investigación de Zhang, et.al., fue el diseño de un modelo único que describiera la estructura base de algunos comportamientos no lineales inherentes a las fibras del nervio auditivo, unida a la labor de mantener el modelo lo más simple que fuese posible.

Al amplificador coclear se le atribuyen diferentes comportamientos con características no lineales. Los fenómenos no lineales que condujeron el punto principal de este estudio fueron los cambios debidos a la compresión en la ganancia y el ancho de banda como función de la intensidad del estímulo, los cambios asociados en la fase de respuestas con fase fija y el enmascaramiento bitonal.

El modelo contempla el aspecto asimétrico del enmascaramiento bitonal, así como la compresión que genera el amplificador coclear junto con sus características no lineales.

Además, el modelo propuesto también reúne algunos otros modelos fenomenológicos de la membrana basal y respuestas del nervio auditivo que fueron motivo de otros estudios.

El mayor esfuerzo de esta investigación fue el de simular de forma realista la respuesta temporal y dependiente de intensidad de las fibras del nervio auditivo, sin embargo; aunque sea este sea un modelo fenomenológico, también se incluyen algunas propiedades descritas en estudios fisiológicos del sistema auditivo.

El modelo que se presenta, consiste de un filtro pasabanda con una ganancia y ancho de banda que varían en el tiempo y una trayectoria de ante-alimentación de banda ancha que permite que las particularidades del enmascaramiento bitonal sean incluidas en un modelo donde se analizan frecuencias características específicas. Esto permite la asimetría del enmascaramiento bitonal.

Para poder llevar a cabo los objetivos descritos anteriormente, el problema fue dividido en cinco bloques cada uno con una función específica. El primero es el bloque de la trayectoria de la señal, seguido del bloque de los vellos internos, el modelo de la sinapsis y

un generador de picos, además del bloque de la trayectoria de control. Éstos se describen por medio de un diagrama a bloques que se puede ver en la figura 1.1 [2].

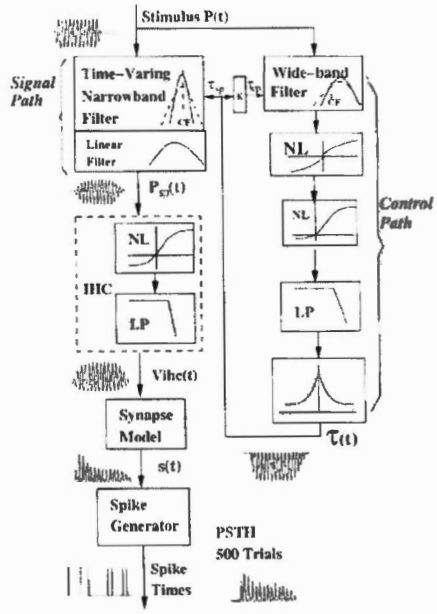


Figura 2.1.1

Los bloques principales de la sección de filtrado no lineal son la trayectoria de la señal y la trayectoria ante-alimentada de control. La trayectoria de la señal es modelada por un filtro que varía en el tiempo y un filtro lineal. En tanto que la trayectoria de control sirve para sintonizar la señal que varía en el tiempo del filtro de la trayectoria de la señal y además es responsable de los efectos de compresión y supresión, éste se modela con un filtro pasabandas que posee un ancho de banda mayor al ancho de banda del filtro que se utiliza en la trayectoria de la señal, un filtro no lineal, y un filtro pasabandas en su ultima etapa.

La salida del filtro de trayectoria de la señal, va directo hacia el bloque del modelo de los vellos internos y finalmente la sinapsis de los nervios auditivos en conjunto con los vellos internos, procesos que suceden dentro de la cóclea.

2.2 Filtro de la trayectoria de la señal

El bloque de la trayectoria de la señal, fue diseñado para representar las propiedades de la membrana basal, y es quien produce las respuestas no lineales, es decir, el enmascaramiento bitonal y la fase dependiente de la intensidad. El filtro recibe en sus entradas al estímulo y a una constante de tiempo dependiente de la señal. Su salida alimenta al modelo de los vellos internos.

Este bloque consiste de un filtro no lineal de tercer orden y de banda angosta que varia en el tiempo. En su fase final un filtro lineal de primer orden. Ambos filtros fueron basados en filtros de tipo gammatone cuya respuesta en frecuencia está dada por:

$$g(t) = u(t-\alpha)(t-\alpha)^{\gamma-1} e^{-(t-\alpha)/\tau} \cos[\omega_{CF}(t-\alpha)] \tag{2.2.1}$$

Donde α es un retraso agregado a la respuesta, τ es una constante de tiempo, ω_{CF} es la frecuencia en radianes correspondiente a la frecuencia característica del modelo de la fibra y γ es el orden del filtro.

El retrazo de la función de la frecuencia característica fue calculado como:

$$\alpha(\mathrm{CF}) = A_D e^{-x_{CF}/A_L} - 2\pi/\omega_{CF} \tag{2.2.2}$$

Donde A_D y A_L son constantes que pueden consultarse en la tabla ubicada en el anexo D y x_{CF} es la distancia en milímetros desde el ápice de la membrana basal de un gato.

La constante τ es especificada por la salida del bloque de la trayectoria de control $\tau_{sp}(t)$ para cada uno de los filtros gammatone.

La constante $\tau_{sp}(t)$ varía en un rango que está dado por τ_{narrow} y τ_{wide}. La constante de tiempo que no varía dependiendo de la señal en el filtro de primer orden es determinada por τ_{wide}.

Los valores de τ_{narrow} y τ_{wide} son establecidos por las propiedades de las fibras del nervio auditivo donde:

$$\tau_{narrow} = \frac{2Q_{10}}{2\pi \mathrm{CF}} \tag{2.2.3}$$

Y Q_{10} es el radio entre la frecuencia característica y el ancho de banda medido 10dB arriba del umbral de la fibra del nervio auditivo. Nuevamente los datos utilizados para determinarla están basados el sistema auditivo de los felinos y está dada de la siguiente manera:

$$\log_{10}(Q_{10}) = 0.4708 \log_{10}(\mathrm{CF}/1000) + 0.4664 \tag{2.2.4}$$

El parámetro τ_{wide} es elegido en base a la ganancia que se desea del filtro a niveles altos. La ganancia del amplificador coclear a una determinada frecuencia característica tiene una relación directa a la diferencia entre τ_{narrow} y τ_{wide}:

$$\tau_{wide} = \tau_{narrow} 10^{-\mathrm{gain(CF)}/60} \tag{2.2.5}$$

Donde:

$$\mathrm{gain(CF)} = \max\{5, \min[70,\ 20 + 42\log_{10}(\mathrm{CF}/1000)]\} \tag{2.2.6}$$

2.3 Trayectoria de control ante-alimentada de banda ancha

El bloque de la trayectoria de la señal necesita una señal variante en el tiempo $\tau_{sp}(t)$ que le permite replicar algunas propiedades de las respuestas dependientes de intensidad. Esta señal es proveída por este bloque. La trayectoria de control fue diseñada para reflejar, además del proceso correspondiente a la frecuencia característica local, el proceso de las frecuencias características vecinas.

Este bloque consiste de un filtro pasabanda que varía en el tiempo con un ancho de banda mayor al ancho de banda que se fijó en el bloque de la trayectoria de la señal y una función simétrica no lineal para comprimir el rango dinámico de la señal de control. Consta de una función no lineal seguida de un filtro pasabajas para controlar el rango dinámico y la dinámica de compresión. En su etapa final, se dispone de una función no lineal para ajustar la fuerza total de compresión.

El bloque de la trayectoria de control de banda ancha se implementa con un filtro de tercer orden del tipo gammatone el cual tiene su frecuencia central desviada 1.2mm con respecto a la frecuencia central que se mide a través de la membrana basal.

El ancho de banda del filtro no lineal del bloque de la trayectoria de control es fijado por $\tau_{cp}(t)$ cuyo valor es un escalamiento de $\tau_{sp}(t)$:

$$\tau_{cp}(t) = K\tau_{sp}(t) \tag{2.3.1}$$

Donde:

$$K = 0.2 + 0.8\,\frac{\tau_{wide}(\mathrm{CF})}{\tau_{narrow}(\mathrm{CF})} \tag{2.3.2}$$

El escalamiento de $\tau_{sp}(t)$ fue hecho para asegurar que el filtro de la trayectoria de control tenga una ancho de banda mayor que el del bloque de la trayectoria de la señal, y que el radio del ancho de banda se mantenga constante. La ganancia del filtro de la trayectoria de control es normalizado a 0dB con respecto a la frecuencia característica del filtro que se utiliza en el bloque de la trayectoria de la señal. Esto ocasiona que en el filtro de la trayectoria de control la dependencia de la intensidad varíe por frecuencias superiores y menores a la frecuencia característica. La asimetría que se obtiene como consecuencia, produce las diferentes propiedades del enmascaramiento en el modelo.

El filtro que se utiliza en el bloque de la trayectoria de control se describe de la siguiente manera:

$$G_{cp}(\omega) \cong \{\mathrm{gain}_{cp}(t)/[1 + jK\tau_{sp}(t)(\omega - \omega_{cp})]\}^{\gamma_{cp}} e^{-j\omega\alpha} \tag{2.3.3}$$

Donde:

$$\text{gain}_{cp}(t) = 1 + [K\tau_{sp}(\omega_{CF} - \omega_{cp})]^2 \tag{2.3.4}$$

y ω_{CF} es la frecuencia correspondiente a la frecuencia característica de la fibra. La frecuencia central del filtro de la trayectoria de control de banda ancha ω_{cp} es una constante.

Se puede mostrar de manera experimental que la respuesta coclear es lineal a intensidades de sonido bajas y se vuelve compresiva a intensidades de medias a altas. Para implementar esta compresión en la intensidad, se utilizaron dos funciones lineales de saturación dentro del bloque de la trayectoria de control. Una función logarítmica simétrica:

$$V[x(t)] = \text{sgn}[x(t)]B_{cp}\log(1 + A_{cp}|x(t)|^{C_{cp}} \tag{2.3.5}$$

Donde $x(t)$ representa la señal de salida de la trayectoria de control y A_{cp}, B_{cp} y C_{cp} son parámetros que determinan el rango dinámico de la señal antes de ser empleada la segunda función no lineal.

La segunda función es una saturación no lineal:

$$\text{out}(V) = \frac{1}{1 - \text{shift}_{cp}} \times \left\{ \frac{1}{1 + e^{-(V - x0_{cp})/s0_{cp}}(1 + e^{-(V - x1_{cp})/s1_{cp}})} - \text{shift}_{cp} \right\} \tag{2.3.6}$$

Donde $x0_{cp}$, $s0_{cp}$, $x1_{cp}$ y $s1_{cp}$ son parámetros y $\text{shift}_{cp} = \frac{1}{1 + e^{x0_{cp}/s0_{cp}}(1 + e^{x1_{cp}/s1_{cp}})}$

Seguidas de las dos funciones no lineales se utiliza un filtro de tercer orden pasabajas. La frecuencia de corte del filtro pasabajas es de 800Hz.

En la última etapa del bloque de la trayectoria de control se usa una función no lineal que convierte la salida del filtro pasabajas $V_{LP}(t)$, a la constante que varía en el tiempo dispuesta en el filtro de la trayectoria de la señal.

$$\tau_{sp}(t) = \tau_{\text{narrow}}\left[R_0 + (1 - R_0)\left(\frac{\tau_{\text{wide}}/\tau_{\text{narrow}} - R_0}{1 - R_0} \right)^{V_{LP}(t)/dc} \right] \tag{2.3.7}$$

$\tau_{sp}(t)$ alterna continuamente entre un valor máximo de τ_{narrow} y un valor mínimo τ_{LB}. El valor de R_0 es determinado por el radio de $\tau_{LB}/\tau_{\text{narrow}}$. El parámetro dc es un estimado de la componente dc de la trayectoria de control a intensidades altas.

2.4 Sinapsis de los vellos internos y los nervios auditivos.

Este bloque se encarga de modelar a los vellos internos en el sistema auditivo. La tarea de los vellos internos es la transducción de estímulos mecánicos de la membrana basal a potenciales eléctricos. El documento menciona que en estudios anteriores se ha encontrado que el potencial de los vellos internos cambia en respuesta a diferentes estímulos y que el coeficiente de sincronía de las fibras es afectado por la relación entre componentes de ca y cd producidos por la respuesta de los vellos internos.

Los vellos internos fueron modelados con una función logarítmica y compresiva:

$$V_{\text{ihc}}(t) = A_{\text{ihc}}[P_{\text{sp}}(t)]\log\left(1 + B_{\text{ihc}}\left|P_{\text{sp}}(t)\right|\right) \tag{2.4.1}$$

Donde $P_{\text{sp}}(t)$ es la salida del bloque de la trayectoria de la señal. Tanto la función $A_{\text{ihc}}[P_{\text{sp}}(t)]$ como el parámetro B_{ihc} fueron ajustados para alcanzar las respuestas apropiadas de los vellos internos.

La asimetría de la función no lineal cambia de manera suave en función del nivel de su entrada $P_{\text{sp}}(t)$, desde 1:1 hasta 3:1, de forma que a intensidades de sonidos bajas, la respuesta en dc se incrementa con una pendiente de 2dB/dB contra la pendiente de 1dB/dB que tiene su respuesta en ac.

La asimetría está dada por:

$$A_{\text{ihc}}[P_{\text{sp}}(t)] = \begin{cases} A_{\text{ihc0}} & \text{para } P_{\text{sp}}(t) > 0 \\ -\dfrac{\left|P_{\text{sp}}(t)\right|^{C_{\text{ihc}}} + D_{\text{ihc}}}{3*\left|P_{\text{sp}}(t)\right|^{C_{\text{ihc}}} + D_{\text{ihc}}} A_{\text{ihc}} & \text{para } P_{\text{sp}}(t) < 0 \end{cases} \tag{2.4.2}$$

Donde A_{ihc0} es un escalar y los valores B_{ihc}, C_{ihc} y D_{ihc} son constantes que no varían con respecto a la frecuencia característica.

El filtro pasabajas en el modelo del vello interno es de séptimo orden y su frecuencia de corte es de 3800Hz. La razón por la que este filtro tiene un orden elevado, es porque representa diversas propiedades de filtrado que posee la membrana basal y los vellos internos además de otros mecanismos pasabandas relacionados al proceso de sinapsis.

La permeabilidad inmediata $P_l(t)$ es una función rectificadora suave que forma parte del modelo del vello interno descrito como:

$$P_l(t) = p_1 \log(1 + e^{p_2 V_{\text{ihc}}(t)}) \tag{2.4.3}$$

Donde p_1 determina la permeabilidad inmediata en reposo y el cambio espontáneo del modelo de la fibra. El parámetro p_2 está dado por:

$$p_2 = \begin{cases} 1165 \text{ para } CF < 685\,\text{Hz} \\ -5430 + 1010\log(CF) \text{ para } CF > 685\,\text{Hz} \end{cases} \tag{2.4.4}$$

Y determina la pendiente que tiene la relación entre P_l y V_{ihc}, pendiente relacionada al umbral que posee el modelo de la fibra.

$s(t)$ es la salida del modelo y representa la relación de descarga que varía en el tiempo antes de que sean incluidos los efectos del periodo refractario. Se ha descrito de la siguiente manera:

$$s(t) = P_l(t)C_l(t). \tag{2.4.5}$$

2.5 Generador de descargas

Se utiliza una simulación de un proceso Poisson no homogéneo para simular los tiempos de descarga en el modelo, este proceso está modificado para incluir los efectos del periodo refractario. Se emplea $s(t)$ que es la salida del bloque de la sinapsis. El proceso se describe de la siguiente manera:

$$R(t) = s(t)[1 - H(t)] \tag{2.5.1}$$

Donde el efecto de descarga $H(t)$ se determinó como una suma de dos exponenciales.

$$H(t) = \begin{cases} c_0 e^{-(t-t_1-R_A)/s_0} + c_1 e^{-(t-t_1-R_A)/s_1} \text{ para } (t-t_1) \geq R_A \\ 1.0 \text{ para } (t-t_1) < R_A \end{cases} \tag{2.5.2}$$

Donde t_1 es el tiempo de la descarga y c_0, c_1, s_0 y s_1 son parámetros que pueden ser consultados en la tabla ubicada en el anexo D.

Este capítulo mostró el modelo del oído propuesto por Zhang, et. al. en el cual nos basaremos para diseñar una herramienta que nos permita extraer las características más significativas en una señal de voz tal y como lo hace nuestro propio oído. En el siguiente capítulo se verá la teoría de la transformada wavelet, la cual será utilizada para esta extracción de características.

Capítulo III

Transformada wavelet

3.1 Introducción

Un factor de gran importancia para el desarrollo del presente trabajo consiste en la extracción de características significativas de una señal que contenga voz esofágica. Con esto nos referimos a un cambio en la representación de las señales con las que se trabaje. Se busca resaltar aquellos atributos que puedan ser utilizados con mayor éxito al clasificar cada fonema vocalizado como la vocal que mejor lo identifique. Al mismo tiempo se pretende eliminar la información redundante y aquella que no sea significativa para el alcance de este proyecto.

En el análisis de señales, normalmente se comienza con un registro de los valores que toma una variable física durante cierto intervalo de tiempo. En ocasiones esta representación temporal resulta adecuada para extraer de ella la información que se busca. Un ejemplo de lo anterior sería la identificación de la planta en un sistema de control de lazo cerrado, con el fin de sintonizar el controlador empleando un método heurístico como el propuesto por Ziegler y Nichols.

Es común encontrarse con problemas en los cuales la representación temporal de las señales a analizar no ofrece directamente la información significativa para el objetivo que se persigue. En estos casos se recurre con frecuencia a la transformada de Fourier para cambiar la señal a una representación espectral. Dicha herramienta permite observar directamente como se distribuye la energía de la señal a lo largo de las componentes de frecuencia que la conforman. Lo anterior resulta de utilidad, por ejemplo, para el filtrado de componentes indeseables como el ruido en una señal.

3.2 Transformada de Fourier de tiempo corto

Lamentablemente el uso de la transformada de Fourier se limita al análisis de señales de carácter estacionario, es decir aquellas que presentan las mismas componentes espectrales en todo momento y siempre en la misma proporción. Algunas aplicaciones, en particular el reconocimiento de voz, requieren del tratamiento de señales que no poseen dicha característica. Estos problemas requieren de una representación que ofrezca de manera simultánea información temporal y espectral. Es decir, es necesario conocer con cierta precisión qué componentes de frecuencia existen en la señal de estudio durante un determinado intervalo de tiempo.

Una manera de abordar este tipo de problemas es utilizar la llamada transformada de Fourier de tiempo corto (STFT). Ésta consiste en dividir la señal a analizar en segmentos por medio de una ventana y computar la transformada de Fourier de cada una de las señales resultantes. Con esta herramienta es posible obtener cierta información espectral y temporal de una señal no-estacionaria. La resolución que otorgue esta representación para el tiempo y la frecuencia dependerá entonces de la ventana que se utilice. Si ésta es muy angosta se tendrá una dispersión temporal pequeña (es decir, la información obtenida de un segmento estará contenida en un fragmento de corta duración de la señal). Al mismo tiempo, la dispersión espectral obtenida con esta ventana será grande (es decir, la transformada de Fourier del segmento deberá contener la información de todas las frecuencias de la señal en un número reducido de valores).

Al modificar las propiedades de la ventana, es posible aumentar o disminuir las dispersiones espectrales y temporales de la nueva representación. Sin embargo, el producto de ambas dispersiones nunca podrá ser cero (es decir, no es posible conocer de manera simultánea que componentes de frecuencia forman una señal y en que momento ocurren). Esto se conoce como la desigualdad de Heisenberg (ver anexo A):

$$\Delta t \cdot \Delta f \geq \frac{1}{4\pi} \tag{3.2.1}$$

El valor mínimo para el producto de las dispersiones puede alcanzarse si se utiliza como ventana una función Gaussiana.

El algoritmo para obtener la STFT requiere de una ventana definida para extraer de la señal la información de sus componentes de frecuencia. Como consecuencia de lo anterior, la dispersión temporal y espectral son constantes durante todo el análisis (es decir, el espectro se divide en bandas de igual tamaño y se conoce con la misma precisión el momento en que existe una componente de baja frecuencia y una de alta frecuencia). Este tipo de representación tiene, entre otras aplicaciones, usos para el análisis de señales biológicas o provenientes de un radar o sonar.

3.3 Análisis en resoluciones múltiples

En ocasiones resultaría más apropiado dar una mayor resolución temporal a ciertas bandas de frecuencia, sacrificando la precisión espectral, y viceversa. Es decir, permitir que la dispersión temporal y espectral varíen sobre el plano tiempo-frecuencia. A este tipo de representaciones se les conoce como un análisis en resoluciones múltiples (MRA). Una descomposición de este tipo es deseable, por ejemplo, si se tiene una señal que presente componentes de baja frecuencia durante intervalos largos de tiempo acompañados por apariciones breves de componentes de alta frecuencia.

Específicamente, un tipo de MRA utilizado exitosamente en diversas aplicaciones es la llamada transformada wavelet. Para este tipo particular de análisis se busca incrementar la

resolución temporal conforme aumente la frecuencia de las componentes a identificar en la señal. Visto de otra manera, la resolución espectral aumenta conforme disminuye la frecuencia durante la descomposición. Al permitir este tipo de variación en las dispersiones temporales y espectrales es posible aislar, con una mayor efectividad que con la STFT, las componentes que aportan la mayor cantidad de energía a una señal. Lo anterior convierte a la transformada wavelet en una herramienta de gran utilidad para aplicaciones como la compresión de datos y remoción de ruido.

Además de los usos mencionados de la transformada wavelet, ésta puede emplearse para la identificación y extracción de las características más representativas de una señal. Lo anterior puede entenderse con mayor claridad si se realiza una analogía entre esta transformación y la descomposición de la señal de entrada a diferentes niveles. Se comienza con una secuencia de valores muestreados de una variable física. Estos datos pueden pensarse como el valor promedio de la señal durante un periodo de muestreo. A continuación se divide el flujo de datos en segmentos pequeños y cada uno de ellos es aproximado como el valor promedio de las muestras que lo forman. Para evitar la pérdida de información, se calcula además la diferencia entre los datos contenidos en cada segmento. Con esto se tienen ahora dos señales, la primera de ellas es una aproximación de la señal original mientras que la segunda contiene los detalles más finos que no aparecen en dicho acercamiento. Ambas señales almacenan en conjunto toda la información con la que se comenzó, pero ahora ésta se encuentra dividida. Es posible tomar ahora la señal de aproximación y repetir el algoritmo mencionado para tener así tres señales. Una de ellas sería una aproximación o promedio de los datos originales y las otras dos guardarían las variaciones más localizadas que se presentan en la variable física de estudio. El algoritmo descrito puede repetirse tantas veces como se desee o como lo permita la longitud de la señal original. Al final, se tendrá una aproximación muy burda y varias secuencias de detalles cada vez más finos que en conjunto describirán por completo a los datos iniciales.

La aplicación descrita anteriormente es de particular interés en este trabajo ya que una descomposición con las características mencionadas permite descartar la información poco útil que pueda contener una señal de voz esofágica para su reconocimiento. De esta manera el clasificador (en éste caso una red neuronal artificial) podrá ser entrenado para identificar los patrones significativos presentes sólo en las componentes de mayor energía para cada fonema vocálico.

3.4 Transformada wavelet continua

A continuación se presenta una definición de la transformada wavelet que formaliza las ideas planteadas en párrafos anteriores.

La transformada wavelet de una señal de energía finita $f(t) \in L^2\{\mathbf{R}\}$ consiste en la descomposición de ésta mediante un conjunto de funciones base $\psi_{\tau,s}(t)$ y se define de la siguiente manera:

$$\gamma(\tau,s)=\langle f,\psi_{\tau,s}\rangle=\int_{-\infty}^{\infty} f(t)\cdot\overline{\psi_{\tau,s}}(t)\,dt \tag{3.4.1}$$

En la ecuación anterior, $\langle f,\psi_{\tau,s}\rangle$ se conoce como el producto escalar de $f(t)$ con $\psi_{\tau,s}(t)$ y se calcula de acuerdo con la integral mostrada. Esta operación puede pensarse, en analogía con espacios vectoriales como $\mathbf{R}^3$, como la proyección de la señal $f(t)$ sobre $\psi_{\tau,s}(t)$. De este modo, podemos interpretar a la transformada wavelet $\gamma(\tau,s)$ como la similitud entre la señal de análisis y las funciones base empleadas para la descomposición.

El producto escalar definido mediante la integral mostrada da lugar a un espacio de funciones conocido como L^2 o, con mayor precisión, $L^2\{\mathbf{R}\}$ si se trabaja con funciones reales. Este espacio es aquel que contiene a todas las funciones cuya magnitud al cuadrado es integrable, es decir, todas las $f(t)$ para las cuales:

$$\int_{-\infty}^{\infty}|f(t)|^2\,dt<\infty \tag{3.4.2}$$

De forma similar a los espacios vectoriales comúnmente encontrados en física, $L^2\{\mathbf{R}\}$ puede caracterizarse mediante una base. Con esto nos referimos a una colección de funciones tales que cualquier señal de energía finita puede expresarse como una combinación lineal de los miembros de la base.

La transformada wavelet, como se mencionó anteriormente, es definida a través de la familia de funciones $\psi_{\tau,s}$, a las que se les denomina wavelets hijas debido a que son generadas mediante la traslación y escalamiento de una función $\psi(t)$ llamada wavelet madre. En otras palabras, por "wavelet madre" nos referimos a una especie de regla que puede utilizarse para generar funciones a partir de dos parámetros, la escala s y traslación τ. Las funciones generadas a partir de un wavelet madre poseen la característica de formar una base en $L^2\{\mathbf{R}\}$. Lo anterior es un requisito para que la descomposición preserve toda la información con la que se contaba en un inicio.

Las wavelets hijas pueden obtenerse una vez establecido un wavelet madre mediante la siguiente definición:

$$\psi_{\tau,s}(t)=\frac{1}{\sqrt{s}}\cdot\psi\left(\frac{t-\tau}{s}\right) \tag{3.4.3}$$

3.5 Transformada wavelet discreta

En general, no es práctico trabajar con valores continuos para los parámetros de escala s y traslación τ. Si se desea emplear la transformada wavelet como una herramienta para el procesamiento de señales con ayuda de una computadora, es necesario definir cierto

margen en la precisión de los cálculos. Lo anterior implica un muestreo del plano escala-traslación. Al llevar a cabo este proceso se procura conservar íntegramente la información contenida en la transformada wavelet. La solución al problema de determinar un mecanismo de muestreo apropiado puede encontrarse como una consecuencia del teorema de Littlewood-Paley (ver anexo B). De manera informal se desprende de dicho teorema que, para funciones en L^2, la escala debe ser discretizada a espacios que sigan un comportamiento geométrico. Comúnmente se elige para este fin potencias enteras de 2, es decir:

$$s_j = 2^j \tag{3.5.1}$$

La traslación es definida con base en la expresión anterior como:

$$\tau_{jk} = k \cdot s_j \tag{3.5.2}$$

donde j y k son números enteros.

Con lo anterior la ecuación (3.4.3) se transforma en la expresión:

$$\psi_k^j(t) = 2^{-\frac{j}{2}} \cdot \psi\left(2^{-j} \cdot t - k\right) \tag{3.5.3}$$

Con base en las definiciones anteriores, es posible ahora visualizar a la transformada wavelet desde el contexto del análisis funcional. Para ello se definen los subespacios W_j de $L^2\{\mathbf{R}\}$ como los subespacios generados mediante la combinación lineal de todas las traslaciones de $\psi^j(t)$ y, dado que la familia $\psi_{\tau,s}$ forma una base de $L^2\{\mathbf{R}\}$ se tiene que:

$$L^2\{\mathbf{R}\} = \bigoplus_{j \in \mathbf{Z}} W_j \tag{3.5.4}$$

Lo anterior puede interpretarse de una manera más visual al decir que la transformada wavelet realiza una división de cualquier señal de energía finita en varias proyecciones de la misma sobre espacios definidos por las wavelets hijas. Es decir, divide al conjunto de todas las funciones cuya magnitud al cuadrado es integrable en subconjuntos tales que los detalles de cierta granularidad de la señal original pueden verse como una nueva función contenida por completo en uno de los subespacios W_j.

Finalmente, se puede encontrar la siguiente expresión para la transformada wavelet inversa al recordar que los valores de la misma son el producto escalar (es decir la proyección) de la señal original sobre cada una de las wavelets hijas:

$$f(t) = \sum_{j \in \mathbf{Z}} \sum_{k \in \mathbf{Z}} \gamma_k^j \cdot \psi_k^j(t) \tag{3.5.5}$$

Mediante las definiciones presentadas anteriormente, la transformada wavelet permite extraer información tanto espectral como temporal de una señal no estacionaria. Sin embargo, al igual que otros tipos de análisis presenta limitaciones. En particular, no es capaz de separar a dos señales superpuestas cuando éstas poseen componentes espectrales en común. Es decir, los casos en los que una señal está formada por dos o más componentes que a su vez coinciden en una misma frecuencia al mismo tiempo. Una aplicación en la que se podría trabajar con señales de este tipo sería al buscar reconocer señales de audio más complejas como dos personas hablando simultáneamente o una canción compuesta por voz e instrumentos musicales. Existen transformaciones adaptivas capaces de abordar problemas como estos, sin embargo para fines de este proyecto es suficiente con la información que la transformada wavelet nos permite identificar en una señal.

En este capítulo hemos mencionado distintas maneras de representar una señal. Se expusieron posibles usos para cada una de ellas y se indicaron algunas de sus limitantes; en particular, aquellas que son relevantes para el objetivo de este proyecto. Finalmente se ofreció una breve introducción a la transformada wavelet. Dicha herramienta se utilizará en el capítulo V para extraer, de una señal de voz, características significativas para su reconocimiento mediante un clasificador. El método de clasificación se tratará en el siguiente capítulo, donde se introducirá el concepto de redes neuronales artificiales.

Capítulo IV

Redes neuronales artificiales

4.1 Introducción

Las computadoras son dispositivos que procesan información, el análisis de estos datos generalmente nos sirve para resolver diferentes tipos de problemas. Sin embargo, los ordenadores aun tienen muchas limitaciones que se deben principalmente a la forma en la que éstos buscan las soluciones. El procedimiento que sigue una computadora convencional para resolver algún problema consiste en la ejecución de un algoritmo o una serie de pasos previamente establecidos. La consecuencia que esto trae, es que la computadora no sabe como resolver problemas para los cuales aun no conocemos una solución. Las redes neuronales, por otra parte, no son programadas para realizar trabajos definidos y aprenden a través del ejemplo.

La creación de las redes neuronales estuvo inspirada en la necesidad de tener algún sistema o estructura que procesara información y realizara tareas de una manera similar a como nuestro cerebro lo hace. En la actualidad son ampliamente usadas en las áreas de reconocimiento de patrones, clasificación, aproximación de funciones y cierto tipo de predicciones.

De acuerdo con Simon Hayikin, una red neuronal es un procesador paralelo y distribuido que tiene una propensión natural para organizar conocimientos experimentales y hacerlos disponibles para su uso. Es similar al cerebro en el sentido de que su conocimiento es adquirido por medio de un proceso de aprendizaje y que éste se guarda en las conexiones interneuronales conocidas como pesos sinápticos.

Las redes neuronales tienen algunas características y propiedades que las hacen muy importantes. Por ejemplo, son sistemas adaptivos ya que a través del tiempo, los pesos sinápticos dentro de la red cambian para adaptarse a las alteraciones que vaya sufriendo la misma. Son sistemas no lineales debido a que los elementos más básicos que la conforman no son lineales. Las redes neuronales, nos brindan la oportunidad de conocer el nivel de confianza de sus resultados, lo que nos permite posteriormente rechazar información ambigua, maneja información contextual porque cada una de las neuronas dentro de la red es afectada por el conocimiento de todas las demás, y es tolerante a fallos debido a su misma naturaleza distribuida.

4.2 Modelo de una neurona artificial

Las redes neuronales están conformadas a partir de unidades elementales llamadas neuronas y las interconexiones que existen entre ellas. Comúnmente, las neuronas están agrupadas en una capa de entrada, una o varias capas intermedias u ocultas y una capa de salida.

En la figura 4.1.1, podemos ver el modelo no lineal de una neurona.

En este modelo se puede apreciar, que la neurona consta de diferentes elementos que le permiten arrojar una salida determinada a partir de las señales de entrada. Cada una de las señales de entrada tienen asociadas a ellas cantidades llamadas fuerzas de conexión o pesos sinápticos. Los pesos pueden ser del tipo excitatorio o inhibitorio, esto quiere decir que los pesos pueden tomar valores tanto positivos como negativos. Las señales de entrada alimentan a otro bloque que se encarga de sumar a todas las señales multiplicadas por su peso. La señal resultante es la entrada de un bloque que contiene una función de activación, esta función de activación tiene la misión de limitar la amplitud de la salida de la neurona para hacerla converger a un valor finito.

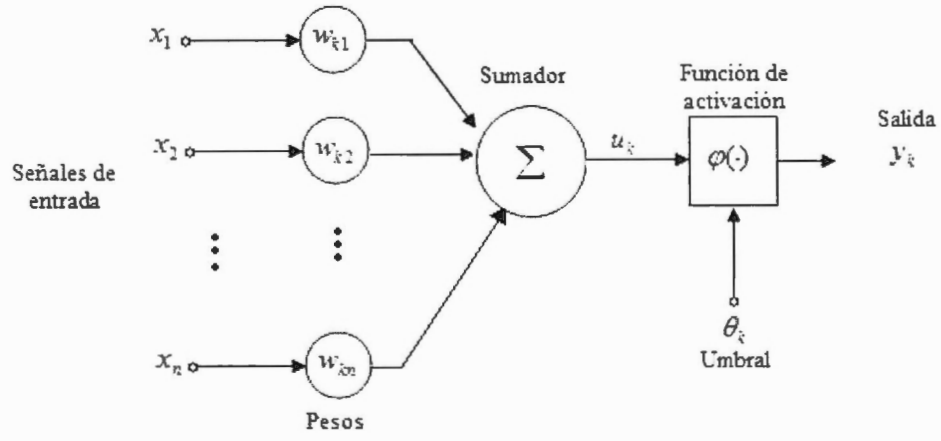


Figura 4.1.1. Modelo no lineal de una neurona.

Podemos modelar a una neurona artificial ' k ' con los siguientes términos matemáticos:

$$u_k = \sum_{j=1}^{n} w_{kj} x_j \tag{4.2.1}$$

$$y_k = \varphi(u_k - \theta_k) \tag{4.2.2}$$

Donde $x_1, x_2, ..., x_n$ son las señales de entrada, $w_{k1}, w_{k2}, ...w_{kn}$ son los pesos de la neurona k, u_k es la salida del sumador, $\varphi(\cdot)$ es la función de activación, θ_k es el umbral y y_k es la salida de la neurona.

4.3 Funciones de activación

Como ya se mencionó anteriormente, la función de activación se encuentra como una última etapa en la neurona y tiene la labor de entregar una salida acotada en términos de las entradas y sus pesos.

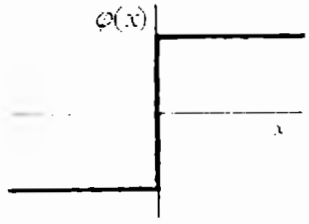
Existen diversas funciones de activación que se pueden utilizar en este bloque, algunas de ellas son la función identidad, función senoidal con intervalos de 0 a 1, la función hiperbólica con intervalos de -1 a 1, etc. En la literatura se encuentran básicamente cuatro tipos principales de funciones de activación.

Función escalón unitario o de umbral.

En esta función de activación, la salida solamente puede tomar dos valores, 0 o 1. La función de umbral se define de la siguiente manera:

$$\varphi(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4.3.1}$$

Su gráfica tiene la siguiente forma:

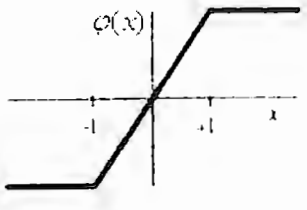


Función lineal a tramos.

Esta función, como su nombre lo indica, es una función lineal limitada por un intervalo que se define de la siguiente manera:

$$\varphi(x) = \begin{cases} 1 & x \geq \frac{1}{2} \\ x & -\frac{1}{2} > x > -\frac{1}{2} \\ 0 & x \leq -\frac{1}{2} \end{cases} \tag{4.3.2}$$

Su gráfica tiene la siguiente forma:

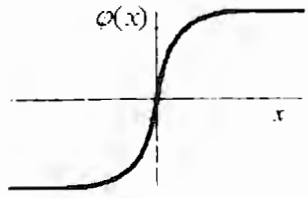


Función sigmoidal.

Esta es la función de activación más usada en el diseño de redes neuronales. La función sigmoidal tiene la propiedad de ser una función creciente que presenta cambios suaves. Se puede variar su pendiente ajustándola a las características de alguna red neuronal en particular. Está únicamente definida en el rango de 0 a 1 de la siguiente manera:

$$\varphi(x) = \frac{1}{1+e^{-ax}} \tag{4.3.3}$$

Y su gráfica tiene la siguiente forma:

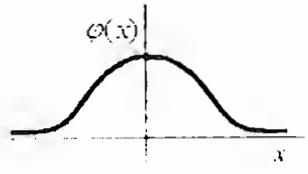


Función gaussiana.

Esta función, como en el caso de las pasadas, sólo está delimitada para el rango de 0 a 1 y presenta cambios suaves. Está definida de la siguiente manera:

$$\varphi(x) = Ae^{-Bx^2} \tag{4.3.4}$$

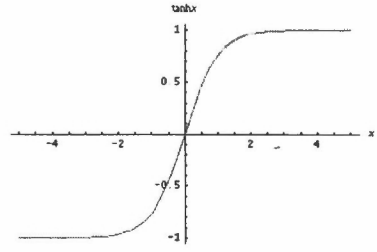
Su gráfica tiene la siguiente forma:



Función tangente hiperbólica.

Esta función es muy similar a la función sigmoidal. Como se puede apreciar en la gráfica, la función tangente hiperbólica presenta cambios suaves y es también creciente. Tiene la ventaja de ser una función diferenciable. Se define de la siguiente manera:

$$\varphi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.3.5}$$



4.4 Arquitecturas de las redes neuronales artificiales

Las redes neuronales no solamente están formadas a partir de sus neuronas, sino que también las conexiones que existen entre ellas juegan un papel muy importante para el desempeño de la red. La topología o arquitectura de una red neuronal se refiere a la estructura en la cual las neuronas que conforman la red están organizadas e interconectadas dentro de ésta.

Existen 4 clases principales dentro de las que pueden ser agrupadas las redes neuronales según su arquitectura.

Redes de una sola capa ante-alimentadas.

Esta es la arquitectura más simple de todas. Como podemos ver en la figura 4.4.1, esta arquitectura consta de una capa de entrada en la que no se realiza ningún tipo de procesamiento, y una capa de salida. Los nodos en la capa de entrada sólo transfieren información a las neuronas en la capa de la salida y esta capa es quien se encarga de realizar el procesamiento de la información que genera una respuesta. Las señales dentro de esta red sólo recorren un solo sentido, de la entrada a la salida, es por eso que se le da la clasificación de ante-alimentadas.

Redes de varias capas ante-alimentadas.

Esta arquitectura está formada por una capa de entrada, una o más capas ocultas y una capa de salida como puede verse en la figura 4.4.2. La función de la capa de entrada es la de proveer a las neuronas de la capa oculta con la información que contienen las señales de entrada. La función de las capas ocultas es la de intervenir entre la entrada externa y la salida de la red, además es en ellas donde sucede el procesamiento de la información que caracteriza a ésta. La capa de salida se encarga de generar la respuesta total de la red que pertenece al patrón de activación que fue dado por los nodos que conforman la capa de entrada. En este tipo de redes, la información sólo fluye de la entrada a la salida, es por eso que también están clasificadas como ante-alimentadas.

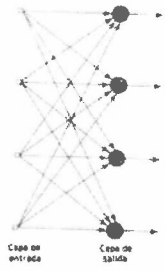


Figura 4.4.1. Red de una sola capa.

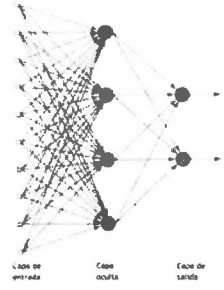


Figura 4.4.2. Red de varias capas.

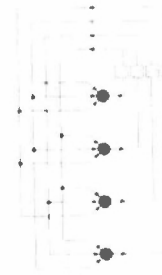


Figura 4.4.3. Red recurrente.

Redes recurrentes.

Las redes recurrentes pueden tener la estructura de las redes de una o varias capas pero con la característica especial de que constan de al menos un lazo de retroalimentación. En otras palabras, las salidas de la última capa también alimentan a la primera capa involucrando a elementos o unidades de retrazo como se puede apreciar en la figura 4.4.3. La presencia de líneas de retroalimentación tiene un gran impacto en el funcionamiento de la red.

Estructuras matriciales.

Las redes matriciales están conformadas por una capa de nodos de entrada y un arreglo de 1 o más dimensiones. Cada neurona que forma parte de la matriz tiene una o varias fuentes que vienen desde la capa de entrada. Las redes neuronales que forman estructuras matriciales, en realidad son redes del tipo de una o varias capas en donde las neuronas de salida están organizadas en filas y columnas. De igual manera, no existen lazos de retroalimentación.

4.5 El perceptrón

El perceptrón fue inventado a finales de 1950 por Frank Rosenblatt. Es la forma más simple de una red neuronal y fue creado con la intención de representar las propiedades más básicas de los sistemas inteligentes.

La característica principal del perceptrón radica en la capacidad de clasificar patrones con la estricta condición de que éstos sean linealmente separables. Esto quiere decir que estos patrones se deben encontrar en lugares opuestos en un hiperplano. Los hiperplanos son objetos de $n-1$ dimensiones dentro de espacios n-dimensionales.

El perceptrón está formado básicamente de una sola neurona con pesos y umbral ajustables.

El algoritmo de aprendizaje del perceptrón es del tipo supervisado, esto quiere decir que se necesita tanto información de las entradas como de las salidas durante la fase de entrenamiento.

La salida de un perceptrón es calculada como la suma de todas las señales de entrada multiplicadas por su peso respectivo y finalmente limitada por una función de activación.

El procedimiento de aprendizaje del perceptrón está gobernado por la regla Delta que dice que los pesos deben ser ajustados por la diferencia entre la salida deseada y la salida actual, es decir:

$$\Delta w_j = \eta(d_j - y_j)x_j \tag{4.5.1}$$

Donde el cambio en el peso es Δw_j, η es el factor de aprendizaje, d_j la salida deseada, y_j la salida actual y x_j la señal de entrada.

La mayoría de problemas de clasificación con los que nos podemos encontrar no son linealmente separables por lo que es fácil imaginar que el perceptrón tiene alcances muy limitados.

Un ejemplo sencillo de un problema que no puede ser resuelto por un perceptrón, es el problema del XOR.

Suponiendo la siguiente situación donde tenemos un perceptrón con dos señales de entrada x_1 y x_2, dos pesos asociados a estas entradas w_1 y w_2, y un umbral θ. Entonces la señal de salida sería:

$$y = \begin{cases} 1 & w_1x_1 + w_2x_2 \geq \theta \\ 0 & w_1x_1 + w_2x_2 < \theta \end{cases} \tag{4.5.2}$$

El problema consiste en elegir los valores para los pesos tal que la salida sea la de la siguiente tabla:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 4.5.1. Tabla de verdad del XOR.

La tabla 4.5.1 puede visualizarse en un esquema donde se grafiquen x_1 y x_2, entonces se obtiene un cuadrado de lados iguales donde las respuestas pertenecientes a la misma clase se encuentran en lados opuestos de este cuadrado. Como puede verse, es imposible trazar una línea recta que separe correctamente a las dos clases (0 y 1). Por lo tanto este problema no tiene solución utilizando un perceptrón.

4.6 Algoritmo de entrenamiento "Propagación hacia atrás"

Existen diferentes algoritmos de entrenamiento para las redes neuronales que dependen de la arquitectura que esta tenga. Uno de los algoritmos que se basa en el aprendizaje supervisado más usados y más fáciles de comprender para la arquitectura de varias capas es el algoritmo de propagación hacia atrás.

La labor de los algoritmos de entrenamiento es encontrar los pesos y los umbrales adecuados para cada neurona que minimicen el error cometido por la red. Es decir, se deben ajustar los pesos y umbrales de las neuronas de tal manera que el error entre la salida deseada y la salida actual, sea el mínimo.

El método de propagación hacia atrás es una generalización del algoritmo de mínimos cuadrados. Se define una función de error y se busca minimizarla por el descenso del gradiente. Los pesos se ajustan con la ayuda de la regla generalizada Delta calculando el error para la entrada actual y propagando hacia atrás este error capa por capa. Una condición de funcionamiento para esta técnica de entrenamiento es que su función de activación sea diferenciable.

El algoritmo de propagación hacia atrás propone una serie de pasos para lograr ese objetivo.

Primero se determina que tan rápido cambia el error conforme la salida de una neurona en la capa de salida cambia. Esto se logra inicialmente calculando la salida de la neurona j:

$$u_j = \sum_{i=1}^{n} w_{ji} x_i \tag{4.6.1}$$

$$y_j = \varphi(u_j - \theta_j) \tag{4.6.2}$$

Donde $x_1, x_2, ..., x_n$ son las señales de entrada, $w_{j1}, w_{j2}, ...w_{jn}$ son los pesos de la neurona j, u_j es la salida del sumador, $\varphi(\cdot)$ es la función de activación que para este ejemplo será la función sigmoidal, θ_j es el umbral y y_j es la salida de la neurona.

Luego se obtiene el error de todas las respuestas de los nodos de la capa de salida:

$$E = \frac{1}{2}\sum_{j \in C}\left(y_j - d_j\right)^2 \tag{4.6.3}$$

Donde y_j es la salida de la neurona j, d_j es la salida deseada de la neurona j y C indica la inclusión de todas las neuronas en esa capa.

Entonces ya se puede determinar que tan rápido cambia el error con respecto a la salida de una neurona en la capa de salida:

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j \tag{4.6.4}$$

Luego se calcula la tasa de cambio del error con respecto a la entrada de un nodo en la capa de salida:

$$EI_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \times \frac{\partial y_j}{\partial x_j} = EA_j y_j (1 - y_j) \tag{4.6.5}$$

Después, se determina como va cambiando el error con respecto al cambio del peso entre el enlace de esa neurona y la anterior.

$$EWij = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial W_{ij}} = EI_j y_i \tag{4.6.6}$$

Para el caso de las neuronas de las capas intermedias, se calcula cómo cambia el error con respecto a la salida de una neurona en una capa previa:

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_{j \in C} \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial y_i} = \sum_{j \subset C} EI_j W_{ij} \tag{4.6.7}$$

Este paso es lo que le da la característica principal al nombre de propagación hacia atrás. Debido a que el cambio en la respuesta de todas las capas previas a la capa de salida afecta la respuesta de ésta, se deben de sumar todos estos efectos por separado.

Una vez que se hayan calculado todos los EA para cada nodo, entonces se procede a calcular los EW para cada uno de estos.

Finalmente, la corrección del peso de cada uno de los nodos se calcula de la siguiente manera:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial W_{ij}} = -\eta EW_{ij} \tag{4.6.8}$$

Donde η es el factor de aprendizaje.

Y la corrección del umbral para cada uno de los nodos sería la siguiente:

$$\Delta \theta_j = -\eta \frac{\partial E}{\partial \theta_j} = \eta EA_j \theta_j (1 - \theta_j) \tag{4.6.9}$$

En este capítulo se revisó la teoría general de las redes neuronales artificiales y el algoritmo de entrenamiento "propagación hacia atrás". Éste finaliza el marco teórico del presente documento. En el siguiente apartado utilizaremos los elementos que fueron descritos anteriormente para realizar el diseño de nuestro sistema propuesto.

Capítulo V

Diseño del sistema

5.1 Descripción de un algoritmo para el reconocimiento de vocales en señales de voz

La finalidad del presente trabajo, como se ha mencionado anteriormente, consiste en el diseño y la implantación de un algoritmo que permita la identificación de los fonemas vocálicos presentes en una señal de voz esofágica. Este capítulo pretende, con base en los conceptos tratados en secciones anteriores, definir un posible sistema capaz de alcanzar el objetivo mencionado.

Los orígenes del lenguaje oral traen como consecuencia que la voz humana se encuentre formada por una gran cantidad de características fortuitas y sea fundamentalmente una convención realizada por la sociedad que la utiliza. Esto puede observarse fácilmente en la diversidad de idiomas desarrollados a causa de las barreras geográficas y culturales que se hicieron presentes a lo largo de la historia humana.

Debido a lo anterior, el problema de reconocimiento de voz pareciera no tener una solución analítica. Sin embargo, la lengua se desarrolló para ser producida y percibida por los seres humanos, esto le otorga algunas características que se han aprovechado para el desarrollo de algoritmos heurísticos capaces de solucionar la problemática mencionada de forma aproximada.

En este trabajo se empleará parte de la información conocida sobre la fisiología del oído humano. Asimismo, se aprovechan en menor medida algunas peculiaridades de la producción del habla. Todo esto con la finalidad de modelar las señales de voz y extraer de las mismas parámetros significativos para el reconocimiento de los fonemas que contienen. Se desea que las características analizadas resalten aquellos atributos más significativos para la identificación de las vocales y sean robustas ante variaciones en las condiciones de medición como cambios en el locutor o en los instrumentos utilizados.

A fin de alcanzar el objetivo central de este proyecto se ha propuesto la división del problema en los puntos mostrados en el diagrama a bloques de la figura 5.1.1.

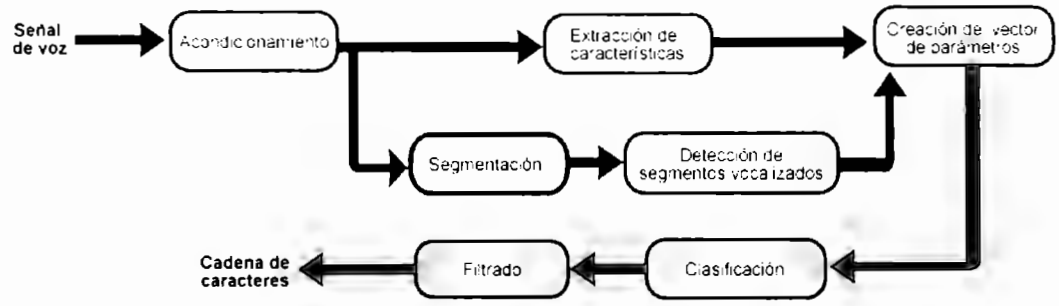


Figura 5.1.1. Diagrama a bloques de un sistema de reconocimiento de voz

A continuación se examina a detalle cada una de las etapas presentadas y el modo en que interactúan en el sistema final.

5.2 Acondicionamiento

La señal de audio ingresa al sistema en esta etapa. En ella se espera recibir como entrada el muestreo de las variaciones en la presión atmosférica debidas a la voz cuyas componentes vocálicas se desean reconocer. La información anterior es procesada con la finalidad de modificar sus características espectrales y que las mismas sean óptimas para las siguientes etapas.

El acondicionamiento de la señal consiste en el filtrado de la misma para reducir el ruido; la normalización de su amplitud y la amplificación de sus componentes de mayor importancia para el reconocimiento de voz.

Durante la etapa de implantación de este algoritmo, se decidió realizar el filtrado inicial por medio de un filtro digital de respuesta al impulso finita (FIR) diseñado con el método de mínimos cuadrados (i.e. buscando minimizar el error cuadrático medio entre una respuesta en frecuencia deseada y la real). En particular, para las pruebas del sistema se utilizó un filtro pasa-altas de cien coeficientes con una frecuencia de corte de 100 Hertz. Con ello se busca eliminar el ruido de 60 Hertz presente en las grabaciones utilizadas.

Además de eliminar las componentes indeseables de la señal de audio, se desea resaltar sus constituyentes de mayor relevancia para el reconocimiento de voz. Esto se logra mediante otro filtro, esta vez pasa-altas de un coeficiente, llamado de pre-énfasis. Un filtro de este tipo presenta una pendiente de 20 deciBeles por década, de modo que las componentes de alta frecuencia son amplificadas. Lo anterior cumple con dos tareas significativas: equilibra una atenuación de similar magnitud presente en las secciones vocalizadas de la voz e imita la sensibilidad adicional del oído humano a sonidos de frecuencias altas [1]. El filtro de pre-énfasis utilizado durante la programación de este algoritmo fue seleccionado de acuerdo con los valores típicos encontrados en la literatura sobre el reconocimiento de voz, y presenta la siguiente respuesta al impulso h:

$$h = \begin{bmatrix} 1 & -0.4 \end{bmatrix} \tag{5.2.1}$$

Finalmente, la etapa de pre-procesamiento de la señal concluye con la normalización de la misma. Con ello se garantiza que señales formadas por componentes semejantes posean, a la salida del bloque de acondicionamiento, la misma energía. Lo anterior será importante en secciones posteriores donde la potencia promedio de la señal se utiliza para tomar decisiones acerca de su contenido fonético.

La etapa de acondicionamiento entrega al resto del sistema una representación temporal de la señal a analizar, adecuada para el reconocimiento de los fonemas vocálicos que pudiera contener. El proceso descrito fue programado para Matlab dentro de la función isVoiced disponible en el anexo C de este documento.

5.3 Segmentación

El alcance de este trabajo requiere de un algoritmo capaz de identificar únicamente los fonemas vocálicos en señales de voz esofágica. Una característica en común para todos los alófonos de las vocales es que presentan un comportamiento vocalizado (i.e. las cuerdas vocales vibran durante su producción). Teniendo presente lo anterior, se propone la división de la señal de entrada en tramas que, posteriormente, serán clasificadas como vocalizadas o no vocalizadas. Esto permitirá que etapas posteriores no inviertan tiempo de cómputo en aquellas secciones donde no podrían existir vocales.

Ya que la identificación del contenido vocálico se realizará trama a trama; la duración de éstas debe ser tal que, en general, no contengan más de un fonema. Asimismo, la información presente en ellas debe ser suficiente para llevar a cabo la identificación de su contenido. A fin de evitar errores debidos a discontinuidades generadas por la segmentación, cada secuencia es multiplicada por una ventana de cambios suaves. La duración de la ventana se elige mayor al tiempo de trama para prevenir la pérdida de información en las transiciones de una secuencia a la siguiente. Con lo anterior, existirá una breve repetición de los datos entre segmentos contiguos.

Para la implantación en Matlab de este bloque, se consideró una segmentación en tramas de 15 milisegundos empleando una ventana con una duración de 30 milisegundos [3]. Ambos valores son consistentes con los utilizados frecuentemente en sistemas para el análisis y reconocimiento de voz. Para el suavizado de las transiciones de un segmento a otro se empleó una ventana de Hamming, definida de acuerdo a la siguiente ecuación:

$$W[k+1] = 0.54 - 0.46\cos\left(2\pi\frac{k}{n-1}\right), k = 0, 1, \ldots, n-1 \qquad (5.3.1)$$

donde n representa al número de muestras contenidas en la ventana y puede calcularse como el producto de la duración de la misma y la frecuencia de muestreo utilizada en la señal de audio.

Siguiendo los pasos señalados, esta etapa opera sobre la señal previamente acondicionada y entrega a su salida un conjunto de secuencias de menor longitud; cada una correspondiente a la información de entrada en intervalos de tiempo distintos. El código de Matlab encargado de llevar a cabo esta tarea forma parte del archivo isVoiced.m (ver anexo C).

5.4 Detección de segmentos vocalizados

Como se mencionó en la sección anterior, para el alcance de este trabajo sólo es necesaria la clasificación de los segmentos con contenido vocálico de las señales de voz. Esta etapa pretende identificar cada una de las secuencias provenientes del bloque de segmentación como vocalizadas o no vocalizadas. Con la información anterior, será posible descartar los datos que no podrían representar a una vocal antes de que estos ingresen al clasificador.

En el capítulo I del presente documento, se mencionó que un fragmento de voz se considera vocalizado si durante su generación el flujo de aire es alterado por las vibraciones de las cuerdas vocales del locutor. Dicho proceso aporta a la señal resultante características que permiten diferenciarla de las secciones no vocalizadas de la voz. En particular, las componentes vocalizadas presentan un comportamiento periódico y transportan una mayor cantidad de energía que los sonidos no vocalizados [3]. La etapa de detección de segmentos vocalizados propuesta para este trabajo aprovecha las dos características mencionadas.

Como primer criterio, se considera la potencia promedio de cada segmento; ésta puede calcularse mediante la siguiente fórmula:

$$P_{prom} = \frac{1}{n} \cdot \sum_{k=1}^{n} |x[k]|^2 \qquad (5.4.1)$$

donde x representa la señal contenida en el segmento a analizar y n el número de muestras que lo forman.

La energía transportada por la señal de voz depende de las condiciones durante la adquisición de los datos. Factores como el cambio de locutor, el tono al hablar o la presencia de ruido pueden modificarla. Esto hace imposible definir un intervalo fijo para la potencia promedio que deba tener un segmento a fin de ser considerado como vocalizado. Debido a esta problemática, el algoritmo para la detección de secuencias vocalizadas compara la potencia promedio del segmento con la que posee la señal en la cercanía del mismo. Específicamente se considera un intervalo de 200 milisegundos alrededor de la trama a analizar. Dicho intervalo se obtuvo de manera empírica como la duración media de una sílaba al hablar a una velocidad normal. Al tomar en cuenta que la señal podría tener intervalos de silencio o regiones largas con un comportamiento vocalizado, por ejemplo en palabras que presenten diptongos, se fijó un límite superior e inferior para el valor con el cual se comparará la potencia promedio de cada segmento.

Siguiendo con el mecanismo para el etiquetado de los segmentos propuesto por Greenwood, et. al. [3], se fijan dos umbrales. Si el cociente de la potencia promedio del segmento entre la que presenta la señal alrededor del mismo (con los límites descritos en el párrafo anterior) es menor al umbral inferior, la trama es considerada no vocalizada. Si dicho cociente es mayor al umbral superior, entonces los datos se clasifican como vocalizados. Para los casos en que la razón de potencias promedio se ubique entre ambos umbrales, el criterio de energía no basta para determinar la naturaleza de la señal.

Cuando lo anterior ocurre, se aprovecha a la periodicidad de los segmentos vocalizados. Específicamente, el algoritmo propuesto realiza una clasificación basada en el número de cruces por cero de la señal (i.e. la cantidad de veces que ésta cambia de signo). Nuevamente se establecen dos umbrales. Estos son similares a los propuestos por Mark Greenwood y Andrew Kinghorn [3]. Ellos consideran un intervalo de 1200 a 5000 cruces por cero por segundo. Si la cantidad de cruces por cero encontrados para el segmento se sitúa dentro de dicho intervalo, entonces éste se considera vocalizado. En caso contrario es marcado como no vocalizado. Los umbrales utilizados para este criterio durante la implantación del algoritmo en Matlab no son fijos como proponen Greenwood y Kinghorn. Esto se debe a que las pruebas del algoritmo demostraron que el intervalo presentado no es el óptimo para cualquier locutor. Cambios en el género, la edad y si el hablante es o no un paciente laringectomizado requieren de modificaciones a dichos umbrales.

La salida de este bloque consiste en una cadena de valores lógicos (uno por cada trama) que indican si el contenido de la señal es o no predominantemente vocalizado. El mismo fue implantado en Matlab como la función analyze dentro del archivo isVoiced.m disponible al final de este documento (anexo C).

5.5 Extracción de características

La extracción de características de una señal de voz es probablemente la etapa de mayor importancia para el desarrollo de este trabajo. El desempeño de este bloque afecta significativamente la calidad de los resultados obtenidos por el sistema en conjunto. Como hemos mencionado con anterioridad, el modelado de la señal de audio se realiza con base en información conocida sobre la fisiología del oído humano.

5.5.1 Construcción de un wavelet madre a partir de un modelo del oído interno

Como se ha visto en capítulos anteriores de este documento, en el sistema auditivo humano, es la cóclea el órgano encargado de transformar una vibración mecánica en un impulso nervioso que posteriormente es interpretado como sonido por el cerebro. Dentro de la cóclea, la membrana basal en conjunto con los bellos exteriores realiza una descomposición espectral de las señales mecánicas. Debido a la heterogeneidad en la estructura física de estos cuerpos, la descomposición no posee la misma resolución para todas las frecuencias. En concreto, entre mayor sea la frecuencia característica para una sección de la membrana basal, mayor será el rango de frecuencias para las cuales ésta

responderá con un impulso en el nervio auditivo. Este fenómeno nos permite abstraer lo que ocurre dentro de nuestro oído como el análisis de resoluciones múltiples de una señal. Específicamente, como parte del desarrollo de este proyecto, se busca emplear el método de la transformada wavelet para conseguir una extracción de características de una señal de audio similar a la realizada por el oído interno humano.

En esta sección se propone una función $\psi(t) \in L^2(\mathbf{R})$ a partir de la cual sea posible generar una base incondicional para dicho espacio funcional. En el contexto de la transformada wavelet, y como se discutió en el capitulo IV de éste documento, a las funciones de este tipo se les denomina "Wavelet madre".

Siguiendo la premisa de que el habla se ha formado durante la historia con la finalidad de ser producido y escuchado por seres humanos, es deseable que el wavelet madre propuesto posea algunas de las características más representativas del receptor natural del habla (i.e. nuestro oído). Con lo anterior en mente, se busca definir $\psi(t)$ tomando como base algunos aspectos del modelo presentado por Zhang, et. al. [2]. En las siguientes secciones de este documento se plantea un algoritmo mediante el cual se combinan otras peculiaridades de dicho modelo con los resultados de este capítulo, así como con la interpolación de la escala Bark propuesta por Schroeder, et. al. [4] para obtener un vector de parámetros que represente el contenido de una señal de voz.

Zhang, et. al. [2] proponen que la dinámica de una sección de la membrana basal que presenta una frecuencia característica f_c, es similar a la de un filtro *gamma-tone* sintonizado a esa misma frecuencia. El nombre de este tipo de filtros proviene de su representación temporal, la cual consiste en el producto de una distribución gama por un tono.

La distribución de probabilidad gama indica la probabilidad de que, desde un tiempo inicial, hayan ocurrido un cierto número de eventos que presentan una distribución Poisson; ésta puede escribirse como [5]:

$$P[\alpha,\theta](x) = \frac{x^{\alpha-1} \cdot e^{\frac{-x}{\theta}}}{\Gamma(\alpha) \cdot \theta^{\alpha}} \quad x > 0 \tag{5.5.1.1}$$

donde α y θ son parámetros denominados "forma" y "escala" respectivamente, y $\Gamma(x)$ es la función gama definida de la siguiente manera [6]:

$$\Gamma(x) = \int_0^{\infty} t^{x-1} \cdot e^{-t} dt \tag{5.5.1.2}$$

En un filtro *gamma-tone*, la forma de la distribución gama se asocia generalmente con el orden del filtro. Lo anterior implica que $\alpha \in \mathbf{N}$ por lo que la ecuación (5.5.1.1) puede reescribirse como:

$$P[\alpha,\theta](x)=\frac{x^{\alpha-1}\cdot e^{\frac{-x}{\theta}}}{(\alpha-1)!\cdot\theta^{\alpha}} \quad x>0 \tag{5.5.1.3}$$

La expresión anterior se conoce como la distribución de Erlang. En este punto introduciremos la restricción $\alpha>1$ con la finalidad de evitar caer en el caso particular para el cual la distribución anterior se reduce a una distribución exponencial.

Siguiendo con el modelo inicial, ahora multiplicamos la distribución anterior por un tono. Éste es elegido de tal manera que su frecuencia sea igual a la tasa con la que ocurren los eventos Poisson que dan origen a la distribución gama (i.e. el promedio de veces que éstos acontecen por unidad de tiempo).

$$\psi_{\theta}^{\alpha}(t)=\frac{1}{(\alpha-1)!\cdot\theta^{\alpha}}\cdot t^{\alpha-1}\cdot e^{\frac{-t}{\theta}}\cdot\cos\left(2\pi\cdot\frac{1}{\theta}\cdot t\right) \quad t>0 \tag{5.5.1.4}$$

Ahora tomamos un caso particular del resultado anterior al permitir que la escala $\theta=1$

$$\psi^{\alpha}(t)=\frac{1}{(\alpha-1)!}\cdot t^{\alpha-1}\cdot e^{-t}\cdot\cos(2\pi\cdot t) \quad t>0 \tag{5.5.1.5}$$

Es posible afirmar que $\forall\alpha\in\mathbf{N}\ \psi^{\alpha}(t)\in L^2(\mathbf{R})$ tomando la norma de $\psi^{\alpha}(t)$

$$\left\|\psi^{\alpha}(t)\right\|_{L^2(\mathbf{R})}^2=\int_{-\infty}^{\infty}\left|\psi^{\alpha}(t)\right|^2dt=\int_{0}^{\infty}\left|\frac{1}{(\alpha-1)!}\cdot t^{\alpha-1}\cdot e^{-t}\cdot\cos(2\pi\cdot t)\right|^2dt \tag{5.5.1.6}$$

Dado que α es un número entero (y por tanto finito), la componente $t^{\alpha-1}$ es de orden exponencial, es decir:

$$\exists(T>0,c>0,M>0)\left\{\forall t>T\left(\left|t^{\alpha-1}\right|\le M\cdot e^{ct}\right)\right\} \tag{5.5.1.7}$$

de modo que la integral converge y la norma de $\psi^{\alpha}(t)$ en $L^2(\mathbf{R})$ existe.

Tomando una función arbitraria $f(t)\in L^2(\mathbf{R})$ podemos generar un conjunto de coeficientes $\gamma^{\alpha}(\tau,s)$ definidos mediante el producto escalar entre $f(t)$ y las traslaciones en $\tau\in\mathbf{R}$ y escalamientos en $s\in\mathbf{R}^{+}$ de $\psi^{\alpha}(t)$

$$\gamma^{\alpha}(\tau,s)=\left\langle f,\psi_{\tau,s}^{\alpha}\right\rangle \tag{5.5.1.8}$$

donde

$$\psi_{\tau,s}^{\alpha}(t)=\frac{1}{\sqrt{s}}\cdot\psi^{\alpha}\left(\frac{t-\tau}{s}\right) \tag{5.5.1.9}$$

El coeficiente $s^{-1/2}$ es introducido con la finalidad de mantener la norma de $\psi^{\alpha}_{\tau,s}(t)$ invariante a los cambios de escala.

Es posible reconstruir la ecuación (5.5.1.4) a partir de la definición anterior al permitir que $s=\theta$ y $\tau=0$, por lo que resulta intuitivo asociar el factor de escalamiento s con la escala θ de la distribución gama que da origen al filtro *gama-tone* presentado en (5.5.1.4).

Retomando la familia de coeficientes planteada en (5.5.1.8) tenemos:

$$\gamma^{\alpha}(\tau,s)=\int_{-\infty}^{\infty} f(t)\cdot\overline{\psi^{\alpha}_{\tau,s}}(t)\,dt \tag{5.5.1.10}$$

Y como consecuencia del teorema de Plancherel podemos escribir la igualdad anterior como:

$$\gamma^{\alpha}(\tau,s)=\frac{1}{2\pi}\cdot\int_{-\infty}^{\infty} F(\omega)\cdot\overline{\Psi^{\alpha}_{\tau,s}}(\omega)\,d\omega=\frac{1}{2\pi}\cdot\int_{-\infty}^{\infty} F(\omega)\cdot e^{i\cdot\tau\cdot\omega}\overline{\Psi^{\alpha}}(s\omega)\,d\omega \tag{5.5.1.11}$$

donde $F(\omega)$, $\Psi^{\alpha}_{\tau,s}(\omega)$ y $\Psi^{\alpha}(\omega)$ representan la transformada de Fourier de $f(t)$, $\psi^{\alpha}_{\tau,s}(t)$ y $\psi^{\alpha}(t)$ respectivamente.

Reescribiendo el resultado anterior se obtiene la siguiente igualdad:

$$\gamma^{\alpha}(\tau,s)=\frac{1}{2\pi}\cdot\int_{-\infty}^{\infty}\left[F(\omega)\cdot\overline{\Psi^{\alpha}}(s\omega)\right]\cdot e^{i\cdot\tau\cdot\omega}\,d\omega \tag{5.5.1.12}$$

Definiendo las funciones $g(\tau)$ y $G(\omega)$ como:

$$\begin{aligned} &g(\tau):\tau\rightarrow\gamma^{\alpha}(\tau,s)\\ &G(\omega):\omega\rightarrow F(\omega)\cdot\overline{\Psi^{\alpha}}(s\omega)\end{aligned} \tag{5.5.1.13}$$

se observa que la ecuación (5.5.1.12) es equivalente a:

$$g(\tau)=\frac{1}{2\pi}\cdot\int_{-\infty}^{\infty} G(\omega)\cdot e^{i\tau\omega}\,d\omega=\mathbf{F}^{-1}\{G(\omega)\} \tag{5.5.1.14}$$

es decir que, $G(\omega)$ es la transformada de Fourier de $g(\tau)$. Lo anterior permite emplear el teorema de Parseval, e integrar ambos lados de la igualdad con respecto a $\frac{ds}{s}$ para llegar a la expresión:

$$\int_{0}^{\infty}\int_{-\infty}^{\infty}\left|\gamma^{\alpha}(\tau,s)\right|^{2}\frac{d\tau\cdot ds}{s}=\frac{1}{2\pi}\cdot\int_{0}^{\infty}\int_{-\infty}^{\infty}\left|F(\omega)\right|^{2}\cdot\left|\Psi^{\alpha}(s\omega)\right|^{2}\frac{d\omega\cdot ds}{s} \tag{5.5.1.15}$$

Si se define a $H \neq 0$ como una constante relacionada únicamente a ψ^{α} y se calcula la norma de $f(t)$ se obtiene, mediante el teorema de Parseval,

$$\|f(t)\|^2_{L^2(\mathbf{R})} = \int_{-\infty}^{\infty} |f(t)|^2 \, dt = \frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} |F(\omega)|^2 \, d\omega = H(\psi^{\alpha}) \cdot \left[\frac{1}{2\pi} \cdot \int_{-\infty}^{\infty} |F(\omega)|^2 \cdot \frac{1}{H(\psi^{\alpha})} d\omega \right] \tag{5.5.1.16}$$

Y al igualar la expresión entre corchetes con la ecuación (5.5.1.15) se obtiene que:

$$H(\psi^{\alpha}) = \frac{1}{\int_0^{\infty} |\Psi^{\alpha}(s\omega)|^2 \frac{ds}{s}} \tag{5.5.1.17}$$

De modo que, si se cumple la condición:

$$\int_0^{\infty} |\Psi^{\alpha}(s\omega)|^2 \frac{ds}{s} < \infty \tag{5.5.1.18}$$

entonces la familia de coeficientes $\gamma^{\alpha}(\tau, s)$ generarían la norma de $f(t)$ de la siguiente manera:

$$\|f(t)\|^2_{L^2(\mathbf{R})} = \int_{-\infty}^{\infty} |f(t)|^2 \, dt = H(\psi^{\alpha}) \cdot \int_0^{\infty} \int_{-\infty}^{\infty} |\gamma^{\alpha}(\tau, s)|^2 \frac{d\tau \cdot ds}{s} \tag{5.5.1.19}$$

En particular, la función descrita en la ecuación (5.5.1.5) cumple con la condición. Esto se puede verificar al construir su transformada de Fourier como [2]:

$$\Psi^{\alpha}(\omega) = \frac{1}{2} \cdot (\alpha - 1)! \cdot \left[\frac{1}{[1 + i \cdot (\omega - 2\pi)]^{\alpha}} + \frac{1}{[1 + i \cdot (\omega + 2\pi)]^{\alpha}} \right] \tag{5.5.1.20}$$

Y sustituyendo este resultado en la integral descrita en (5.5.1.18):

$$\int_0^{\infty} |\Psi^{\alpha}(s\omega)|^2 \frac{ds}{s} = \int_0^{\infty} \left| \frac{1}{2} \cdot (\alpha - 1)! \cdot \left[\frac{1}{[1 + i \cdot (s\omega - 2\pi)]^{\alpha}} + \frac{1}{[1 + i \cdot (s\omega + 2\pi)]^{\alpha}} \right] \right|^2 \frac{ds}{s} \tag{5.5.1.21}$$

La expresión anterior toma un valor real para todos los valores positivos de ω. Esto implica que la familia de funciones $\psi^{\alpha}_{\tau,s}(t)$ que origina a dichos coeficientes forma una base incondicional de $L^2(\mathbf{R})$ y por tanto $\psi^{\alpha}(t)$ puede considerarse un wavelet madre para cualquier valor positivo de su parámetro α.

5.5.2 Transformada wavelet continua

Debido a sus similitudes con la dinámica del oído interno, se utilizará la transformada wavelet como parte de la extracción de características. Retomando la información

presentada en el capítulo IV, la transformada wavelet de una señal de energía finita $x(t) \in L^2\{\mathbf{R}\}$ se define como:

$$\gamma(\tau,s) = \langle x, \psi_{\tau,s} \rangle = \int_{-\infty}^{\infty} x(t) \cdot \overline{\psi_{\tau,s}}(t)\, dt \qquad (5.5.2.1)$$

donde $\psi_{\tau,s}$ se obtiene a partir de un wavelet madre $\psi(t)$ empleando la fórmula:

$$\psi_{\tau,s}(t) = \frac{1}{\sqrt{s}} \cdot \psi\left(\frac{t-\tau}{s}\right) \qquad (5.5.2.2)$$

Para el presente proyecto, la función wavelet madre a utilizar será la mostrada en la ecuación (5.5.1.5).

La transformada wavelet puede relacionarse con la correlación cruzada entre la señal y el wavelet escalado. La correlación de dos señales se define de la siguiente manera:

$$C[f,g](L) = \int_{-\infty}^{\infty} f(t) \cdot \overline{g}(t-L)\, dt \qquad (5.5.2.2)$$

Y en tiempo discreto la definición anterior se convierte en:

$$C\{f,g\}[k] = \sum_{i=0}^{n-k-1} f[i] \cdot \overline{g}[i-k] \qquad (5.5.2.3)$$

Si definimos a $\psi_s(t)$ como el wavelet madre escalado por s, tenemos la siguiente relación:

$$\gamma(\tau,s) = \int_{-\infty}^{\infty} x(t) \cdot \overline{\psi_{\tau,s}}(t)\, dt = \int_{-\infty}^{\infty} x(t) \cdot \overline{\psi_s}(t-\tau)\, dt = C[x,\psi_s](\tau) \qquad (5.5.2.4)$$

Y con ello podemos calcular la transformada wavelet continua para señales en tiempo discreto como:

$$\gamma(k,s) = \sum_{i=0}^{n-k-1} x[i]\overline{\psi_s^*}[i-k] \qquad (5.5.2.5)$$

donde la traslación k solo puede tomar valores enteros y ψ_s^* se refiere al muestreo del wavelet madre luego de ser escalado por s.

5.5.3 Muestreo del plano escala-traslación

El resultado de la sección anterior nos muestra que, al trabajar con señales en tiempo discreto, estamos obligados a muestrear el eje de la traslación. Esto quiere decir que el parámetro τ en la ecuación 5.5.2.2 no puede tomar cualquier valor real. Este fenómeno es

común al utilizar sistemas digitales para el procesamiento de una señal. A continuación presentaremos una manera de muestrear el eje de la escala. Esto se realiza con dos objetivos en mente: obtener una discretización de la transformada wavelet que pueda ser calculada en una computadora e incorporar el principio psicoacústico de las bandas críticas en la descomposición planteada.

En el capítulo IV se mencionó que el algoritmo para computar la transformada wavelet discreta realiza un muestreo diádico del plano escala-translación. Esta decisión se justifica ya que permite el uso de filtros digitales y reducción del número de muestras de la señal para obtener la descomposición de una señal en tiempo discreto. Con lo anterior es posible diseñar algoritmos cuya ejecución requiera de muy poco tiempo de cómputo.

Sin embargo, la extracción de características propuesta en este trabajo busca la descomposición de la señal en bandas que asemejen la respuesta en frecuencia de la cóclea. Lo anterior obliga a realizar un muestreo del plano escala-traslación distinto al usual. Específicamente, este proyecto pretende efectuar una descomposición de la señal que muestre de manera directa sus componentes para cada una de las bandas críticas definidas por la escala Bark.

La escala Bark es una mapeo logarítmico de la frecuencia. Ésta fue creada de forma tal que la resolución espectral del oído humano sea de un Bark para cualquier frecuencia característica. Al definirse con base en un parámetro biológico, no es posible hallar una expresión exacta que relacione la frecuencia en Hertz con la misma en Barks. Sin embargo diversos autores han propuesto interpolaciones basadas en mediciones experimentales. Para fines de este proyecto se empleará la interpolación propuesta por Schroeder et. al. [4]

$$Z = 7\ln\left(\frac{f}{650} + \sqrt{\left(\frac{f}{650}\right)^2 + 1}\right) \qquad (5.5.3.1)$$

La escala presentada determina a veinticuatro frecuencias críticas. Estas se definen mediante incrementos de 1 Bark a lo largo de toda la banda audible.

Con base en la interpolación mostrada, podemos definir un muestreo de la escala apropiado para la transformada wavelet continua de forma tal que la descomposición de la señal de voz entregue de manera directa la información referente a cada una de las bandas críticas del oído humano:

$$s_j = \frac{1}{325} \cdot \frac{e^{\frac{j}{7}}}{e^{\frac{2j}{7}} - 1} \quad j = 1,2,\ldots \qquad (5.5.3.2)$$

El muestreo propuesto para el eje de la escala cumple con el teorema de Littlewood-Paley ya que:

$$\lim_{j\to+\infty}\frac{s_{j+1}}{s_j}=\lim_{j\to+\infty}\frac{e^{\frac{j+1}{7}}\cdot\left(e^{\frac{2j}{7}}-1\right)}{e^{\frac{j}{7}}\cdot\left(e^{\frac{2(j+1)}{7}}-1\right)}=\lim_{j\to+\infty}\frac{e^{\frac{3j+1}{7}}}{e^{\frac{3j+2}{7}}}=e^{-\frac{1}{7}}\neq 1 \qquad (5.5.3.3)$$

De modo que no existe pérdida de información durante la discretización propuesta para la trasformada wavelet continua.

5.5.4 Implantación del algoritmo

Al programar el algoritmo descrito, es necesario fijar un límite para el soporte numérico del wavelet. Esto se debe a que la distribución gama, a pesar de tender a cero en infinito, sólo alcanza dicho valor en el tiempo cero. Esto obliga a establecer un límite en la precisión numérica tal que el tiempo de cómputo sea razonable para la aplicación. Para este trabajo se consideró adecuado un límite de 0.01. Es decir, el wavelet madre se considera toma un valor de cero cuando su envolvente es menor a dicho umbral. Esto puede observarse dentro de la función earDWT en el anexo C.

A la salida del bloque de extracción de características se tiene entonces un conjunto de señales. Cada una de ellas presenta la información que contiene su entrada para las bandas críticas definidas por la escala Bark. De modo que el número de señales de salida dependerá únicamente de la frecuencia de muestreo con la que se adquirieron los datos de audio. Específicamente, la relación entre la cantidad de descomposiciones a calcular y la frecuencia de muestreo de las señales de voz está dada como:

$$j_{\max}=\operatorname{int}\left(7\cdot\ln\left(\frac{f_s}{1300}+\sqrt{\left(\frac{f_s}{1300}\right)^2+1}\right)\right) \qquad (5.5.4.1)$$

Un ejemplo de lo anterior sería al trabajar con señales de voz muestreadas a 8 kiloHertz. Para este caso particular, se tiene un valor de $j_{\max}$ de diecisiete.

5.6 Construcción de un vector de parámetros

Esta etapa se encarga de generar la información que se empleará como entrada para el clasificador; para ello se recurre a las características de la señal obtenidas de las etapas anteriores. Específicamente estas son la trasformada wavelet de la señal y la clasificación de sus segmentos como vocalizados o no vocalizados.

Al efectuar la segmentación de la señal, se eligieron tiempos de trama y ventana adecuados para el reconocimiento de fonemas individuales. Aprovechando esto, se busca crear un vector de parámetros para cada segmento. En este punto, las porciones de la señal que fueron identificadas como no vocalizadas son descartadas. Esto evitará que el clasificador

deba ajustarse a discriminar los sonidos no vocalizados y pueda estar dirigido únicamente a distinguir entre las vocales.

La información obtenida de la etapa de extracción de características no se encuentra dividida en segmentos. Esto obliga a separar nuevamente cada una de las señales que entrega dicho bloque siguiendo un algoritmo similar al utilizado en el bloque de segmentación. La única diferencia con respecto a la segmentación inicial es que la ventana a utilizar es una ventana rectangular en lugar de una ventana de Hamming. Esto para evitar cálculos innecesarios ya que, como se verá mas adelante, los cambios abruptos al inicio y final del segmento no tendrán efectos indeseables en esta etapa.

El vector de parámetros, al ser la entrada para un clasificador heurístico, no debe ser demasiado grande. Si lo fuera, la complejidad del algoritmo de clasificación lo haría demasiado lento y podría incluso evitar que el error que éste comete converja a un valor pequeño. Para evitar la situación anterior, se busca extraer la característica más representativa de cada una de las bandas empleadas para la descomposición. En el contexto del análisis funcional, es común elegir como característica principal de una función su norma para el espacio que la contenga. Lo anterior, al trabajar con señales de energía finita, estaría dado como su norma en L^2. Sin embargo un parámetro equivalente utilizado con frecuencia en el contexto del análisis de señales es la energía de la misma. Lo anterior puede calcularse para cualquier señal $x(t)$ como:

$$E\{x\} = \|x(t)\|_{L^2}^2 = \langle x, x \rangle = \int_{-\infty}^{\infty} x(t) \cdot \overline{x}(t)\, dt \tag{5.6.1}$$

Esto puede rescribirse para señales en tiempo discreto de longitud n como:

$$E = \sum_{k=1}^{n} |x[k]|^2 \tag{5.6.2}$$

Además de la información que presenta directamente la transformada wavelet, resulta útil para el clasificador contar con la energía total de la señal previa a la descomposición. Esto se debe a que, al trabajar con señales físicas, la energía total de un sistema siempre es conservada. Lo anterior, en conjunto con la normalización efectuada en la primera etapa de este algoritmo, permitirá al clasificador realizar una discriminación entre los posibles fonemas vocálicos contenidos en un segmento, considerando las diferencias energéticas involucradas en la producción las distintas vocales.

En el diseño del vector de parámetros se debe contemplar la naturaleza del clasificador a utilizar. Específicamente, se propone el uso de una red neuronal artificial con una arquitectura de alimentación hacia delante. Al no poseer elementos de retraso, y recordando que las señales de voz no son estacionarias, encontramos la necesidad de incluir en el vector de parámetros información sobre la dinámica espectral de la entrada al

sistema. Esto se logra al añadir componentes definidos como el cambio entre el vector anterior y el actual.

Tomando en cuenta lo anterior, podemos ahora definir al vector de parámetros como la concatenación de los siguientes componentes:

- Energía total del segmento
- Energía contenida en cada uno de los niveles de la descomposición usando la transformada wavelet
- Cambio en la energía de la señal con respecto al segmento anterior
- Cambio en la energía contenida en cada descomposición con respecto a la trama previa

La longitud del vector de parámetros dependerá entonces del número de bandas utilizadas para la descomposición. Este valor, como se vio con anterioridad, está a su vez limitado por la frecuencia de muestreo de la señal de entrada. En particular, si se utiliza una frecuencia de muestreo de 8 kiloHertz, el vector de parámetros estará formado por 36 componentes.

El algoritmo propuesto para esta etapa fue programado dentro de la función featureExtraction disponible en el anexo C.

5.7 Clasificación por medio de una red neuronal artificial

Al llegar a esta etapa, los datos de entrada se han reducido a un vector de parámetros por cada segmento vocalizado de la señal a analizar. Ahora se pretende agrupar cada uno de esos vectores en cinco clases representando a las vocales A, E, I, O y U. Para lograr lo anterior se propone el uso de una red neuronal artificial. Ésta tendrá una arquitectura de varias capas ante-alimentadas y será entrenada empleando el algoritmo de propagación hacia atrás.

De un modo más específico, la red neuronal artificial propuesta consta de una capa de entrada cuyo número de nodos es igual a la longitud del vector de parámetros; dos capas ocultas de dimensiones configurables completamente interconectadas y una capa de salida formada por cinco nodos (uno por cada clase). Los nodos de la capa de entrada simplemente se utilizan para repetir la información de entrada y entregarla a cada nodo de la primera capa intermedia. Ambas capas ocultas utilizan como función de activación la tangente hiperbólica definida de la siguiente manera:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5.7.1}$$

Por último, los nodos de la capa de salida tienen un comportamiento lineal y su salida es igual a su entrada.

Cada nodo, a excepción de los de la capa inicial, determina su entrada como la suma ponderada de todas las señales de salida de la capa anterior y un nivel de referencia o desplazamiento. Los pesos y niveles de referencia son determinados durante un proceso de entrenamiento siguiendo, como se mencionó con anterioridad, el algoritmo de propagación hacia atrás.

La red neuronal artificial es entrenada mediante un conjunto de datos obtenidos de archivos de audio. Inicialmente se realiza la generación de vectores de parámetros para cada segmento de los archivos de entrenamiento. Al mismo tiempo se crean vectores de salida deseados con base en el conocimiento previo del contenido fonético del archivo.

Antes del entrenamiento, cada una de las componentes de los vectores de entrada y salida es normalizada. Esto se logra mediante dos operaciones. La primera consiste en restar a cada uno de los elementos de cada vector la media encontrada para su valor entre los datos de entrenamiento. A continuación se divide el resultado entre la desviación estándar que presenta la componente en cuestión. Resumiendo lo anterior, se tiene la siguiente transformación para la i-ésima componente de cada vector V:

$$\hat{V}_i = \frac{V_i - \mu[V_i]}{\sigma[V_i]} \tag{5.7.2}$$

Como se mencionó, este proceso se realiza tanto en las entradas de entrenamiento como en las salidas deseadas y tiene la finalidad de evitar que alguna componente notablemente mayor en magnitud que el resto tenga un impacto más significativo en los pesos obtenidos luego del entrenamiento.

Cuando la red se utiliza para la clasificación de un vector de parámetros, éste es nuevamente normalizado con base en la media y desviación estándar de los datos de entrenamiento. Por otro lado, se realiza la operación inversa a la salida de la red; es decir, los resultados que entregue son multiplicados por la desviación estándar de las salidas deseadas y a estos valores se les suma la media determinada al momento de entrenar. De este modo se garantiza que la normalización durante el entrenamiento no perjudicará el desempeño de la red neuronal artificial.

Como parámetros para el algoritmo de propagación hacia atrás, se estableció un factor de aprendizaje inicial de 0.01. Éste es incrementado en un 5% con cada iteración de entrenamiento que logra disminuir el error cuadrático medio de la red. Asimismo, el factor de aprendizaje se reduce en un 30% luego de una iteración que incrementa el criterio de error mencionado en más de un 4%. Estos datos fueron elegidos como aquellos que, durante las pruebas, redujeron el error del clasificador en menos tiempo.

Para detener el entrenamiento se consideró un error cuadrático medio deseado de 0.1 o un número máximo de cinco mil iteraciones.

La implantación del algoritmo permite que todos los datos anteriores puedan modificarse fácilmente si se deseara en el código de las funciones train y initThresholds.

Asimismo, puede consultarse la implantación del algoritmo para la generación de datos de entrenamiento y para la clasificación de un vector de parámetros dentro de los archivos createData.m y forward.m. Todo esto se encuentra disponible en el anexo C.

Al tener en cuenta la arquitectura de la red neuronal artificial y las características de sus nodos de salida, es claro que el clasificador entregará, por cada segmento de la señal original, un vector de cinco componentes. Cada componente representará la certeza con que se ha identificado en dicho segmento alguna de las cinco vocales.

5.8 Filtrado de los resultados

La última etapa del algoritmo expuesto emplea la salida del clasificador para generar una cadena de caracteres. Se desea representar en ella cada uno de los fonemas vocálicos detectados en la señal, así como las secciones donde no se observó ninguno.

El primer paso de esta sección consiste en analizar los resultados de la red neuronal artificial para cada segmento vocalizado. Si la etapa de clasificación no fue capaz de distinguir con precisión que algún segmento se trataba de una de las cinco vocales, el mismo es representado por el símbolo "_" indicando la ausencia de componentes vocálicas o una transición entre fonemas.

Lo anterior se determina con base en dos umbrales configurables, dependientes de la calidad de los datos de entrenamiento utilizados. En concreto, se busca que la certeza con que el segmento fue clasificado como alguna vocal sea mayor a uno de estos umbrales. Asimismo, se espera que la red neuronal fije probabilidades de aparición menores a un segundo umbral para cuatro de las cinco vocales.

En caso de cumplir con los criterios mencionados, se busca la componente máxima del vector de salidas del clasificador y, con base en su posición, se asigna al segmento uno de los símbolos "a", "e", "i", "o" o "u".

Luego de los pasos anteriores se tendrá una cadena de caracteres cuya longitud será igual a la duración de la señal dividida entre el tiempo por trama. Con la finalidad de eliminar posibles errores en la clasificación de algunos segmentos, esta cadena es tratada mediante un filtro modal.

Inicialmente, se eliminan los símbolos "_" que aparezcan entre cúmulos grandes de vocales. En concreto, se ubica una ventana sobre cada uno de estos símbolos y si la mitad o más de la cadena obtenida está formada por vocales, el símbolo se elimina.

El siguiente paso involucra la división de la cadena en cúmulos de vocales, separados por segmentos no vocalizados o de transición. Cada uno de estos conjuntos es filtrado mediante una ventana que lo recorre caracter a caracter y asigna a cada posición la moda

de los elementos dentro de la ventana. Los grupos resultantes son concatenados con el símbolo "_" entre ellos. Por último, se genera una cadena fácil de visualizar al eliminar los caracteres repetidos; es decir, no se permite que dos caracteres contiguos sean idénticos.

La cadena resultante sería entonces la salida del sistema diseñado e implantado en este proyecto. El código generado en Matlab para esta etapa puede apreciarse dentro de la función getVowels en el anexo C.

Con lo anterior concluimos la presentación del sistema realizado durante el desarrollo de este proyecto. Como se ha mencionado, el mismo fue implantado bajo el ambiente de Matlab y el código completo se encuentra disponible en el anexo C. En el siguiente capítulo, presentaremos los resultados obtenidos al permitir que el sistema expuesto opere sobre grabaciones de voz tanto de un hablante normal como de un locutor laringectomizado.

Capítulo VI

Resultados

6.1 Condiciones de la realización de las pruebas y resultados.

Para la realización de las siguientes pruebas, fueron grabados dos individuos con las características que se describen a continuación:

- Hombre de 23 años de edad. (H23)
- Hombre laringectomizado mayor de 30 años de edad. (HL30)

Las grabaciones de los dos individuos fueron realizadas en condiciones de ambiente y equipo diferente. A continuación se describirán cada una de éstas.

El equipo que fue utilizado para la grabación de la voz del individuo H23 fue el siguiente:

Micrófono:

Sony F-V220 Dynamic Microphone.

Computadora:

Modelo: DELL Inspiron 8200

Procesador: Pentium 4-M 1.7GHz

Memoria RAM: 512 MB

Tarjeta de sonido: Crystal WDM Audio, Cirrus Logic Inc.

Contexto ambiental en el que fueron realizadas las grabaciones de voz del individuo H23:

Habitación de casa con condiciones de ruido controladas.

El equipo que fue utilizado para la grabación de voz esofágica del individuo HL30 fue el siguiente:

Grabadora de cinta magnética. Posteriormente el audio fue digitalizado.

Contexto ambiental en el que se realizaron las grabaciones de voz esofágica del individuo HL30:

Cubículo ubicado dentro del Instituto Nacional de Rehabilitación. Las condiciones de ruido no fueron controladas.

En ambas circunstancias, los datos de audio están almacenados en un contenedor WAV. Todos los archivos son monaurales y su frecuencia de muestreo es de 8 kHz.

En el caso de los archivos de voz para el individuo H23, se grabaron con 16 bits por muestra mientras que los archivos de voz para el individuo HL30 fueron digitalizados con una resolución de 8 bits por muestra.

Las pruebas fueron realizadas en el siguiente equipo:

Computadora Modelo: Toshiba Satellite 2405

Procesador: Pentium 4 1.6 GHz

Memoria RAM: 512 MB

Sistema Operativo: Microsoft Windows XP

La plataforma de desarrollo y pruebas fue la siguiente:

MATLAB Versión 6.5.0 R-13

6.2 Gráficas de las diferentes etapas del sistema para voz normal y voz esofágica

A continuación se presenta la cadena de caracteres que arroja el sistema propuesto en su fase final después de procesar un archivo de audio y las gráficas que representan a la señal original y las diferentes etapas por las que ésta pasa. Se utilizan cinco grabaciones de voz normal y cinco grabaciones de voz esofágica en las que, en conjunto se presentan cada una de las cinco vocales.

En la realización de las siguientes pruebas se utilizaron los datos de entrenamiento que a continuación se presentan:

Para el individuo H23 se emplearon 10 archivos de cada vocal. Los archivos contenían la vocal individual o bien la vocal repetida dentro de una sola palabra.

Para el individuo HL30 se utilizaron 53 archivos para la vocal "a", 16 archivos para la vocal "e", 15 archivos para la vocal "i", 28 archivos para la vocal "o" y 13 para la vocal "u". De la misma manera, la información que contenían estos archivos podía ser la vocal individual o la vocal repetida dentro de una sola palabra.

El entrenamiento de la red neuronal (incluyendo la extracción de características de los datos de entrenamiento) con la información del individuo H23, realizó 5000 iteraciones demorándose un tiempo de 10 minutos 37 segundos, logrando un error cuadrático medio final de 8.71543%. En el caso del proceso con los datos del individuo HL30 se realizaron

también 5000 iteraciones, se demoró un tiempo de 12 minutos y 55 segundos, obteniendo un error cuadrático medio final de 19.701%

Palabra: "A E I O U"

Individuo H23

Cadena de caracteres a la salida del sistema:

_a_e_i_o_u_

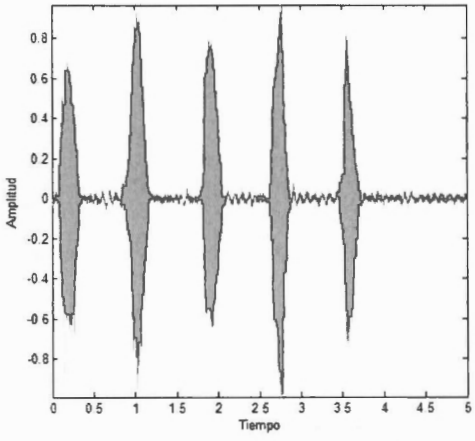


Figura 6.2.1. Señal de voz original.

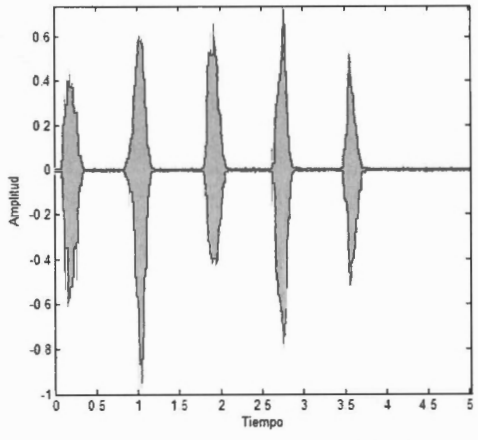


Figura 6.2.3. Señal de voz después del preprocesamiento.

Individuo HL30

Cadena de caracteres a la salida del sistema:

_a_e_i_o_u_

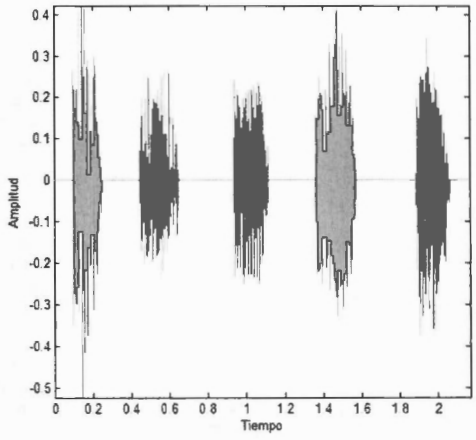


Figura 6.2.2. Señal de voz esofágica original.

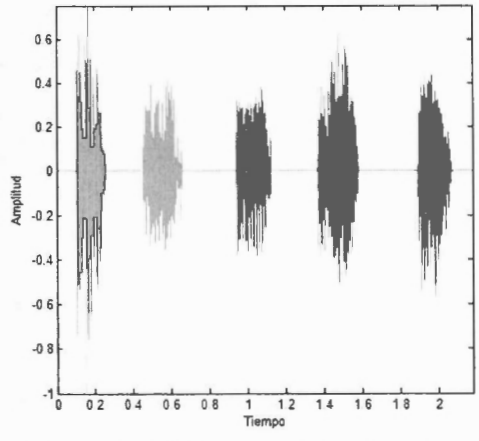


Figura 6.2.4. Señal de voz esofágica después del preprocesamiento.

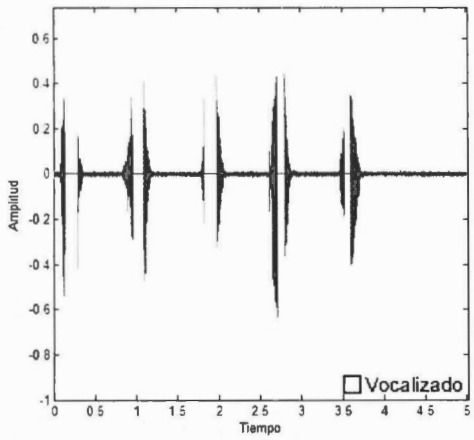


Figura 6.2.5. Componentes vocalizados y no vocalizados de la señal de voz.

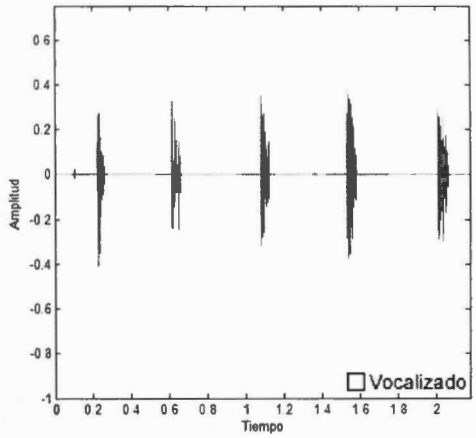


Figura 6.2.6. Componentes vocalizados y no vocalizados de la señal de voz esofágica.

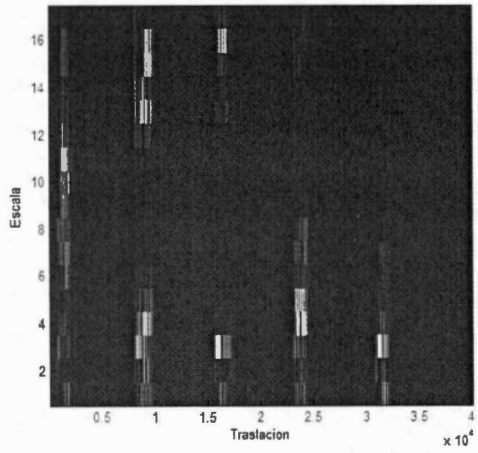


Figura 6.2.7. Trasformada wavelet de la señal de voz.

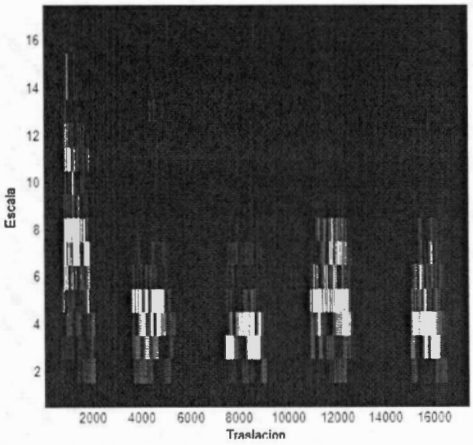


Figura 6.2.8. Trasformada wavelet de la señal de voz esofágica.

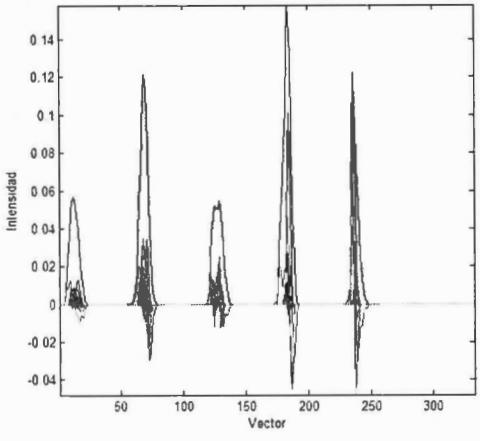


Figura 6.2.9. Vector de parámetros extraídos de la señal de voz.

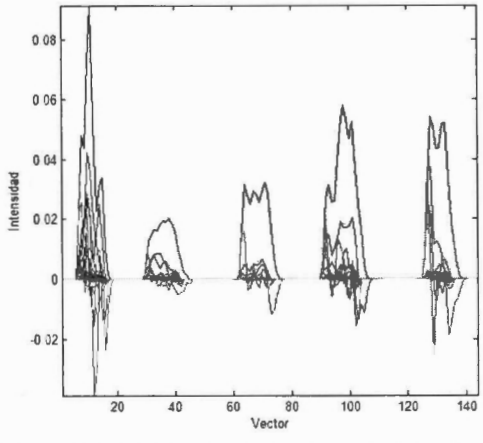


Figura. 6.2.10 Vector de parámetros extraídos de la señal de voz esofágica.

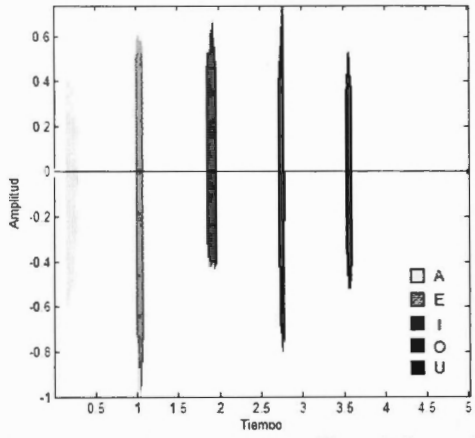


Figura 6.2.11. Componentes vocálicos de la señal de voz.

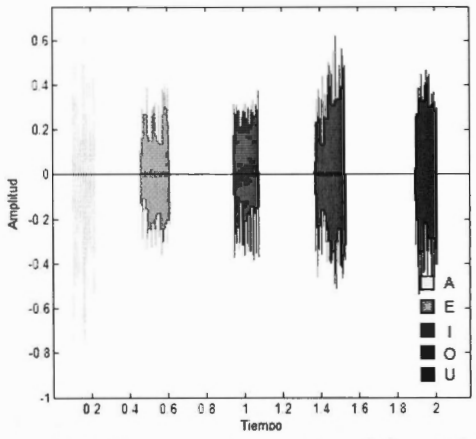


Figura 6.2.12. Componentes vocálicos de la señal de voz esofágica.

Palabra: "Bala"

Individuo H23

Cadena de caracteres a la salida del sistema:

_a_a_

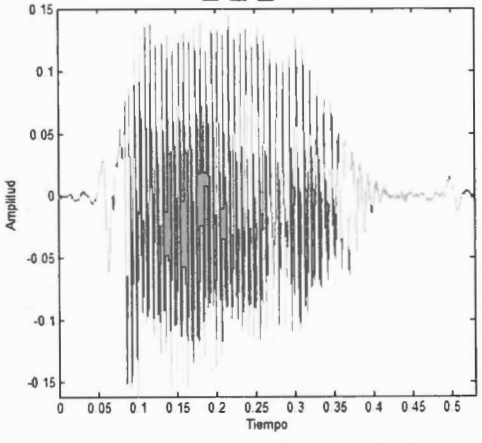


Figura 6.2.13. Señal de voz original.

Individuo HL30

Cadena de caracteres a la salida del sistema:

_a_a_

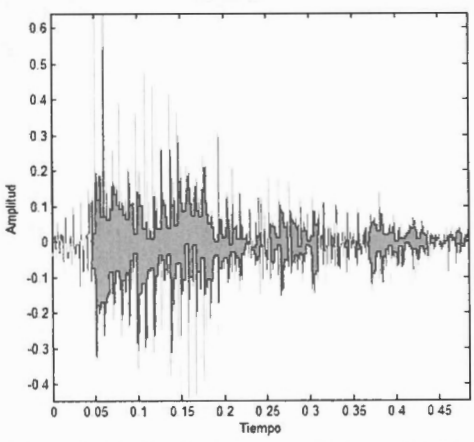


Figura 6.2.14. Señal de voz esofágica original.

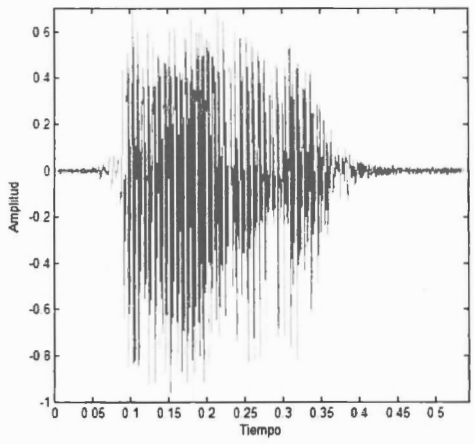


Figura 6.2.15. Señal de voz después del preprocesamiento.

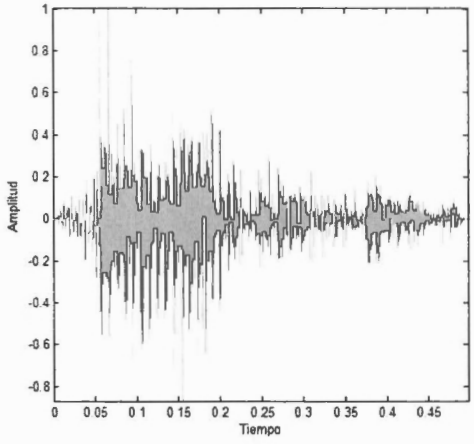


Figura 6.2.16. Señal de voz esofágica después del preprocesamiento.

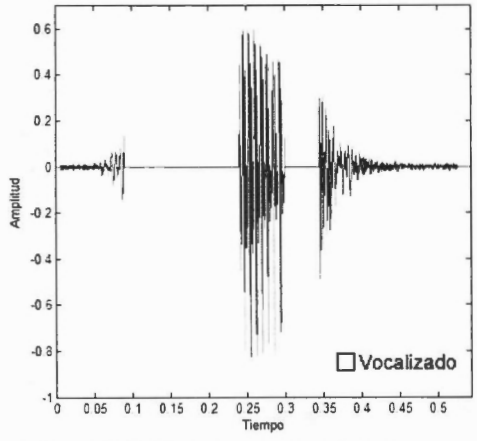


Figura 6.2.17. Componentes vocalizados y no vocalizados de la señal de voz.

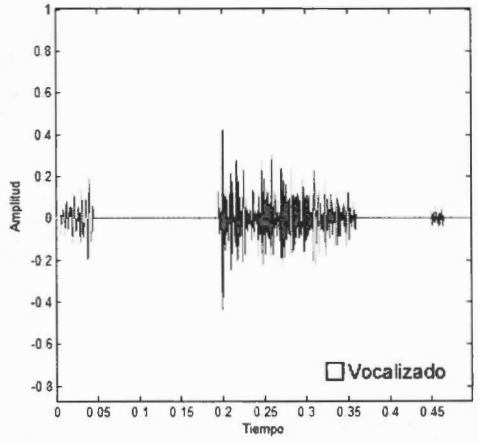


Figura 6.2.18. Componentes vocalizados y no vocalizados de la señal de voz esofágica.

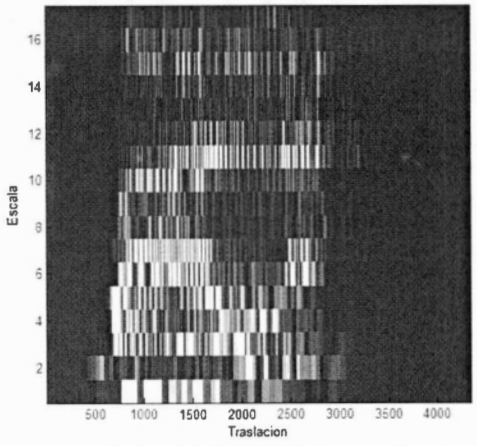


Figura 6.2.19. Trasformada wavelet de la señal de voz.

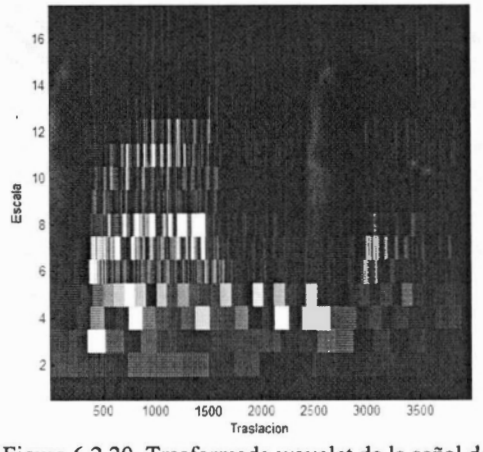


Figura 6.2.20. Trasformada wavelet de la señal de voz esofágica.

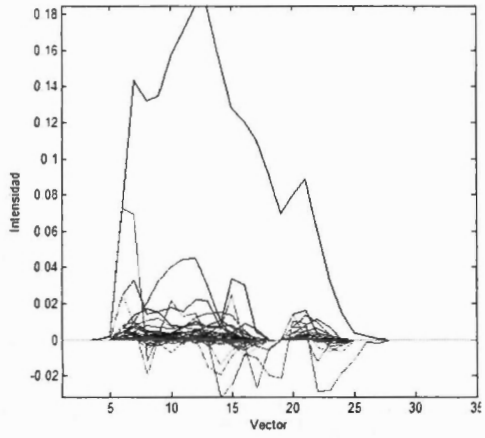


Figura 6.2.21. Vector de parámetros extraídos de la señal de voz.

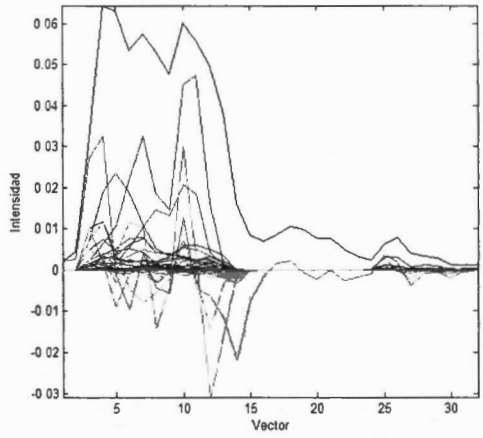


Figura 6.2.22. Vector de parámetros extraídos de la señal de voz esofágica.

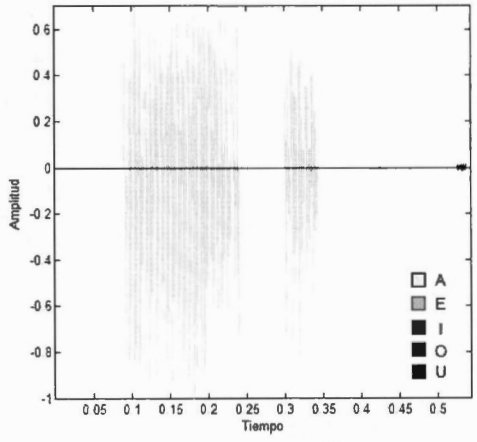


Figura 6.2.23. Componentes vocálicos de la señal de voz.

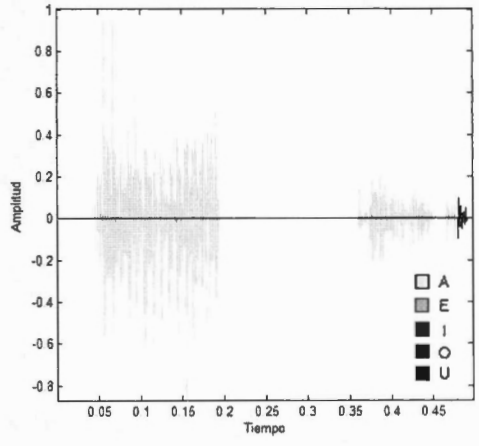


Figura 6.2.24. Componentes vocálicos de la señal de voz esofágica.

Palabra: "Besar"

Individuo H23

Cadena de caracteres a la salida del sistema:

_e_a_

Individuo HL30

Cadena de caracteres a la salida del sistema:

_e_a_

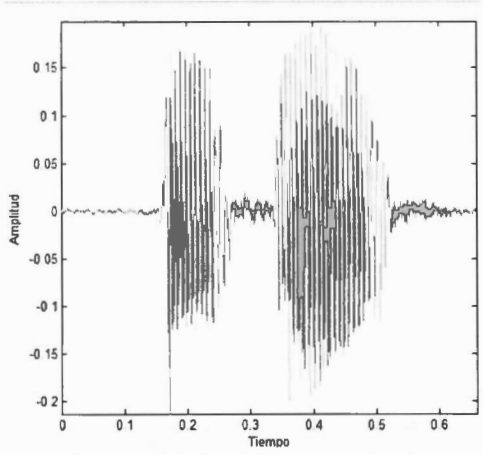


Figura 6.2.25. Señal de voz original.

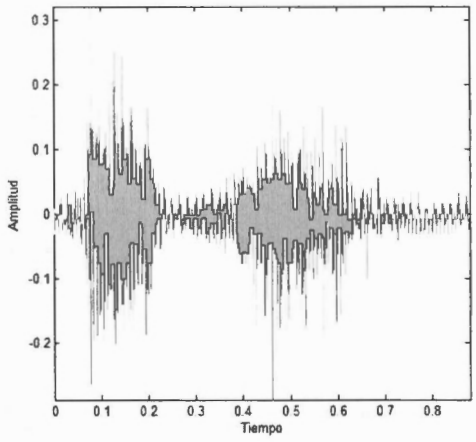


Figura 6.2.26. Señal de voz esofágica original.

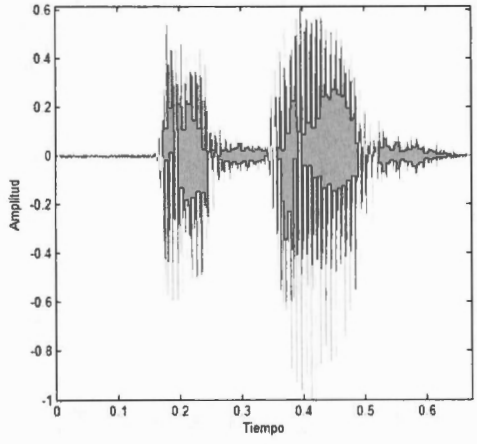


Figura 6.2.27. Señal de voz después del preprocesamiento.

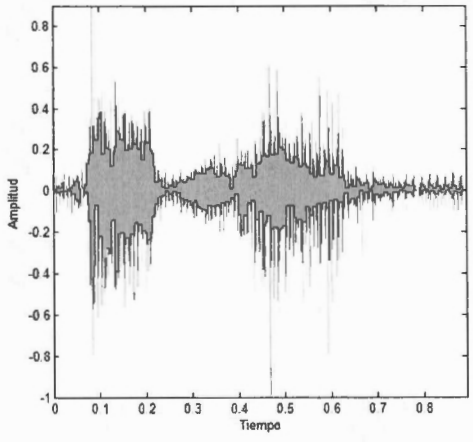


Figura 6.2.28. Señal de voz esofágica después del preprocesamiento.

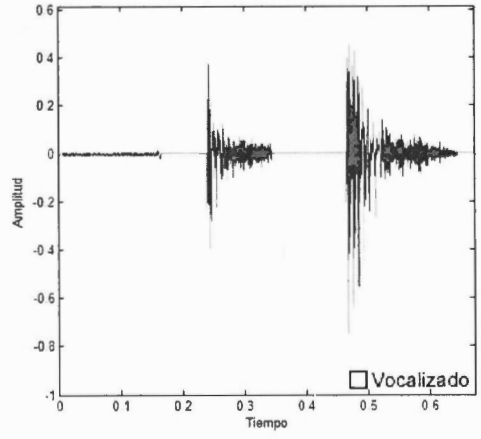


Figura 6.2.29. Componentes vocalizados y no vocalizados de la señal de voz.

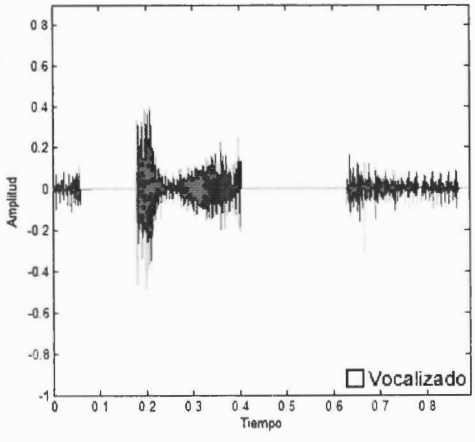


Figura 6.2.30. Componentes vocalizados y no vocalizados de la señal de voz esofágica.

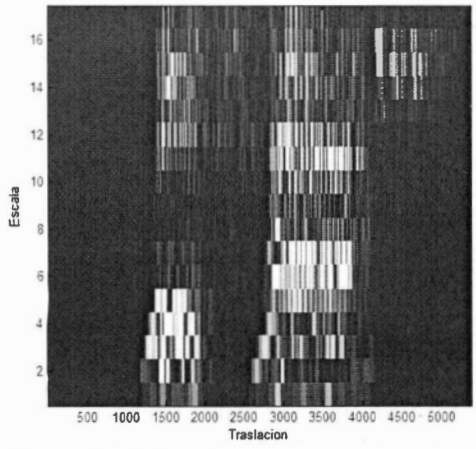


Figura 6.2.31. Trasformada wavelet de la señal de voz.

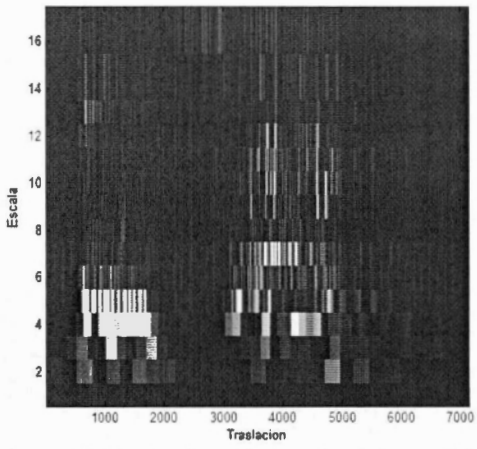


Figura 6.2.32. Trasformada wavelet de la señal de voz esofágica.

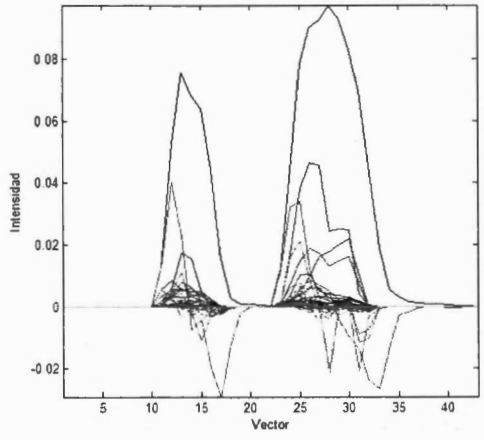


Figura 6.2.33. Vector de parámetros extraídos de la señal de voz.

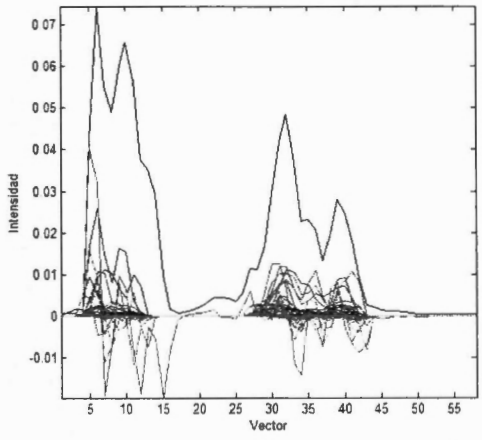


Figura 6.2.34. Vector de parámetros extraídos de la señal de voz esofágica.

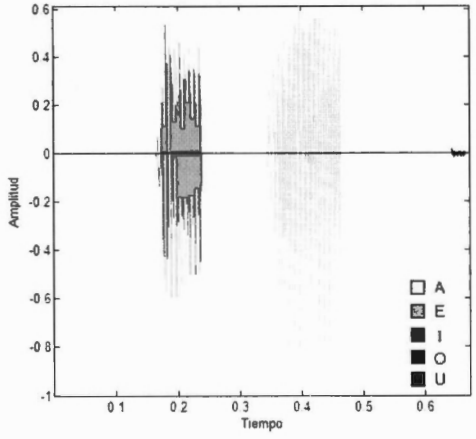


Figura 6.2.35. Componentes vocálicos de la señal de voz.

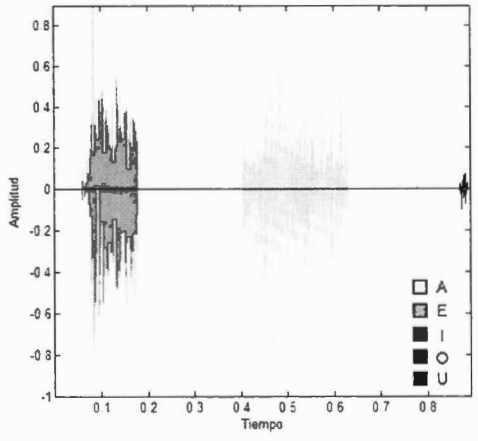


Figura 6.2.36. Componentes vocálicos de la señal de voz esofágica.

Palabra: "Cupo"

Individuo H23

Cadena de caracteres a la salida del sistema:

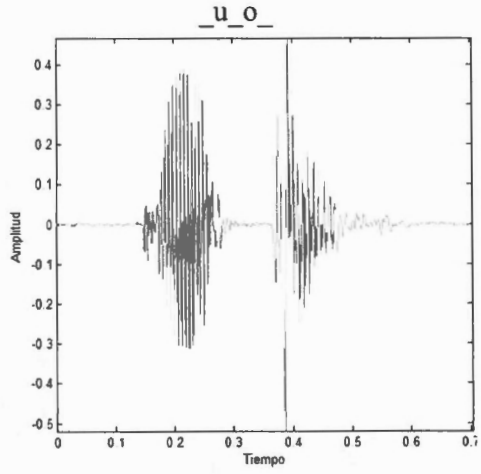


Figura 6.2.37. Señal de voz original.

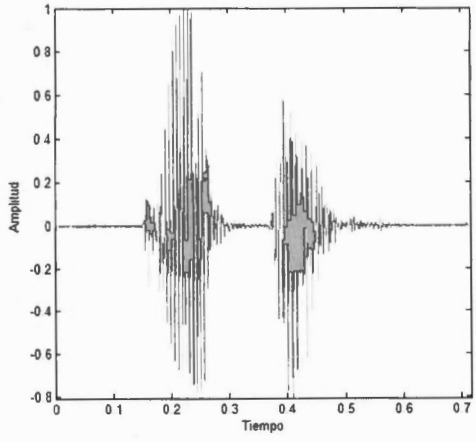


Figura 6.2.39. Señal de voz después del preprocesamiento.

Individuo HL30

Cadena de caracteres a la salida del sistema:

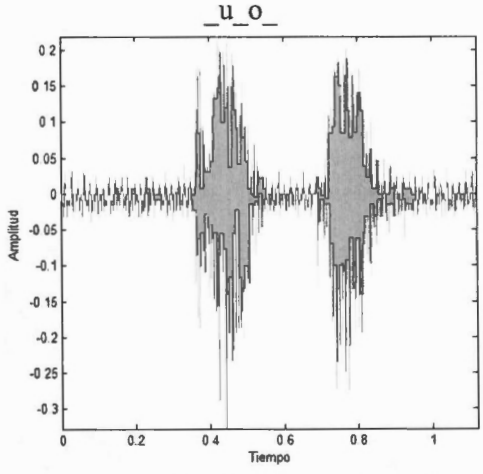


Figura 6.2.38. Señal de voz esofágica original.

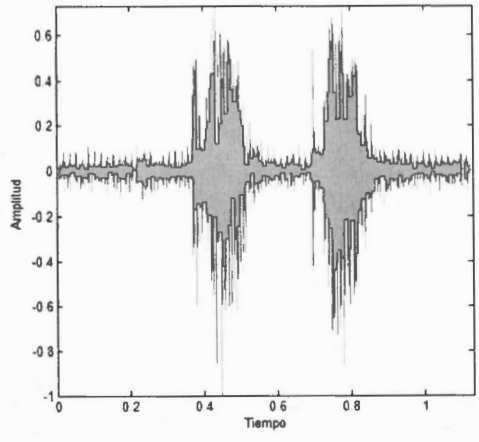


Figura 6.2.40. Señal de voz esofágica después del preprocesamiento.

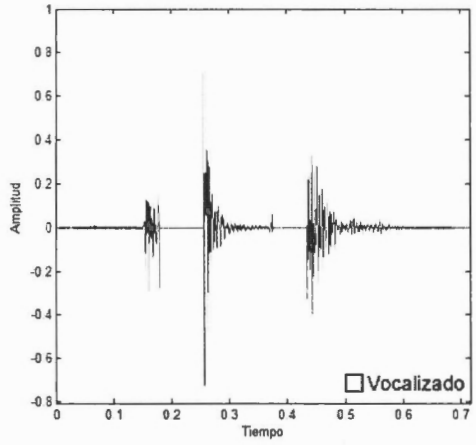


Figura 6.2.41. Componentes vocalizados y no vocalizados de la señal de voz.

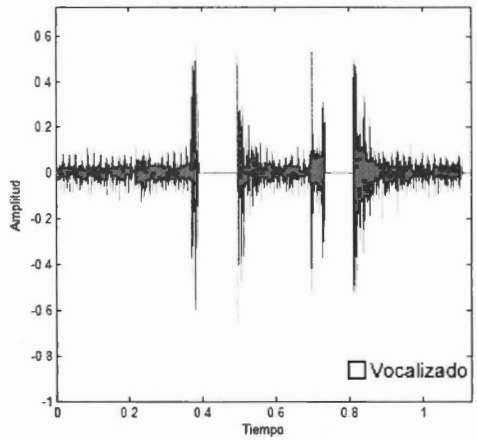


Figura 6.2.42. Componentes vocalizados y no vocalizados de la señal de voz esofágica.

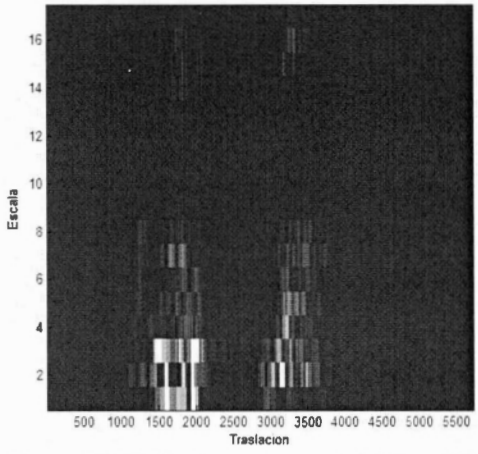


Figura 6.2.43. Trasformada wavelet de la señal de voz.

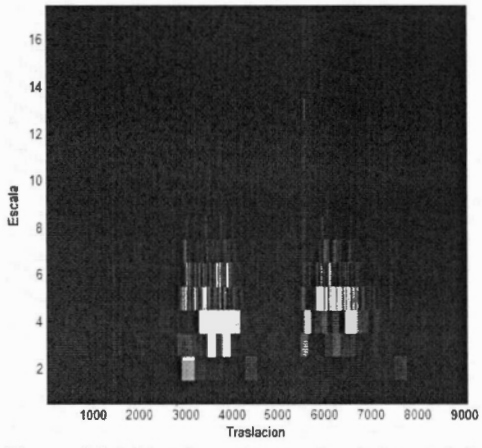


Figura 6.2.44Trasformada wavelet de la señal de voz esofágica.

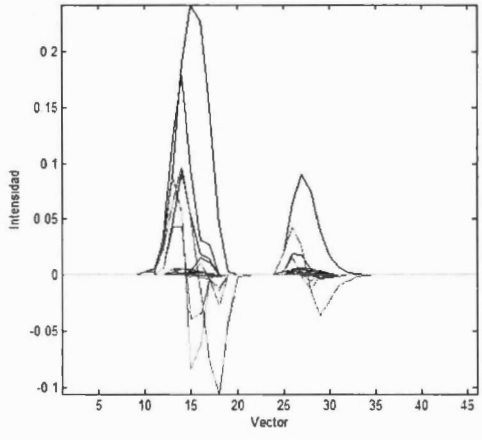


Figura 6.2.45. Vector de parámetros extraídos de la señal de voz.

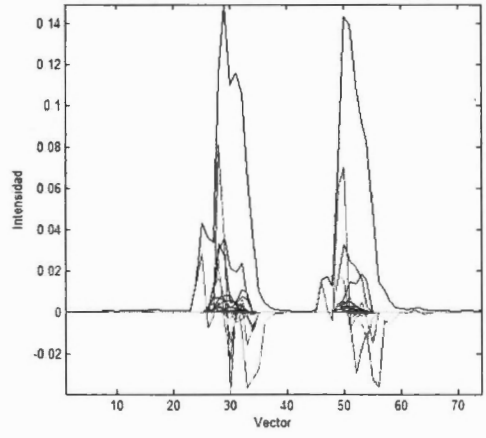


Figura 6.2.46. Vector de parámetros extraídos de la señal de voz esofágica.

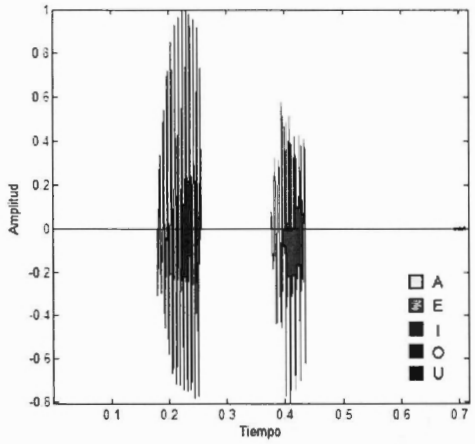


Figura 6.2.47. Componentes vocálicos de la señal de voz.

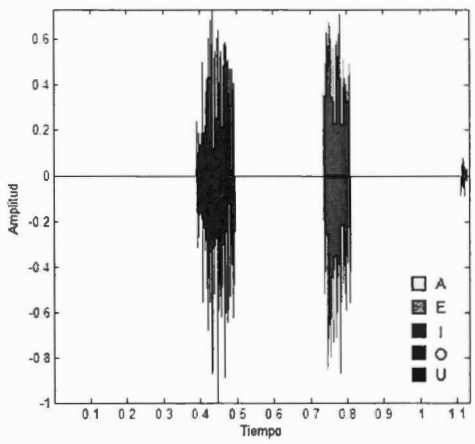


Figura 6.2.48. Componentes vocálicos de la señal de voz esofágica.

Palabra:
"Adicto"

Individuo H23

Cadena de caracteres a la salida del sistema:

_a_i_o_

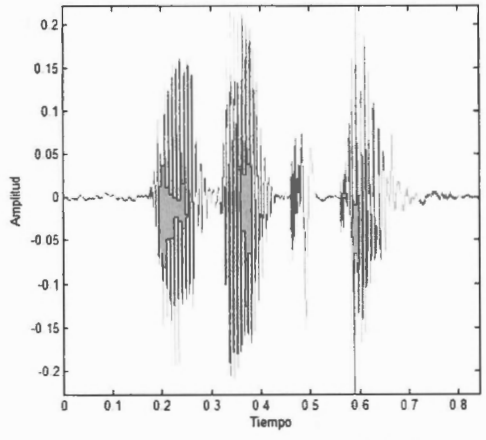


Figura 6.2.49. Señal de voz original.

Individuo HL30

Cadena de caracteres a la salida del sistema:

_a_i_o_

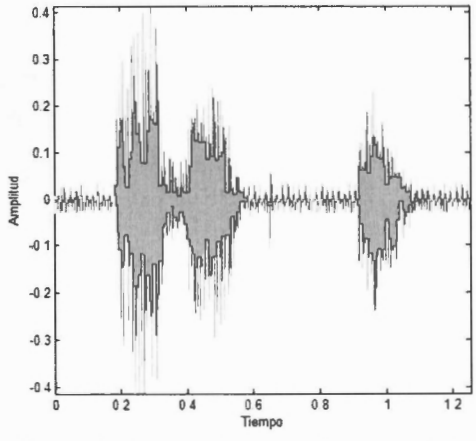


Figura 6.2.50. Señal de voz esofágica original.

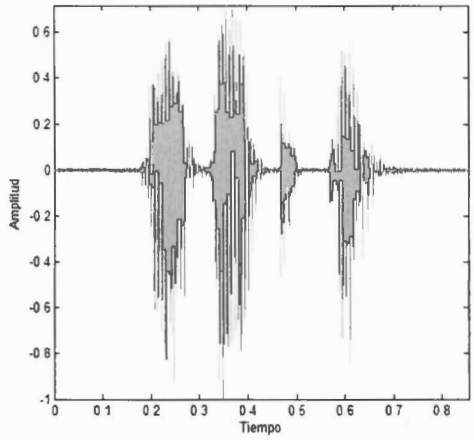


Figura 6.2.51. Señal de voz después del preprocesamiento.

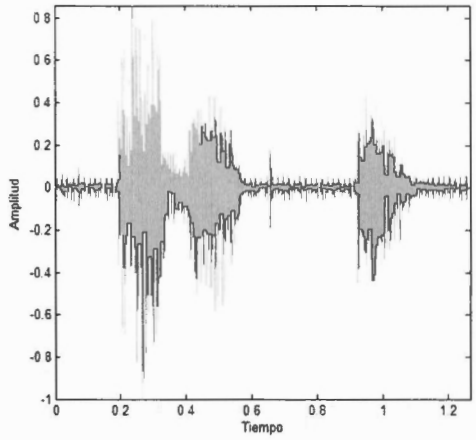


Figura 6.2.52. Señal de voz esofágica después del preprocesamiento.

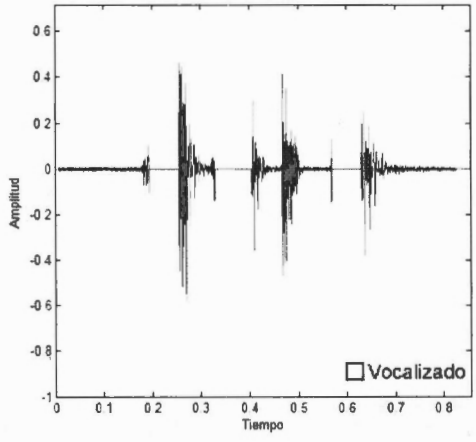


Figura 6.2.53. Componentes vocalizados y no vocalizados de la señal de voz.

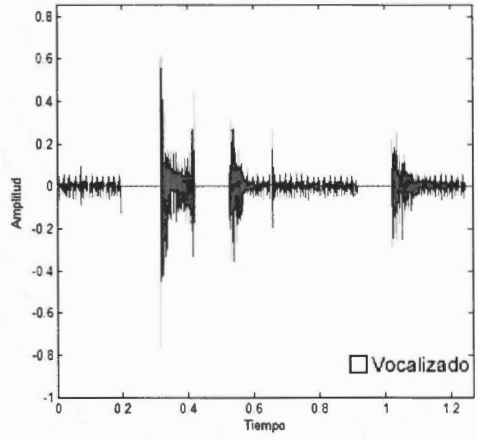


Figura 6.2.54. Componentes vocalizados y no vocalizados de la señal de voz esofágica.

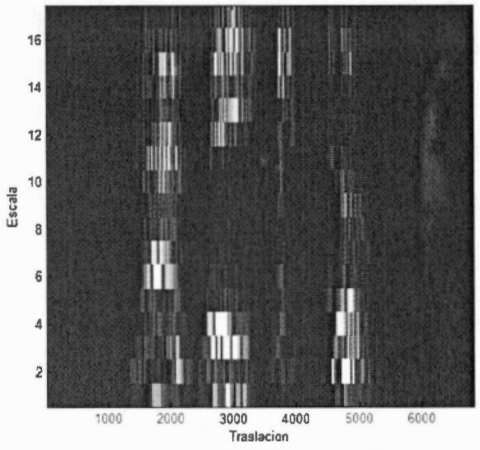


Figura 6.2.55. Trasformada wavelet de la señal de voz.

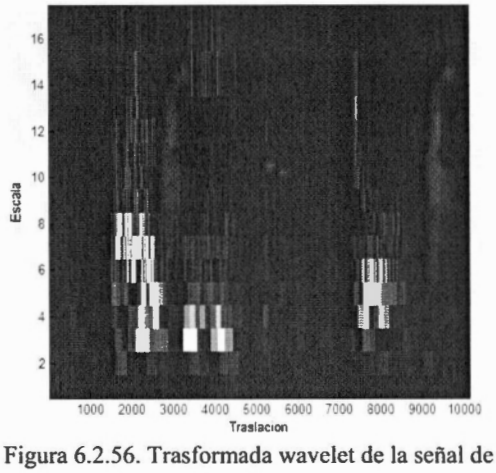


Figura 6.2.56. Trasformada wavelet de la señal de voz esofágica.

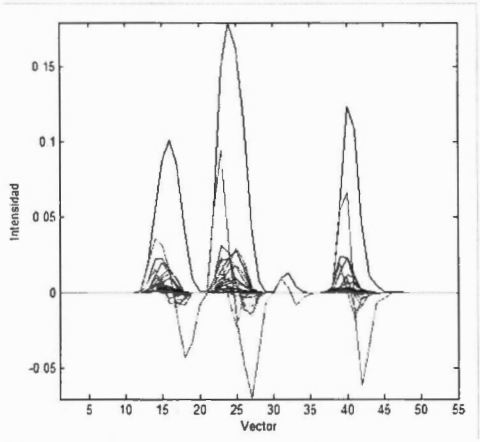


Figura 6.2.57. Vector de parámetros extraídos de la señal de voz.

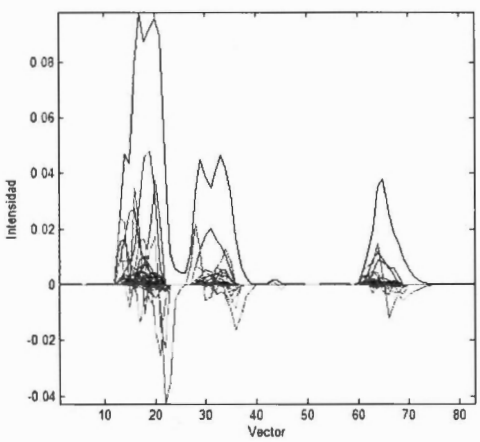


Figura 6.2.58. Vector de parámetros extraídos de la señal de voz esofágica.

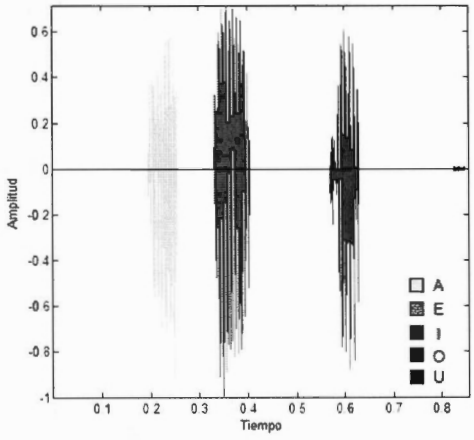


Figura 6.2.59. Componentes vocálicos de la señal de voz.

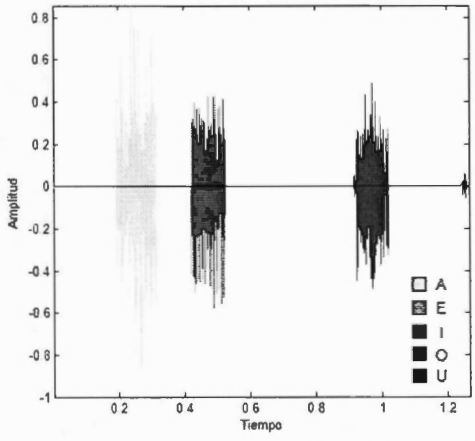


Figura 6.2.60. Componentes vocálicos de la señal de voz esofágica.

6.3 Porcentaje de éxito en el reconocimiento de las vocales A E I O U para voz normal y voz esofágica.

Los datos de entrenamiento que se emplearon para las pruebas con el individuo H23 fueron 10 archivos de cada vocal. Los archivos la contenían de forma individual o ésta repetida dentro de una sola palabra.

Para las pruebas de las palabras pronunciadas por el individuo HL30 se utilizaron 13 archivos para cada vocal. De la misma manera, la información que contenían estos archivos podía ser la vocal individual o la vocal repetida dentro de una sola palabra.

El entrenamiento de la red neuronal (incluyendo la extracción de características de los datos de entrenamiento) para el individuo H23 se realizó con 5000 iteraciones demorándose un tiempo de 10 minutos 37 segundos, logrando un error cuadrático medio final de 8.71543%. Para el caso del individuo HL30 se realizaron también 5000

iteraciones, se demoró un tiempo de 7 minutos y 3 segundos, obteniendo un error cuadrático medio final de 15.2967%

Para la obtención de resultados se reconocieron un total de 37 palabras del individuo H23 y 85 palabras para el individuo HL30.

Las siguientes gráficas indican la relación de las veces que se encontró cada una de las vocales donde se esperaba, entre el numero real de las apariciones de dicha vocal en las palabras usadas para la prueba.

Individuo H23

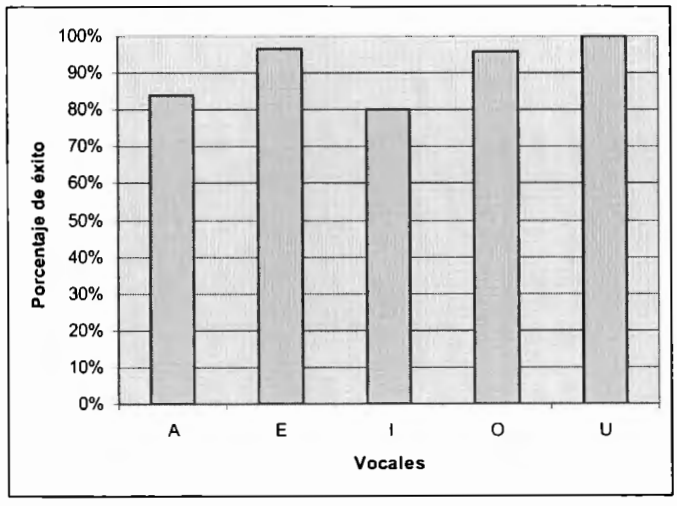


Figura 6.3.1

Individuo HL30

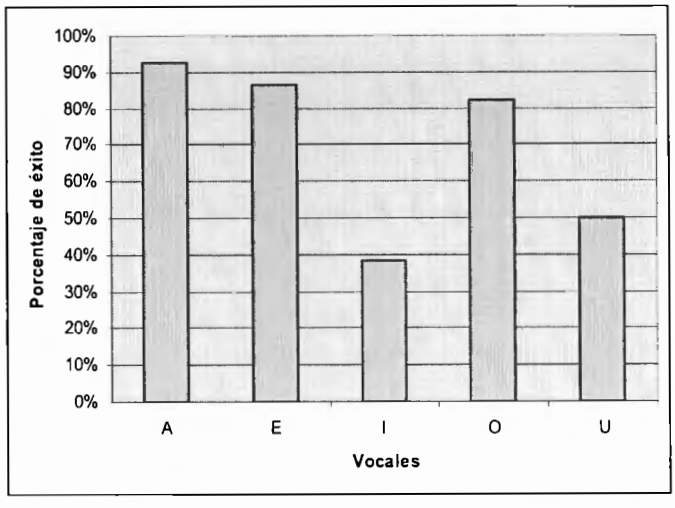


Figura 6.3.2

6.4 Análisis de resultados

El sistema planteado en este proyecto busca el reconocimiento de los fonemas vocálicos del español. Esto es, se desean identificar las vocales A, E, I, O y U dentro de una grabación de voz esofágica. Al buscar el reconocimiento de un conjunto específico de fonemas, el desempeño del sistema se puede evaluar empleando dos criterios: Su capacidad para distinguir entre las componentes de una grabación que no pertenecen al conjunto que se desea identificar. Y el éxito obtenido al agrupar las secciones de voz de interés dentro de sus respectivas clases.

Uno de los elementos principales del algoritmo presentado, que permite descartar las secciones de audio que no contienen vocales, es el responsable de la detección de segmentos vocalizados. Se evaluará el desempeño de este bloque con base en los resultados mostrados en la sección anterior.

Posteriormente, se busca valorar la capacidad del método propuesto para identificar a cada una de las cinco vocales. Para este segundo criterio, se asumirá un resultado perfecto al descartar las tramas que no contienen algún fonema vocálico. Únicamente observaremos el éxito con el cuál se logró clasificar correctamente a las porciones de voz cuyo contenido fonético presentaba alguna vocal.

Por último, se emplearán las evaluaciones previas y las características de las pruebas realizadas para estimar el desempeño del sistema en conjunto.

En cada etapa se compararán los resultados obtenidos al trabajar con señales de voz normal y de voz esofágica, teniendo en consideración la calidad de los datos con los que se contaba al momento de efectuar las pruebas.

6.4.1 Detección de segmentos vocalizados

Como primer criterio para la evaluación del sistema presentado, se observarán únicamente los resultados a la salida del bloque de detección de segmentos vocalizados.

En la figura 6.2.5 se muestra la clasificación de tramas como vocalizadas o no vocalizadas dentro de una grabación de voz normal en la que únicamente se pronuncian vocales. Es posible apreciar que las transiciones de silencio hacia una vocal y de una vocal a silencio fueron erróneamente identificadas como secciones no vocalizadas. Lo anterior no es muy significativo en el caso de la A y la I; es decir, la longitud de las regiones clasificadas erróneamente es pequeña. Sin embargo, en el caso de la O y de la U, la mayor parte de la duración de la vocal fue descartada. A pesar de esto, cada uno de los fonemas posee una región en la cual se detectó su naturaleza vocalizada. Por otra parte, se observa que el algoritmo empleado no tiene problemas para descartar las regiones de silencio ya que ninguna trama de ellas fue considerada vocalizada.

Al observar la figura 6.2.6 podemos ahora comparar el desempeño del bloque cuando éste trabaja con grabaciones de voz esofágica. Al igual que el caso anterior, los datos analizados contienen únicamente vocales y silencio. En esta ocasión es posible apreciar que las transiciones de silencio a vocal fueron identificadas de manera correcta como sonidos vocalizados. Sin embargo, las transiciones de vocal a silencio nuevamente se etiquetaron incorrectamente como formadas por tramas no vocalizadas. A diferencia de la voz normal, la mayor parte de todas las vocales fue identificada como vocalizada. Asimismo, se puede apreciar que nuevamente no existen tramas de silencio clasificadas de manera incorrecta.

Si se comparan los resultados anteriores, se observan las siguientes características significativas acerca del desempeño del bloque de identificación de tramas vocalizadas para grabaciones sin fonemas consonánticos:

Los segmentos de silencio siempre son descartados.

Las regiones cuasi-estacionarias de los fonemas vocálicos siempre son consideradas vocalizadas.

Las transiciones entre una vocal y un bloque de silencio presentan el mayor obstáculo para el algoritmo utilizado.

El desempeño es superior al analizar una grabación de voz esofágica comparado con una grabación de voz normal.

Ahora se analizan los resultados al identificar las regiones vocalizadas en grabaciones que contienen fonemas consonánticos.

Comenzando con las pruebas realizadas para un hablante normal, se observa la salida del bloque de detección de segmentos vocalizados para cuatro palabras. En la figura 6.2.17, correspondiente a la palabra "bala", podemos notar que el fonema /b/ fue clasificado casi en su totalidad como vocalizado. Esto es correcto ya que dicho fonema es una consonante vocalizada; sin embargo, resulta indeseable para este trabajo ya que el clasificador solo busca agrupar el contenido fonético en clases definidas por las vocales. Por otra parte el fonema /l/ fue descartado y ambas apariciones de la vocal A se identificaron como vocalizadas casi en su totalidad. Sin embargo, la transición entre la segunda A y los segmentos de silencio fue descartada. Finalmente, se aprecia como todas las regiones de silencio fueron identificadas apropiadamente.

La siguiente palabra analizada fue "besar". En la figura 6.2.29 se observa nuevamente como el fonema /b/ no fue descartado, a diferencia de los fonemas no vocalizados /s/ y /r/ que si fueron eliminados. Ambas vocales A y E se catalogaron como vocalizadas en su mayor parte; al igual que en ocasiones pasadas el error cometido para la clasificación de las vocales ocurre en la transición hacia otro sonido. Al igual que en casos anteriores, las tramas que no contienen voz fueron descartadas.

La tercera palabra empleada en las pruebas fue "cupo". Analizando la figura 6.2.41 observamos que ambas vocales U y O se identificaron como sonidos vocalizados en su mayor parte. El error nuevamente se dio en la transición final desde cada vocal hacia el fonema /p/ o un a bloque de silencio. Los fonemas consonánticos /k/ y /p/, así como las tramas libres de voz se identificaron correctamente como segmentos no vocalizados.

Por último, se introdujo al sistema una grabación de la palabra "adicto". Los resultados arrojados por el bloque de detección de segmentos vocalizados se muestran en la figura 6.2.53. Al igual que en las pruebas anteriores, todas las vocales presentes en la grabación fueron catalogadas como vocalizadas en su mayor parte. Únicamente las tramas de transición al final de la vocal fueron erróneamente descartadas en los casos de la A y la O. Todas las tramas de la I, por otro lado, se clasificaron correctamente. Del mismo modo los segmentos que contienen los fonemas /d/, /k/ y /t/, así como los de silencio fueron identificados como no vocalizados. Es importante resaltar que la energía de las consonantes obstruentes /k/ y /t/ es similar a la de algunos alófonos vocálicos; pese a esto, el algoritmo implantado logró identificarlas como no vocalizadas.

Ahora observamos el desempeño de esta etapa cuando su entrada consiste en grabaciones de voz esofágica. Nuevamente comenzamos con la palabra "bala". La gráfica de los resultados puede observarse en la figura 6.2.18. Se aprecia un comportamiento similar del algoritmo propuesto, a pesar de la calidad inferior presente en las grabaciones de voz esofágica con las que se llevaron a cabo las pruebas. De nuevo se observa que las tramas etiquetadas como vocalizadas son algunas de las pertenecientes al fonema /b/ y aquellas donde aparece la vocal A en la grabación. El fonema /l/, así como las tramas de silencio fueron nuevamente descartadas por el algoritmo. A diferencia de lo ocurrido con las grabaciones de voz normal, ahora un fragmento pequeño de la segunda A fue catalogado como no vocalizado. Pese a lo anterior, los resultados con voz esofágica son ligeramente mejores a los obtenidos con voz normal, ya que las transiciones al inicio y final de las vocales no fueron descartadas.

En la figura 6.2.30 se muestran los resultados para la grabación de voz esofágica que contiene la palabra "besar". Se observa que el fonema /b/ es nuevamente incluido en los resultados. Lo mismo ocurre con las vocales, sin embargo las tramas finales de la E fueron descartadas. Ambos fonemas consonánticos /s/ y /r/, así como los segmentos carentes de voz se marcaron como no vocalizados.

Al analizar la palabra "cupo" se obtienen los resultados de la figura 6.2.42. En ella se aprecia como el algoritmo descartó los bloques de silencio así como los fonemas /k/ y /p/ en su totalidad. Ambas vocales fueron clasificadas como vocalizadas en sus regiones cuasi-estacionarias, sin embargo las tramas finales de la U y ambas transiciones alrededor de la O se identificaron erróneamente como no vocalizadas.

Finalmente se observan los resultados para la palabra "adicto" en voz esofágica. Estos se encuentran en la figura 6.2.54. Es posible apreciar como los fonemas vocálicos fueron

correctamente identificados, a excepción de sus tramas de transición hacia otro fonema o silencio. Esto no fue muy significativo en el caso de la A y la I, sin embargo fue eliminado casi un tercio del fonema /o/. Todos los fonemas consonánticos y tramas libres de voz se clasificaron como no vocalizados; inclusive aquellas tramas que contenían a los fonemas /k/ y /t/.

Los resultados expuestos confirman que el algoritmo utilizado para la detección de segmentos vocalizados tiene un desempeño similar para la voz de un hablante normal y la voz esofágica. Las pruebas realizadas inclusive muestran un desempeño ligeramente superior en este último caso.

Se observa además que el método empleado no presenta problemas para la identificación de regiones de silencio. Esto se cumple aún en grabaciones donde se presenta una cantidad considerable de ruido como las utilizadas en las pruebas de voz esofágica.

Por otra parte, se detectó que el sistema presenta problemas para descartar las consonantes vocalizadas como la B. Esto ocurre tanto en hablantes normales como en pacientes laringectomizados y es evidente en las pruebas con las palabras "bala" y "besar".

A diferencia de lo anterior, se confirmó que el bloque de detección de segmentos vocalizados es capaz de descartar las consonantes no vocalizadas del español. El desempeño es satisfactorio aún si éstas se presentan como alófonos de una energía comparable a la de las vocales. En las pruebas realizadas tanto con voz normal como con voz esofágica no se detectó ningún segmento que contuviera una consonante no vocalizada y que fuese clasificado erróneamente.

El desempeño es también satisfactorio al clasificar las componentes cuasi-estacionarias de los fonemas vocálicos. En las pruebas efectuadas, sólo se detectó un error en este aspecto para la palabra "bala" articulada por un hablante esofágico.

Finalmente, uno de los problemas recurrentes en todas las pruebas realizadas fue la incorrecta clasificación de las primeras y últimas tramas de los fonemas vocálicos. Lo anterior no tiene un impacto significativo en el desempeño global del sistema. Esto se debe a que basta con identificar las regiones cuasi-estacionarias de las vocales para que etapas posteriores tengan la información necesaria para reconocer los fonemas vocálicos presentes en la señal de entrada.

Podemos entonces concluir que la etapa de detección de segmentos vocalizados tiene un desempeño satisfactorio para la detección de vocales y la eliminación de tramas libres de voz o aquellas que contengan consonantes no vocalizadas. Sin embargo, es incapaz de distinguir entre las consonantes vocalizadas y las vocales. Esto último podría, en algunas circunstancias, ocasionar problemas en etapas posteriores del sistema. La razón principal de ello es que el clasificador limita la agrupación de cada segmento a cinco clases definidas exclusivamente por las vocales.

6.4.2 Reconocimiento de fonemas vocálicos

Se busca ahora evaluar el desempeño del sistema presentado en este proyecto para la identificación de los fonemas vocálicos como su vocal asociada. Lo anterior estará dado por la eficacia de los bloques de extracción de características, creación de vectores de parámetros, clasificación y filtrado de resultados. Se asume en esta sección que la detección de segmentos vocalizados eliminó todas las tramas que no contienen algún alófono vocálico.

Uno de los principales criterios para evaluar la extracción de características y creación de vectores de parámetros es la semejanza entre los datos obtenidos para el mismo fonema en circunstancias distintas. Para este trabajo, resulta particularmente importante conocer las variaciones que se tienen al analizar señales de voz normal y esofágica.

La sección anterior presentó gráficas de la transformada wavelet para las grabaciones de un hablante normal y de voz esofágica, así como los vectores de parámetros obtenidos a partir de ésta. Es posible apreciar como la voz esofágica en general presenta coeficientes más pequeños que la voz normal (es decir, posee una menor cantidad de energía). Además, en el caso de las vocales aisladas "A E I O U", las componentes de frecuencias más altas presentes en la voz de un hablante común aparecen con una atenuación significativa para un paciente laringectomizado.

Pese a las diferencias anteriores, existen similitudes muy importantes entre las descomposiciones de la voz normal y la esofágica. En todos los ejemplos mostrados se presenta una distribución de la energía similar para la misma palabra articulada por el individuo H23 y el individuo HL30.

Al comparar las gráficas para los vectores de parámetros, se pueden distinguir patrones similares para ambos individuos. La diferencia más clara entre la voz de un hablante ordinario y la esofágica, además de la intensidad, se puede apreciar en la componente de energía total de la señal (la de mayor amplitud). Mientras que la voz normal presenta cambios suaves en este parámetro, la voz esofágica se caracteriza por incrementos y decrementos repentinos en su energía. Lo anterior no impide el reconocimiento de la misma. Si se observa con detenimiento, cuando la energía total de la señal sufre una alteración, las componentes de energía contenida en las bandas críticas varían de un modo similar. Este comportamiento es el esperado ya que la energía total debe ser igual a la suma ponderada de la energía en cada subbanda. Los factores involucrados en dicha operación dependen exclusivamente de la base de funciones utilizada para computar la transformada wavelet. Ya que este parámetro no varía, un cambio en la energía total de la señal está asociado con cambios en la energía de cada descomposición.

La mejor manera para evaluar la capacidad del sistema en el reconocimiento de fonemas vocálicos, es observando directamente los resultados que entrega para cada fragmento de

una señal que contenga cierto alófono de alguna vocal. Esto puede apreciarse en las gráficas de componentes vocálicos presentadas en la sección anterior.

Se observa que, tanto en la voz del individuo H23 como en la del sujeto HL30, todas las secciones vocálicas fueron clasificadas correctamente como la vocal que contenían. En el caso de las tramas pertenecientes al fonema /b/, estas fueron etiquetadas como parte de la vocal que les sucedía. Este error no es considerado un problema de la etapa de reconocimiento, ya que la red neuronal artificial utilizada como clasificador fue entrenada para agrupar a cada segmento en clases generadas exclusivamente por las vocales.

El desempeño del sistema para las pruebas mencionadas es ideal. Sin embargo, al tratarse de sólo cinco palabras, éstas no son suficientes para determinar con una certeza adecuada el porcentaje de éxito en el reconocimiento. Pruebas posteriores con conjuntos de palabras más extensos arrojaron los resultados visibles en las figuras 6.3.1 y 6.3.2.

Para la voz del individuo H23, un hablante normal, se observó un porcentaje de éxito mayor al 90% en la identificación de fonemas vocálicos. Las pruebas efectuadas no mostraron ningún error al identificar el fonema /u/. Por otro lado la vocal I presentó el porcentaje de reconocimiento más bajo de todos los fonemas vocálicos, siendo éste del 80%.

Los resultados anteriores son satisfactorios, tomando en cuenta que se realiza el reconocimiento fonema a fonema. Esto significa que no se dispone de un contexto y un modelo del lenguaje o diccionario para la corrección de errores.

La tendencia observada con base en las grabaciones del individuo HL30, un hablante esofágico, mostró un porcentaje de éxito en el reconocimiento similar al obtenido en las pruebas del individuo H23 para los fonemas /a/, /e/ y /o/. En estos tres casos se reconocieron más del 80% de las apariciones de su vocal respectiva en las grabaciones examinadas. Los alófonos de la I y de la U, por otro lado, arrojaron un porcentaje de éxito menor al 40% para la primera y del 50% para la segunda.

El bajo rendimiento obtenido para la identificación las últimas dos vocales puede ser atribuido principalmente a los datos de entrenamiento y a la mala calidad en las grabaciones de voz esofágica disponibles. Por otro lado, los resultados obtenidos para el reconocimiento de las vocales A, E y O, a pesar de las condiciones de ruido y de emplear como entrada al sistema voz esofágica, indican que el algoritmo de reconocimiento propuesto es adecuado para los fines de este proyecto.

6.4.3 Desempeño global del sistema

Tomando en consideración los puntos tratados anteriormente, se observa que el sistema diseñado presenta un desempeño satisfactorio. En particular las etapas para el reconocimiento de fonemas vocálicos muestran un porcentaje de éxito notable si se considera lo limitado de los datos de entrenamiento. Este desempeño mantuvo su calidad al

pasar de voz de un hablante normal a voz esofágica cuando se tuvieron suficientes datos para llevar a cabo el adiestramiento adecuado del clasificador.

La convergencia del error cuadrático medio durante el entrenamiento de la red neuronal artificial es igualmente aceptable, así como el tiempo que demora este proceso. La flexibilidad con la que fue implantado el sistema permite a su vez modificar sus parámetros más importantes con la finalidad de optimizar su desempeño para alguna situación específica.

Por otra parte, la detección de segmentos vocalizados puede ajustarse al locutor variando los umbrales para los criterios de energía y cruces por cero. Sin embargo, esta sección es la que introduce la mayoría de los errores observados durante las pruebas efectuadas. Su principal deficiencia se encuentra al analizar tramas que contienen fonemas consonánticos vocalizados. Dado que el clasificador se ve obligado a agrupar a dichas tramas como una vocal, esto introduce símbolos incorrectos en el resultado.

Capítulo VII

Documentación de la interfaz gráfica

7.1 Introducción

El sistema consta de dos interfaces gráficas, una interfaz gráfica que facilita el uso de éste y otra que permite la sintonización y configuración de algunos parámetros variables.

La primera se invoca por medio del archivo "interfaz.fig".

Esta interfaz gráfica se divide en 3 secciones principales:

- Sección de reconocimiento
- Sección de entrenamiento
- Sección de grabadora de voz

Estas pueden verse en la siguiente figura:

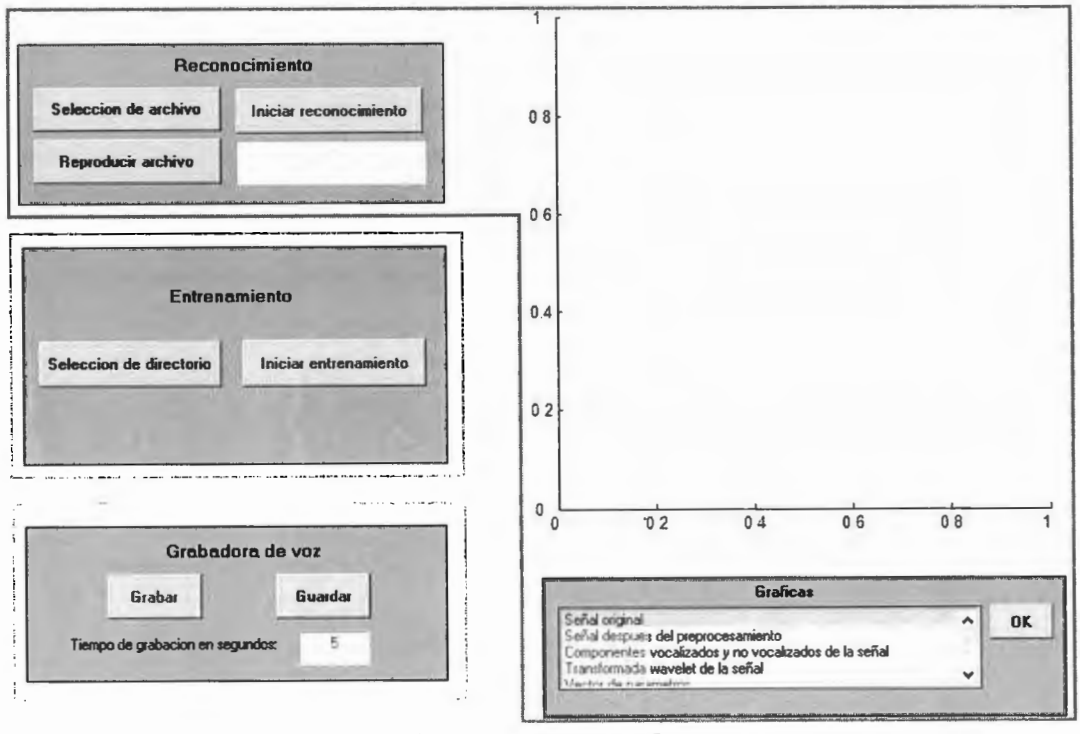


Figura 7.1.1. Interfaz gráfica

A la segunda se accede por medio del archivo "configuración.fig".

Esta interfaz se divide en 5 secciones principales:

- Número de nodos en las capas ocultas de la red neuronal artificial
- Iteraciones máximas de la red neuronal artificial
- Umbrales de energía
- Umbrales de cruces por cero
- Umbrales de certeza

Estas secciones pueden verse en la figura 7.1.2

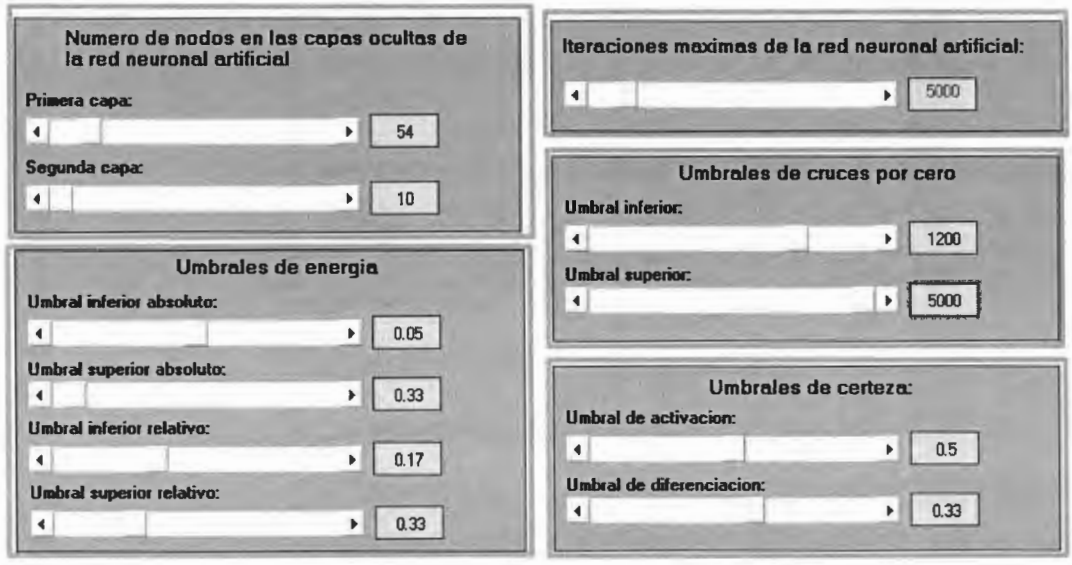


Figura 7.1.2. Interfaz gráfica de configuración

Sección de reconocimiento:

Esta sección permite elegir, reproducir, reconocer y graficar las diversas etapas a las que está sometido un archivo en particular.

El formato del archivo debe de ser "WAV"; sin embargo, no existe restricción en relación a la frecuencia de muestreo, bits por muestra o duración que éste tenga.

Las gráficas que pueden visualizarse son las siguientes:

- Señal original
- Señal después del preprocesamiento
- Componentes vocalizados y no vocalizados de la señal
- Transformada wavelet de la señal
- Vector de parámetros
- Componentes vocálicos de la señal

Sección de entrenamiento:

Esta sección fue creada para permitir al usuario efectuar el entrenamiento de la red neuronal de una forma simple y sencilla.

El interesado tiene la opción de elegir el directorio donde se encuentran los archivos de audio para el entrenamiento y posteriormente puede iniciar éste.

Sección de grabadora de voz.

En esta sección se le brinda al usuario la oportunidad de grabar y guardar sus propios archivos de voz en formato WAV con frecuencia de muestreo de 8 kHz y 16 bits por muestra. El usuario debe ingresar el tiempo de grabación en segundos.

Sección de número de nodos en las capas ocultas de la red neuronal artificial.

Esta sección le permite al usuario elegir el número de capas ocultas utilizadas en el diseño de la red neuronal artificial.

Sección de iteraciones máximas de la red neuronal artificial.

En esta sección, el usuario tiene la opción de cambiar las iteraciones máximas que hará la red neuronal artificial antes de terminar el entrenamiento.

Sección de umbrales de energía.

Esta sección le da al usuario la oportunidad de elegir los umbrales de energía que mejor se adapten a su voz.

Sección de umbrales de cruces por cero.

La sección de umbrales de cruces por cero fue creada con la intención de darle al usuario la opción de adaptar el número de cruces por cero que mejor se adapte a su voz.

Sección de umbrales de certeza.

Esta sección le brinda al usuario la oportunidad de elegir los umbrales de certeza con la que se hace la validación de las salidas de la red neuronal artificial.

7.2 Instrucciones de uso

7.2.1 Reconocimiento

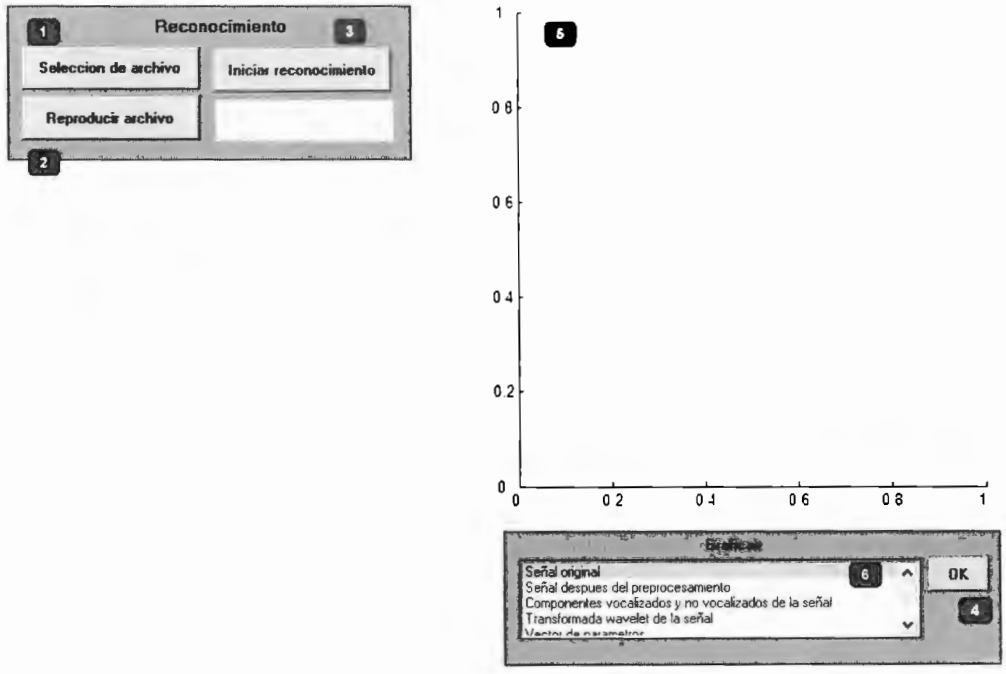


Figura 7.2.1.1

Esta sección consta de 4 botones: Selección de archivo (1), Reproducir archivo (2), Iniciar reconocimiento (3) y OK (4). Además, se cuenta con unos ejes (5) y una lista de selección (6).

Para elegir el archivo de voz que se desea reconocer, se debe presionar el botón de "Selección de archivo" como puede verse en la figura 7.2.1.2. Posteriormente se selecciona éste en la nueva ventana y se presiona "aceptar".

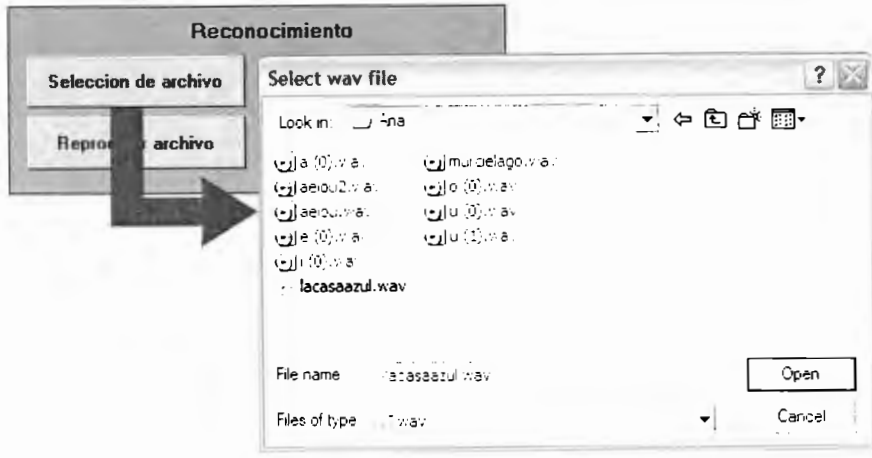


Figura 7.2.1.2

Una vez que se tiene el archivo de voz a reconocer, el usuario tiene la opción de reproducir el archivo o iniciar el reconocimiento.

Si desea reproducir el archivo, entonces se debe presionar el botón de "Reproducir archivo" y en seguida se escuchará el contenido de éste.

Si desea iniciar el reconocimiento, se debe presionar el botón de "Iniciar reconocimiento". Cuando el reconocimiento haya terminado, el cuadro de texto mostrará las vocales que fueron reconocidas en el archivo.

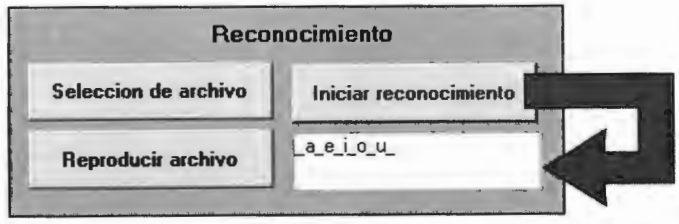


Figura 7.2.1.3

Una vez realizado el reconocimiento del archivo de audio, se puede graficar la señal en sus diferentes etapas.

Esto se logra seleccionando la gráfica en la lista y posteriormente presionando "OK"

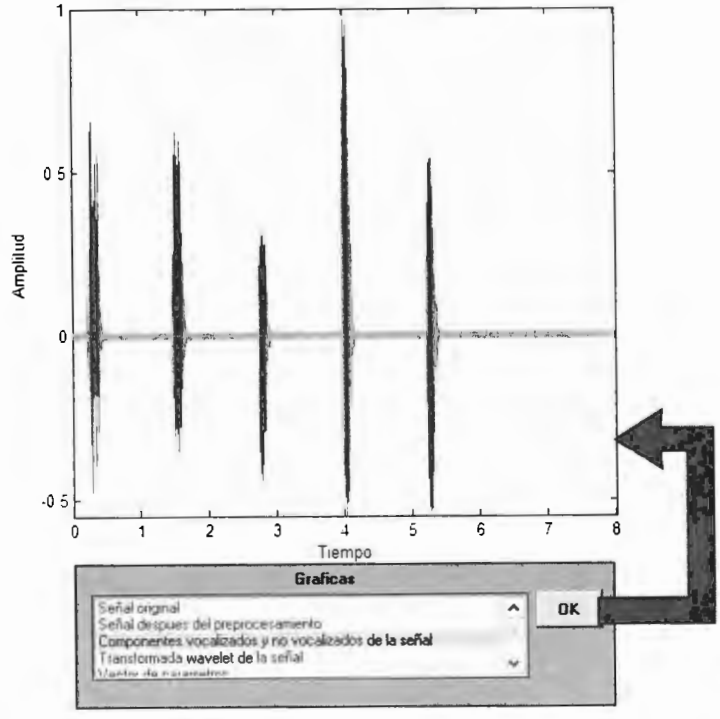


Figura 7.2.1.4

7.2.2 Entrenamiento:

Figura 7.2.2.1

Esta sección consta de 2 botones: Selección de directorio (1) e Iniciar entrenamiento (2).

Para elegir el directorio donde se guardan los archivos de audio con los que se desea realizar el entrenamiento, se presiona el botón "Selección de directorio" y posteriormente se elige el directorio en la nueva ventana presionando "Aceptar".

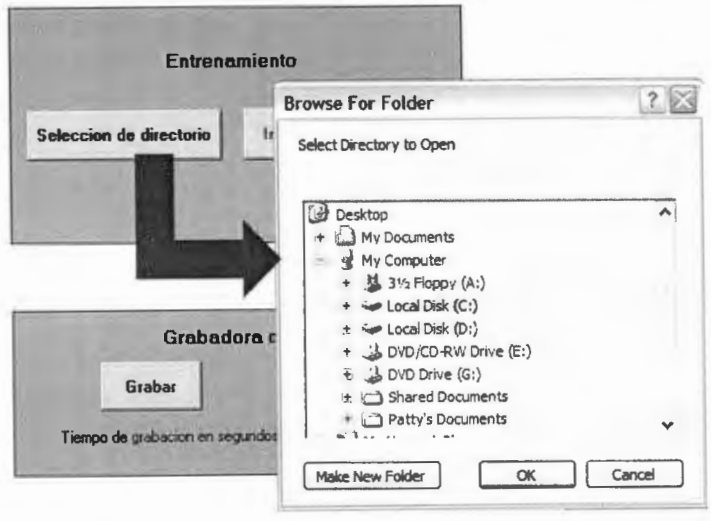


Figura 7.2.2.2

Para iniciar el reconocimiento, sólo se debe presionar el botón de "Iniciar el reconocimiento" y posteriormente mirar el progreso en el cuadro de texto.

7.2.3 Grabadora de voz

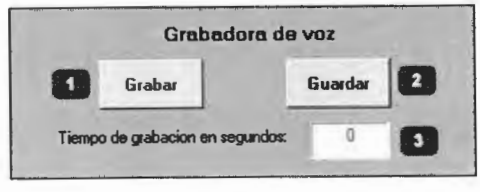


Figura 7.2.3.1

Esta sección consta de 2 botones: Grabar (1), Guardar (2). También cuenta con un cuadro de texto (3) donde se introduce el tiempo de grabación en segundos.

Para grabar un archivo de voz, primero se debe especificar el tiempo de grabación en segundos que se desea que dure ésta. Posteriormente se presiona el botón de "Grabar" e inmediatamente iniciará la grabación. Cuando el tiempo haya transcurrido, se debe presionar el botón "Guardar" para guardar el archivo de audio.

Figura 7.2.3.2

7.2.4 Interfaz de configuración

La interfaz de configuración permite editar cualquier sección por medio de la barra de desplazamiento. En los cuadros de texto se muestran los valores por defecto para cada uno de los criterios.

Conclusiones

El presente trabajo propone un algoritmo, diseñado con base en los rasgos más generales del oído humano, para el reconocimiento de los fonemas vocálicos del español en una señal que contiene voz esofágica. Su desarrollo involucró la unificación de ideas provenientes de distintas ramas del conocimiento como la anatomía, la fonética, la psicoacústica, el análisis funcional y los métodos para la clasificación de patrones. El sistema presentado emplea técnicas con fundamentos sólidos y ampliamente utilizadas en diversas aplicaciones como la transformada wavelet y las redes neuronales artificiales. Se introdujeron modificaciones convenientes a los procedimientos usuales y se demostró la validez de las mismas. Finalmente se implantó el algoritmo bajo el ambiente de Matlab para evaluar su desempeño al operar sobre datos reales.

Los resultados obtenidos muestran que la descomposición de una señal de voz, basada en el modelado de la membrana basal, presenta de manera directa características útiles para la identificación de fonemas. Al utilizar una red neuronal artificial como clasificador, se logró el reconocimiento de las vocales en hablantes normales con un porcentaje de éxito mayor al 90%.

Al analizar voz esofágica, el desempeño fue similar para las vocales A, E y O. Sin embargo, sólo se alcanzó un porcentaje de reconocimiento del 50% para los alófonos de la I y de la U. Asimismo, se incrementó notablemente el número de tramas no vocálicas clasificadas como una vocal. Ambos problemas tienen su origen más probable en la calidad de las grabaciones de voz esofágica con las que se contó. Éstas fueron insuficientes para realizar un entrenamiento adecuado de la red neuronal en el caso de los fonemas /i/ y /u/. Por otra parte, la mayoría de las grabaciones contenían componentes de ruido con una potencia significativa; en algunos casos incluso voces de fondo y otros sonidos que podrían ocasionar que tramas de silencio fueran erróneamente etiquetadas como vocalizadas.

En un contexto libre de ruido, sin embargo, también existen situaciones en las que tramas no vocálicas equivocadamente se identifican como vocales. Esto ocurre debido a la presencia en algunas palabras de fonemas consonánticos vocalizados como /b/, /d/ o /g/. El clasificador utilizado se ve obligado a asignar las tramas que contienen este tipo de información a una de las cinco clases definidas como cada una de las vocales.

La problemática anterior no es fácil de superar ya que los fonemas consonánticos vocalizados presentan un comportamiento muy similar al de las vocales. Este proyecto no busca ofrecer una alternativa para separar las vocales de las consonantes en una etapa previa al reconocimiento. En su lugar, se diseñó una extracción de características que

permita incrementar el número de símbolos fonéticos a identificar sin reducir el porcentaje de éxito.

Al emplear una técnica de análisis en resoluciones múltiples, el modelado de la señal efectuado por el sistema propuesto captura las componentes de alta frecuencia presentes en las transiciones entre fonemas consonánticos. Esto le permitiría a un clasificador adecuado reconocer estructuras de mayor complejidad que los fonemas, por ejemplo sílabas. Con ello se resolvería el problema de las consonantes vocalizadas de un modo más sencillo a que si se tratara de eliminarlas en una de las etapas iniciales.

Trabajo futuro

Una de las actividades a corto plazo, a fin de continuar con este proyecto, sería la obtención de una base de datos de voz esofágica. Este es un requisito para conocer con mayor precisión el desempeño del sistema propuesto. Es necesario contar con grabaciones, hasta donde sea posible, libres de ruido. Además deben contener muestras representativas de las estructuras fonológicas que se busque clasificar (ya sean fonemas o sílabas).

A largo plazo, se sugiere un mejor aprovechamiento de las técnicas de análisis en resoluciones múltiples desarrolladas durante este trabajo. Específicamente su uso para el reconocimiento de cuerpos fonológicos que otros procedimientos no son capaces de caracterizar como los fonemas /b/, /d/ y /s/. Al estar fundamentada en un modelo de la membrana basal y emplear un muestreo del plano escala-traslación consistente con una de las características psicoacústicas de mayor importancia, la discretización de la transformada wavelet continua expuesta ofrece información útil para la clasificación de señales cuyo receptor natural sea el ser humano. Al mismo tiempo, las variaciones de la escala intrínsecas de la transformada wavelet le permiten a la extracción de características utilizada operar sobre señales no estacionarias, a diferencia de otros métodos como los coeficientes de predicción lineal.

Desigualdad de Heisenberg

Sea $f \in L^2(\mathbf{R}^n)$, para cualquier $j \in \{1,2,\ldots,n\}$ se cumple que:

$$\int_{-\infty}^{\infty} x_j^2 |f(x)|^2 dx \cdot \int_{-\infty}^{\infty} y_j^2 |\hat{f}(y)|^2 dy \geq \frac{1}{4}\left(\int_{-\infty}^{\infty} |f(x)|^2 dx \right)^2$$

De modo que, en el caso de una señal $x(t)$ de energía $E < \infty$, definimos $X(j\omega)$ como la transformada de Fourier de $x(t)$, y mediante la identidad de Parseval tenemos que:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(j\omega)|^2 d\omega$$

A continuación definimos el centro temporal t_c de la señal a partir del primer momento alrededor del origen de la energía que ésta transporta en cada instante:

$$t_c = \frac{1}{E} \int_{-\infty}^{\infty} t |x(t)|^2 dt$$

Análogamente, definimos el centro espectral ω_c de la señal a partir del primer momento alrededor del origen de su energía para cada frecuencia:

$$\omega_c = \frac{1}{2\pi \cdot E} \int_{-\infty}^{\infty} \omega |X(j\omega)|^2 d\omega$$

A partir de los resultados anteriores, definimos la dispersión temporal Δt para la medición de la señal mediante su segundo momento alrededor del centro temporal t_c (i.e. la desviación estándar de su representación temporal):

$$\Delta t = \frac{1}{E} \int_{-\infty}^{\infty} (t - t_c)^2 |x(t)|^2 dt$$

Del mismo modo, la dispersión espectral $\Delta\omega$ de la transformada de Fourier de la señal es definida con base en su segundo momento alrededor del centro espectral ω_c (i.e. la desviación estándar de su representación espectral):

$$\Delta\omega = \frac{1}{2\pi \cdot E} \int_{-\infty}^{\infty} (\omega - \omega_c)^2 |X(j\omega)|^2 d\omega$$

Empleando las definiciones anteriores y la desigualdad de Heisenberg obtenemos el límite inferior para el producto de las dispersiones temporales y espectrales de una señal:

$$\Delta t \cdot \Delta\omega \geq \frac{1}{2}$$

Teorema de Littlewood-Paley

Sea $f \in L^p$, podemos escribir su expansión en series de Fourier como:

$$f(x) = \lim_{N \to \infty} \sum_{n=-N}^{N} \hat{f}(n) \cdot e^{2\pi \cdot i \cdot n \cdot x}$$

Agrupando los términos de la sumatoria en bloques de crecimiento exponencial $\Delta_j f$ tenemos:

$$f(x) = \hat{f}(0) + \lim_{J \to \infty} \sum_{j=1}^{J} (\Delta_j f)(x)$$

donde, si se emplean bloques diádicos se tiene la siguiente definición:

$$(\Delta_j f)(x) = \sum_{2^{j-1} \le |n| \le 2^j} \hat{f}(n) \cdot e^{2\pi \cdot i \cdot n \cdot x}$$

Es entonces posible caracterizar el comportamiento local de f a partir de los bloques $\Delta_j f$. Específicamente, el teorema de Littlewood-Paley dice que, si para algún $\varepsilon > 0$ se cumple

$$\sup_x |(\Delta_j f)(x)| \le C \cdot 2^{-j\varepsilon}$$

entonces f es continua.

signalParams.m

```
%signalParams
%
%Escribe los archivos:
%   frameDuration.txt       - Intervalo de tiempo en segundos que separa las
tramas
%   windowDuration.txt      - Duracion en segundos de la ventana a utilizar
%   method.txt              - Metodo para la extraccion de caracteristicas
%   supportThreshold.txt    - Umbral para el soporte numerico del wavelet


function signalParams;
    %Tiempo entre tramas de 15ms
    frameDuration=0.015;
    %Duracion de la ventana de 30ms
    windowDuration=0.03;
    %Metodo a utilizar para la extraccion de caracteristicas:
    %ear - Wavelet obtenido a partir del modelo del oido
    %filters - Banco de filtros de ancho de banda critico
    %daub4 - Wavelet de Daubechies de 4 coeficientes
    %haar - Wavelet de Haar
    method='ear';
    %Umbral para el soporte numerico del wavelet obtenido a partir del
    %modelo del oido
    supportThreshold=0.01;

    %Almacena los parametros en archivos de datos
    save data/frameDuration.txt frameDuration -ascii;
    save data/windowDuration.txt windowDuration -ascii;
    save data/method.txt method -ascii;
    save data/supportThreshold.txt supportThreshold -ascii;
return;
```

initThresholds.m

```
%initThresholds
%
%Escribe los archivos:
%   umbrales.txt    - Parametros para el acondicionamiento y clasificacion


function initThresholds;
    %Inicializa la variable que almacenara los parametros
    th=zeros(11,1);


    %Numero de nodos en la primer capa oculta
    th(1)=54;
    %Numero de nodos en la segunda capa oculta
    th(2)=10;
    %Numero maximo de iteraciones para el entrenamiento
    th(3)=5000;
    %Umbral de energia inferior absoluto
    th(4)=0.05;
    %Umbral de energia superior absoluto
    th(5)=0.33;
    %Umbral de energia inferior relativo
    th(6)=0.17;
    %Umbral de energia superior relativo
    th(7)=0.33;
    %Umbral inferior de cruces por cero
    th(8)=1200;
    %Umbral superior de cruces por cero
    th(9)=5000;
    %Umbral de activacion
    th(10)=0.5;
    %Umbral de diferenciacion
    th(11)=0.33;
```

```
    %Guarda los parametros en un archivo de datos
    save data/umbrales.txt th -ascii;
return;
```

getVowels.m

```
%word=getVowels(file,play)
%
%Recibe:
%   file    - Ruta al archivo que se desea analizar
%   play    - Bandera que indica si se desea reproducir el archivo
%
%Regresa:
%   word    - Cadena de caracteres filtrados representando las vocales en
%             la grabacion analizada

function word=getVowels(file,play);
    %Inicializa parametros omitidos
    if(nargin<2)
        play=0;
    end;

    %Se empleara un filtrado modal con una ventana de 10 simbolos
    filterSize=10;

    %Obtiene el contenido vocalico del archivo
    vowels=detectVowels(file,play);

    %Elimina tramas erroneamente clasificadas como no vocalizadas
    vowels=lowPass(vowels,'_',filterSize-4);

    %Busca solo las componentes vocalizadas
    vowelsIndex=find(vowels~='_');
```

```
    if(isempty(vowelsIndex))
        %Si no existen, termina
        word='_';
        return;
    end;

    %Busca los limites entre grupos de vocales consecutivas
    vowelsLimits=[0 find(vowelsIndex(1:end-1)~=vowelsIndex(2:end)-1)
length(vowelsIndex)];

    %Determina la longitud del grupo de vocales mas largo
    maxLength=max(vowelsLimits(2:end)-vowelsLimits(1:end-1));

    %Genera una matriz con los grupos de vocales ordenados por filas
    prevLimit=vowelsLimits(1);
    vowelsGroups=[];
    for(i=2:length(vowelsLimits))

currentGroup=vowels(vowelsIndex(prevLimit+1):vowelsIndex(vowelsLimits(i)));
        prevLimit=vowelsLimits(i);
        %Si el grupo es menor a la longitud maxima, agrega simbolos 'x' al final
        vowelsGroups=[vowelsGroups;[currentGroup repmat('x',1,maxLength-
length(currentGroup))]];
    end;

    %Realiza un filtrado modal de cada grupo de vocales
    vowels=[];
    for(i=1:length(vowelsGroups(:,1)))
        %Se eliminan los simbolos que pudieron ser agregados
        currentS=vowelsGroups(i,find(vowelsGroups(i,:)~='x'));
        filteredS=[];
        %Si el grupo es menor al tamaño del filtro, se obtiene su moda
        if(length(currentS)<=filterSize)
            filteredS=findMode(currentS);
```

```
            if(length(filteredS)>1)
                filteredS=filteredS(1);
            end;
        else
            %De lo contrario, se obtiene la moda de cada grupo de filterSize
simbolos
            for(i=ceil(filterSize/2):length(currentS)-floor(filterSize/2))
                currentVowel=findMode(currentS(i-floor((filterSize-
1)/2):i+ceil((filterSize-1)/2)));
                if(length(currentVowel)>1)
                    %Si mas de un simbolo comparten el numero maximo de
apariciones se toma el mas alejado de los bordes
                    if(i<length(currentS)/2)
                        currentVowel=currentVowel(end);
                    else
                        currentVowel=currentVowel(1);
                    end;
                end;
                filteredS=[filteredS currentVowel];
            end;
        end;
        %El resultado se agrega a la cadena resultante
        vowels=[vowels '_' filteredS '_'];
    end;

    %Se eliminan los simbolos identicos a su predecesor
    word=vowels(1);
    for(i=2:length(vowels))
        if(vowels(i)~=word(end))
            word=[word vowels(i)];
        end;
    end;
return;
```

```
%dataMode=findMode(data)
%Obtiene la moda del conjunto data
function dataMode=findMode(data);
    %Agrupa simbolos iguales
    sortedData=[];
    for(i=1:length(data))
        if(isempty(find(sortedData==data(i))))
            %Si es la primera aparicion del simbolo, agrega su clase
            sortedData=[sortedData;data(i) 1];
        else
            %De lo contrario, incrementa la cuenta del simbolo
            [x,y]=find(sortedData==data(i));
            sortedData(x,2)=sortedData(x,2)+1;
        end;
    end;
    %Busca el (los) simbolo(s) con un mayor numero de apariciones
    dataMode=sortedData(find(sortedData(:,2)==max(sortedData(:,2))),1);
return;


%s=lowPass(data,symbol,windowSize)
%Filtrado modal de las apariciones de symbol dentro del vector data
function s=lowPass(data,symbol,windowSize);
    s=data;
    %Busca las apariciones de symbol
    sIndex=find(data==symbol);
    for(i=1:length(sIndex))
        %Filtrado modal con una ventana de windowSize elementos
        if(length(find(data(max(1,sIndex(i)-floor((windowSize-
1)/2)):min(length(data),sIndex(i)+ceil((windowSize-
1)/2)))~=symbol))>=ceil(windowSize/2))
            s(sIndex(i))=0;
        end;
    end;
```

```
    %Elimina de la cadena final los simbolos indeseados
    s=s(find(s~=0));
return;
```

detectVowels.m

```
%vowels=detectVowels(file,play)
%
%Recibe:
%   file - Nombre del archivo en formato WAV que se desea analizar
%   play - Bandera que indica si se desea (1) o no (0) reproducir el sonido
contenido en
%          el archivo file
%
%Utiliz los archivos:
%   frameDuration.txt       - Duracion en segundos de cada trama
%   windowDuration.txt      - Longitud de la ventana en segundos
%   method.txt              - Metodo a utilizar para la extraccion de
caracteristicas
%   supportThreshold.txt    - Umbral para el soporte numerico de la transformada
wavelet
%   nInputs.txt             - Numero de caracteristicas a utilizar en el
reconocimiento
%
%Regesa:
%   vowels - Cadena de caracteres de longitud floor((length(data)-
windowSamples)/frameSamples)+1
%            donde length(data) es el numero de muestras contenidas en el
%            archivo file; windowSamples es el numero de muestras empleadas
%            para el analisis de cada segmento; y frameSamples es el numero
%            de muestras entre un segmento a clasificar y el proximo.
%            Cada caracter de la cadena vowels puede tomar 6 valores
%            "_","a","e","i","o" o "u" que identifican a su respectivo
%            segmento como "no vocalizado/silencio" o como un segmento en
%            el que predomina el fonema señalado
```

```
function vowels=detectVowels(file,play);

    %Inicializa parametros omitidos

    if(nargin<2)

        play=0;

    end;


    %Si existen datos previos, pregunta si se desean eliminar

    if(exist('data/frameDuration.txt','file') |
exist('data/windowDuration.txt','file') | exist('data/method.txt','file') |
exist('data/supportThreshold.txt','file') | exist('data/umbrales.txt','file'))

        confirm=input(sprintf('Borrar los datos anteriores ''s''/''n'' [''n''] >
'));

        if(~isempty(confirm) & strcmp(confirm,'s'))

            cd data;

            delete *.txt;

            cd ..;

        end;

    end;


    %Si no existen los archivos de configuracion, los genera con valores por
defecto

    if(~exist('data/frameDuration.txt','file') |
~exist('data/windowDuration.txt','file') | ~exist('data/method.txt','file') |
~exist('data/supportThreshold.txt','file'))

        signalParams;

    end;

    if(~exist('data/umbrales.txt','file'))

        initThresholds;

    end;


    %Si la red neuronal no ha sido entrenada, llama a trainSpeech

    if(~(exist('data/W1.txt','file') & exist('data/W2.txt','file') &
exist('data/W3.txt','file')...

            & exist('data/B1.txt','file') & exist('data/B2.txt','file') &
exist('data/B3.txt','file') & exist('data/nInputs.txt','file')))

        trainSpeech('audio/trainingData');
```

```
    end;

    %Obtiene los parametros para la segmentacion
    frameDuration=load('data/frameDuration.txt');
    windowDuration=load('data/windowDuration.txt');
    %Obtiene los parametros para la extraccion de caracteristicas
    method=char(load('data/method.txt'));
    supportThreshold=load('data/supportThreshold.txt');
    %Obtiene los parametros para el reconocimiento
    nInputs=load('data/nInputs.txt');
    thresholds=load('data/umbrales.txt');
    activationTH=thresholds(10);
    differentiationTH=thresholds(11);

    %Lee las muestras de la señal desde el archivo de audio
    [data,samplingFrequency]=wavRead(file);
    %Y lo reproduce si se solicita
    if(play)
        sound(data,samplingFrequency);
    end;

    %Acondiciona la señal e identifica sus tramas vocalizadas
    [data,bVoiced]=isVoiced(data,samplingFrequency);
    %Genera un eje de tiempo para las componentes vocalizadas
    tVoiced=1:length(data);
    for(i=1:length(bVoiced))
        if(~bVoiced(i))
            tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
        end;
    end;
```

```
    %Inicializa las variables para la generacion de la cadena resultante
    vowels=repmat('_',size(bVoiced));
    code=['a' 'e' 'i' 'o' 'u'];

    %Obtiene vectores de parametros para cada trama de la señal

featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,0,tVoiced);

    for(i=1:length(bVoiced))
        if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))
            %Solo se busca un fonema para las tramas vocalizadas
            pattern=[(1:length(code))' forward(featureVector(:,i))];
            %Se buscan los dos valores mas grandes
            pattern=flipud(sortrows(pattern,2));
            if(~(pattern(1,2)<activationTH | pattern(2,2)>differentiationTH))
                %Si el mayor excede el umbral de activacion y el siguiente no
excede el umbral de diferenciacion,
                %se agrega el simbolo correspondiente al fonema
                vowels(i)=code(pattern(1,1));
            end;
        end;
    end;
return;
```

isVoiced.m

```
%[data,bVoiced]=isVoiced(data,samplingFrequency)
%
%Recibe:
%   data                - Vector formado por las muestras de la señal de audio
%   samplingFrequency   - Frecuencia de muestreo para la señal de audio
%
%Utiliza los archivos:
%   frameDuration.txt   - Duracion en segundos de cada trama
```

```
%   windowDuration.txt  - Longitud de la ventana en segundos
%   umbrales.txt        - Umbrales para los criterios de energia y de cruces por
cero
%
%Regresa:
%   data    - Vector que contiene las muestras de la señal luego del
%             acondicionamiento
%   bVoiced - Vector de longitud
%             floor((length(data)-windowSamples)/frameSamples)+1 que indica
%             si su trama correspondiente es vocalizada (1) o no (0)

function [data,bVoiced]=isVoiced(data,samplingFrequency);
    %Obtiene los parametros adicionales de archivos de datos
    frameDuration=load('data/frameDuration.txt');
    windowDuration=load('data/windowDuration.txt');
    thresholdTMP=load('data/umbrales.txt');
    pThresholdLA=thresholdTMP(4);
    pThresholdHA=thresholdTMP(5);
    pThresholdL=thresholdTMP(6);
    pThresholdH=thresholdTMP(7);
    zThresholdL=thresholdTMP(8);
    zThresholdH=thresholdTMP(9);

    %Diseño del FIR pasa altas para eliminar ruido
    BHigh=FIR(samplingFrequency,0.9*100,100,100,100,'high');
    %Respuesta al impulso para el FIR de pre-enfasis
    BPreEmphasis=[1 -0.4];
    %Filtrado de la señal
    data=conv(BHigh,data);
    data=conv(BPreEmphasis,data);
    %Normalizacion
    data=data/max(abs(data));
```

```
    %Calculo del numero de muestras por trama y por ventana
    frameSamples=round(samplingFrequency*frameDuration);
    windowSamples=round(samplingFrequency*windowDuration);
    %Numero de segmentos que forman la señal
    nSegments=floor((length(data)-windowSamples)/frameSamples)+1;

    %Generacion de una ventana de Hamming
    W=hamming(windowSamples);

    %Inicializacion del vector de salida bVoiced
    bVoiced=zeros(1,nSegments);

    for(i=0:nSegments-1)
        %Ventana de 200ms para el calculo de la potencia promedio en la cercania
de la trama
        Pwindow=data(max(0,i-
round(0.1/frameDuration))*frameSamples+1:min(nSegments-
1,i+round(0.1/frameDuration))*frameSamples+windowSamples);
        %Calculo de la potencia promedio, acotada por los limites absolutos
        Pmax=min(max(pThresholdLA^2,meanPower(Pwindow)),pThresholdHA^2);
        %Segmentacion y ventaneo de la señal
        currentSegment=data(i*frameSamples+1:i*frameSamples+windowSamples).*W;
        %Clasificacion del segmento como vocalizado o no vocalizado

bVoiced(i+1)=analyze(currentSegment,Pmax,windowDuration,pThresholdL,pThresholdH,z
ThresholdL,zThresholdH);
    end;
    %Filtrado de segmentos erroneamente clasificados como no vocalizados
    bVoiced=lowPass(bVoiced,0,5);
    %Eliminacion de ruido de corta duracion clasificado como vocalizado
    bVoiced=lowPass(bVoiced,1,5);
return;

%bIsVoiced=analyze(data,Pmax,windowDuration,pThresholdL,pThresholdH,zThresholdL,z
ThresholdH)
```

```
%Decide si un segmento enviado en el vector data presenta caracteristicas

%de un sonido vocalizado o no con base en su energia y el numero de cruces

%por cero que se presenten dentro del mismo

function
bIsVoiced=analyze(data,Pmax,windowDuration,pThresholdL,pThresholdH,zThresholdL,zT
hresholdH);

    bIsVoiced=0;

    bIsVoicedArray=zeros(1,2);

    %Calculo de la potencia promedio del segmento

    Pprom=meanPower(data);

    %Se evalua si la razon entre la potencia promedio del segmento y la de su
cercania es menor a

    %pThresholdL (0), mayor a pThresholdL y menor al pThresholdH (1), o mayor a
pThresholdH (2)

    bIsVoicedArray(1)=(Pprom>=pThresholdL*Pmax)+(Pprom>=pThresholdH*Pmax);

    %Obtencion del numero de cruces por cero en el segmento

    Z=zeroCross(data);

    %Se determina si el numero de cruces por cero por segundo de grabacion es
menor a zThresholdL (0),

    %mayor a zThresholdL y menor a zThresholdH (1), o mayor a zThresholdH (2)


bIsVoicedArray(2)=(Z>=zThresholdL*windowDuration)+(Z>=zThresholdH*windowDuration)
;

    %Decide si el segmento es vocalizado o no vocalizado con base en los 2
criterios anteriores

    bIsVoiced=(bIsVoicedArray(1)==2 | (bIsVoicedArray(1)==1 &
bIsVoicedArray(2)==1));

return;


%Pprom=meanPower(x)

%Calcula la potencia promedio de la señal x

function Pprom=meanPower(x);

    Pprom=x'*x/length(x);

return;


%Z=zeroCross(x)
```

```
%Determina el numero de veces que la señal x cruza por cero

function Z=zeroCross(x);

    Z=sum(abs(sign(x(2:length(x)))-sign(x(1:length(x)-1))))/2;

return;


%s=lowPass(data,symbol,windowSize)

%Filtrado modal de las apariciones de symbol dentro del vector data

function s=lowPass(data,symbol,windowSize);

    s=data;

    %Busca las apariciones de symbol

    sIndex=find(data==symbol);

    for(i=1:length(sIndex))

        %Filtrado modal con una ventana de windowSize elementos

        if(length(find(data(max(1,sIndex(i)-floor((windowSize-
1)/2)):min(length(data),sIndex(i)+ceil((windowSize-
1)/2)))~=symbol))>=ceil(windowSize/2))

            s(sIndex(i))=(~symbol);

        end;

    end;

return;
```

featureExtraction.m

```
%featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurat
ion,supportThreshold,vectorLength,method,bDisplay,tVoiced)

%

%Recibe:

%   data                - Vector de muestras de la señal a analizar

%   samplingFrequency   - Frecuencia de muestreo para data

%   frameDuration       - Intervalo de tiempo, en segundos, entre tramas

%   windowDuration      - Duracion de la ventana en segundos

%   supportThreshold    - Umbral para el soporte numerico del wavelet

%   vectorLength        - Numero de caracteristicas a obtener

%   method              - Metodo para la extraccion de caracteristicas

%   bDisplay            - Bandera que indica si se desea desplegar en pantalla el
progreso
```

```
%   tVoiced             - Eje de tiempo para las secciones vocalizadas
%
%Utiliza los archivos:
%   subbands.txt    - Lista de las bandas criticas a utilizar en la
%                     descomposicion
%
%Regresa:
%   featureVector  - Vectores de parametros obtenidos de data (1 por trama)


function
featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,vectorLength,method,bDisplay,tVoiced);
    %Inicializacion de parametros omitidos
    if(nargin<9)
        tVoiced=1:length(data);
    end;
    if(nargin<8)
        bDisplay=0;
    end;
    if(nargin<7)
        method='ear';
    end;
    if(vectorLength<1 | nargin<7)
        return;
    end;

    data=data(:);

    %Calculo del numero de muestras por trama y por ventana
    frameSamples=round(samplingFrequency*frameDuration);
    windowSamples=round(samplingFrequency*windowDuration);
    %Numero de segmentos que forman la señal
    nSegments=floor((length(data)-windowSamples)/frameSamples)+1;
```

```
    %Inicializacion de variables para el analisis
    featureVector=[];
    newFeatureVector=zeros(vectorLength,1);
    waveletDecomposition=[];

    if(strcmp(method,'haar'))
        %Utilizar una descomposicion mediante el wavelet de Haar
        if(bDisplay)
            fprintf('\tComputando la descomposicion de la señal a %d
niveles',ceil(log2(length(data))));
        end;
        originalLength=length(data);
        %Ajuste de la ventana a la siguiente potencia de 2
        windowSize=2^ceil(log2(length(data)));
        %Agregar ceros a la señal si es necesario
        data=[data;zeros(windowSize-length(data),1)];
        %Descomposicion por el wavelet de Haar
        [wD,L]=haarDWT(data);
        %Reacomodo del resultado en forma matricial
        index=1;
        for(i=2:length(L)-1)
            if(bDisplay)
                fprintf('.');
            end;
            %Se toma el i-primer nivel de la descomposicion
            temp=wD(index:index+L(i)-1)';
            index=index+L(i);
            sMult=ceil(originalLength/length(temp));
            %Cada coeficiente abarca todo su cono de influencia
            temp=resample(temp,sMult,1,0);
            temp=[repmat(temp(1),1,floor(sMult/2)) temp];
            %El resultado forma la i-primera fila de la matriz
```

```
            waveletDecomposition=[waveletDecomposition;temp(1:originalLength)];
        end;
        if(length(waveletDecomposition(:,1))>vectorLength/2-1)
            %El analisis se limita al numero de bandas solicitadas
            waveletDecomposition=waveletDecomposition(end-
vectorLength/2+2:end,:);
        end;
        %Se invierte la matriz, la primera fila representara la escala mas
        %grande (y frecuencia mas pequeña)
        waveletDecomposition=flipud(waveletDecomposition);
        %Se recupera la señal original
        data=data(1:originalLength);
        if(bDisplay)
            fprintf('ok\n');
        end;

    elseif(strcmp(method,'daub4'))
        %Utilizar una descomposicion mediante un wavelet de Daubechies de 4
coeficientes
        if(bDisplay)
            fprintf('\tComputando la descomposicion de la señal a %d
niveles',min(vectorLength/2-1,wmaxlev(length(data),'db4')));
        end;
        %Calcula la descomposicion por el wavelet de Daubechies en todas
        %las escalas requeridas o permitidas
        [wD,L]=wavedec(data,min(vectorLength/2-
1,wmaxlev(length(data),'db4')),'db4');
        %Reacomodo del resultado en forma matricial
        index=1;
        for(i=2:length(L)-1)
            if(bDisplay)
                fprintf('.');
            end;
            %Se toma el i-primer nivel de la descomposicion
```

```
            temp=wD(index:index+L(i)-1)';
            index=index+L(i);
            sMult=ceil(L(end)/length(temp));
            %Cada coeficiente abarca todo su cono de influencia
            temp=resample(temp,sMult,1,0);
            temp=[repmat(temp(1),1,floor(sMult/2)) temp];
            %El resultado forma la i-primera fila de la matriz
            waveletDecomposition=[waveletDecomposition;temp(1:L(end))];
        end;
        %Se invierte la matriz, la primera fila repressenta la escala mas
        %grande (y frecuencia mas pequeña)
        waveletDecomposition=flipud(waveletDecomposition);
        if(bDisplay)
            fprintf('ok\n');
        end;

    elseif(strcmp(method,'filters'))
        %Utilizar una descomposicion mediante un banco de filtros de ancho de
banda critico
        if(bDisplay)
            fprintf('\tComputando la descomposicion de la señal a %d
niveles',vectorLength/2-1);
        end;
        for(i=1:vectorLength/2-1)
            if(bDisplay)
                fprintf('.');
            end;
            %Se lee la respuesta al impulso de cada filtro desde un archivo de
datos
            if(exist(['data\filters\B' num2str(i) '.txt'],'file'))
                B=load(['data\filters\B' num2str(i) '.txt']);
                %Se filtra la señal empleando cada uno de los filtros
                signalWindow=conv(data,B)';
```

```
                %Y el resultado se agrega directamente en la matriz de la
descomposicion

waveletDecomposition=[waveletDecomposition;signalWindow(1:length(data))];
            end;
        end;
        if(bDisplay)
            fprintf('ok\n');
        end;
    else
        %Utilizar una descomposicion mediante un wavelet basado en el modelado
del oido interno
        %La forma de la distribucion gama se fija en 3
        alpha=3;
        %Se generan las bandas por defecto para realizar la descomposicion
        subbands=1:min(vectorLength/2-
1,floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1))));
        %Y se buscan bandas predefinidas en un archivo de datos
        if(exist('data/subbands.txt','file'))
            subbands=load('data/subbands.txt');
            subbands=[subbands(1:min(length(subbands),vectorLength/2-
1));(1:vectorLength/2-1-length(subbands))'];
        end;
        %Se repiten bandas para obtener todas las solicitadas
        subbands=repmat(subbands,ceil((vectorLength/2-1)/length(subbands)),1);
        subbands=subbands(1:vectorLength/2-1);
        if(bDisplay)
            fprintf('\tComputando la descomposicion de la señal en %d
niveles',vectorLength/2-1);
        end;
        for(Z=1:vectorLength/2-1)
            if(bDisplay)
                fprintf('.');
            end;
            %Se calcula la frecuencia caracteristica de cada banda critica
```

```
            fc=325*(exp(2*subbands(Z)/7)-1)/exp(subbands(Z)/7);

            %Y se obtienen los coeficientes de la transformada wavelet,
agregandolos directamente a la matriz de la descomposicion


waveletDecomposition=[waveletDecomposition;earDWT(data,samplingFrequency,fc,alpha
,windowDuration,supportThreshold,0,tVoiced)'];

        end;

        if(bDisplay)

            fprintf('ok\n');

        end;

    end;


    %Si la descomposicion abarca menos niveles de los requeridos por la red

    %neuronal, se agregan ceros y termina la extraccion de caracteristicas

    if(length(waveletDecomposition(:,1))<vectorLength/2-1)

        waveletDecomposition=[waveletDecomposition;zeros(vectorLength/2-1-
length(waveletDecomposition(:,1)),length(waveletDecomposition(1,:)))];

    end;


    %Se construyen vectores de parametros para cada trama

    for(j=0:nSegments-1)

        %Se conserva el vector de parametros anterior

        oldFeatureVector=newFeatureVector;

        %Y se inicializa el nuevo vector de parametros

        newFeatureVector=zeros(vectorLength,1);

        if(~isempty(find(tVoiced==j*frameSamples+1)) |
~isempty(tVoiced==j*frameSamples+windowSamples))

            %Solo se generan vectores para las secciones vocalizadas

            currentData=data(j*frameSamples+1:j*frameSamples+windowSamples);

            %La primer componente es la energia del segmento

            newFeatureVector(1)=currentData'*currentData/length(currentData);

            for(i=2:ceil(vectorLength/2))

                %A continuacion se concatena la energia de cada nivel de la
descomposicion
```

```
                signalDecomposition=waveletDecomposition(i-
1,j*frameSamples+1:j*frameSamples+windowSamples);

newFeatureVector(i)=signalDecomposition*signalDecomposition'/length(signalDecompo
sition);
            end;
            %Por ultimo se agrega el cambio entre el segmento anterior y el
actual
            for(i=ceil(vectorLength/2)+1:vectorLength)
                newFeatureVector(i)=newFeatureVector(i-ceil(vectorLength/2))-
oldFeatureVector(i-ceil(vectorLength/2));
            end;
            %Y el vector es agregado a la variable de salida
            featureVector=[featureVector newFeatureVector];
        else
            %Las tramas no vocalizadas obtienen vectores marcados con todas sus
componentes iguales a -1
            newFeatureVector=zeros(size(newFeatureVector));
            featureVector=[featureVector repmat(-1,size(newFeatureVector))];
        end;
    end;

    %Se eliminan valores identicos a cero para evitar divisiones entre cero
    %durante el entrenamiento de la red
    featureVector(find(featureVector==0))=1e-10;
return;

%[C,L]=haarDWT(data)
%Descompone la señal data utilizando el wavelet de Haar
function [C,L]=haarDWT(data);
    data=data(:);
    %Inicializa el registro de transiciones en la salida
    C=[];
    %Inicializa la variable de salida
    L=length(data);
```

```
    %Se utilizan todos los niveles de descomposicion permitidos
    for(i=1:floor(log2(length(data))))
        %Se obtiene la descomposicion del nivel i
        C=[C;flipud((data(1:2:end)-data(2:2:end))/sqrt(2))];
        %Calcula la aproximacion del nivel i
        data=(data(1:2:end)+data(2:2:end))/sqrt(2);
        %Registra la transicion de C en la variable de salida L
        L=[L;L(end)/2];
    end;
    %Reacomoda la señal para que inicie co nla descomposicion a frecuencias
    %mas altas
    C=flipud([C;data]);
    L=flipud([L;1]);
return;
```

earDWT.m

```
%phi=earDWT(data,samplingFrequency,characteristicFrequency,alpha,windowDuration,s
upportThreshold,bDisplay,tVoiced,maxScales)
%
%Recibe:
%   data                    - Muestras de una señal de audio
%   samplingFrequency       - Frecuencia utilizada para el muestreo de data
%   characteristicFrequency - Frecuencia caracteristica que se desea analizar
%   alpha                   - Forma de la distribucion gamma usada como
envolvente
%   windowDuration          - Duracion de la ventana en segundos
%   supportThreshold        - Umbral para el soporte numerico del wavelet
%   bDisplay                - Bandera para mostrar mensajes sobre el progreso
%   tVoiced                 - Eje de tiempos vocalizados
%   maxScales               - Numero maximo de escalas a utilizar para el
%                             enmascaramiento bitonal (0 - inf)
%
%Regresa:
%   phi - Coeficientes de la transformada wavelet de data que poseen
```

```
%         informacion acerca de sus componentes de frecuencia
characteristicFrequency, empleando

%         un wavelet madre basado en el modelado del oido humano


function
phi=earDWT(data,samplingFrequency,characteristicFrequency,alpha,windowDuration,su
pportThreshold,bDisplay,tVoiced,maxScales);

    %Inicializacion de parametros omitidos

    if(nargin<9)

        maxScales=1;

    end;

    if(nargin<8)

        tVoiced=[];

    end;

    if(nargin<7)

        bDisplay=0;

    end;


    %Adecuacion de forma en parametros vectoriales

    data=data(:);

    tVoiced=tVoiced(find(tVoiced~=0));


    if(bDisplay)

        fprintf('characteristicFrequency=%1.5g\n',characteristicFrequency);

    end;


    %Obtencion de escalas dinamicas para simular el enmascaramiento bitonal

[s,compressedS]=dynamicScale(data,samplingFrequency,characteristicFrequency,alpha
,windowDuration,maxScales);


    %Calculo del wavelet madre escalado por todos los valores de compressedS

    psi_j=[];

    for(j=1:length(compressedS))
```

```
        psi_j=[psi_j
sampledDaughterWavelet(alpha,compressedS(j),0,samplingFrequency,windowDuration)];
    end;
    %Forzar un soporte numerico de acuerdo con el umbral supportThreshold
    psi_j(find(abs(psi_j)<supportThreshold))=0;

    %Eleccion del metodo para obtener los coeficientes de la transformada wavelet
    if(length(compressedS)==1)
        %Si solo se utiliza una escala,
phi=flipud(iFFT(FFT(flipud(data))*FFT(psi)))
        method='FFT Filtering';
    else
        %De lo contrario, obtener las correlaciones en cero de la señal con cada
wavelet hija
        method='Zero Lag Cross Correlation';
    end;

    if(bDisplay)
        fprintf('\tUsing %d scales\tMethod: %s\n',length(compressedS),method);
    end;

    %Calculo de los coeficientes de la transformada wavelet
    if(strcmp(method,'FFT Filtering'))
        %Utilizar la transformada rapida de Fourier
        phi=halfCorrelation(data,psi_j(1:min(end,length(data))));
    else
        %Obtener los coeficientes directamente
        phi=zeros(size(data));
        k=0;
        if(~isempty(tVoiced))
            %Solo calcular la transformada para las muestras vocalizadas
            data(find(tVoiced==0))=0;
        end;
        while(k<length(data))
```

```
            %Coeficiente para la escala s(k+1) y traslacion k

phi_jk=zeroLagCorrelation(data(k+1:min(length(data),floor(k+windowDuration*sampli
ngFrequency))),psi_j(:,find(s(k+1)==compressedS)));

            %Se agrega el coeficiente en todo su cono de influencia

phi(k+1:min(length(phi),round(k+2*pi*samplingFrequency*s(k+1))))=phi_jk;

            %Siguiente traslacion, fuera del cono de influencia

            k=round(k+2*pi*samplingFrequency*s(k+1));

        end;

    end;

return;


%C=zeroLagCorrelation(A,B)

%Calcula la correlacion entre las señales A y B para un retraso de cero

function C=zeroLagCorrelation(A,B);

    %Solo se requiere la region donde ambas señales son distintas de cero

    B=B(1:length(A));

    index=find((A~=0)&(B~=0));

    A=A(index);

    B=B(index);


    C=0;

    if(~isempty(B))

        %Calculo de la correlacion en cero

        C=A'*B/length(B);

    end;

return;


%C=halfCorrelation(A,B);

%Calcula la correlacion cruzada entre las señales A y B para retrasos positivos

%utilizando la transformada rapida de Fourier

function C=halfCorrelation(A,B);

    C=flipud(fftfilt(B,flipud(A)))/length(B);
```

```
return;


%Z=bark(f)

%Convierte el valor de frecuencia f en Hertz a Barks

function Z=bark(f);

    Z=7*log(f/650+sqrt((f/650).^2+1));

return;


%f=ibark(Z)

%Convierte el valor de frecuencia Z en Barks a Hertz

function f=ibark(Z);

    f=325*(exp(2*Z/7)-1)./exp(Z/7);

return;


%[s,compressedS]=dynamicScale(data,samplingFrequency,characteristicFrequency,alph
a,windowDuration,maxScales)

%Genera las escalas adecuadas para el analisis de la señal, de forma que la
descomposicion imite al

%enmascaramiento bitonal

function
[s,compressedS]=dynamicScale(data,samplingFrequency,characteristicFrequency,alpha
,windowDuration,maxScales);

    if(maxScales==1)

        %Si solo se utiliza una escala, esta se determina por la frecuencia
caracteristica

        s=repmat(1/characteristicFrequency,size(data));

        compressedS=1/characteristicFrequency;

    else

        if(maxScales==0)

            %Utilizar todas las escalas posibles

            maxScales=length(s);

        end;

        %Obtiene el valor maximo para la escala
```

```
Tnarrow=max(2*log10(2*bark(characteristicFrequency))/(pi*characteristicFrequency)
,1/characteristicFrequency);

        %Calcula el valor minimo de la escala

        Twide=max((58.4341*min(15500,max(760,characteristicFrequency))^-
0.7)*Tnarrow,2/samplingFrequency);

        %Siguiente banda critica

        ff=ibark(bark(characteristicFrequency)+1);

        %Escala para la descomposicion del lazo de control

        Tf=(0.2+0.8*Twide/Tnarrow)/ff;

        %Obtencion de la wavelet hija para la descomposicion del lazo

        psi=sqrt(1+(2*pi*Tf*(characteristicFrequency-
ff))^2)*sampledDaughterWavelet(alpha,Tf,0,samplingFrequency,windowDuration);

        %Una escala, se utiliza la transformada rapida de Fourier

        phi=halfCorrelation(data,psi(1:min(end,length(data))));

        %Filtrado pasa bajas de los resultados con una frecuencia de corte de
800Hz

        V_LP=conv(phi,FIR(samplingFrequency,800*0.9,800,100,100,'low'));

        %Calculo del factor de escala para cada muestra

        kS=0.05+0.95*min(0.5393,max(0.0191,61.5111*characteristicFrequency^-0.7-
0.0526)).^(abs(V_LP)/0.37);

        %Obtencion de la escala para cada traslacion, acotada por los limites
Twide y Tnarrow

        s=max(Twide,min(Tnarrow,Tnarrow*kS(1:length(data))));

        %Redondeo en las escalas de acuerdo con el teorema de Nyquist-Shannon

s=10^(floor(log10(1/samplingFrequency)))*round(s*10^(ceil(log10(samplingFrequency
))));


        %Compresion del rango dinamico de la escala para que tome solo maxScales
valores

        compressedS=[];

        for(j=1:length(s))

            if(isempty(find(s(j)==compressedS)))

                if(length(compressedS)<maxScales)

                    compressedS=[compressedS;s(j)];

                else
```

```
                    s(j)=s(find(abs(compressedS-s(j))==min(abs(compressedS-
s(j)))));
                end;
            end;
        end;
    end;
return;


%psi_a_sT=sampledDaughterWavelet(alpha,s,T,samplingFrequency,windowDuration)
%Obtiene las muestras de un wavelet hija escalado por s y trasladado por T
%con una frecuencia de muestreo de samplingFrequency
function
psi_a_sT=sampledDaughterWavelet(alpha,s,T,samplingFrequency,windowDuration);
    %Genera un eje de tiempo asegurando que el valor final del wavelet sea cero
    t=(1/samplingFrequency:1/samplingFrequency:s*(round(2*windowDuration/s)/2-
0.25)+T)';
    %Y agrega ceros para completar el tiempo de ventana
    t=[t;zeros(round(samplingFrequency*(windowDuration+T-t(end))),1)];
    %Obtiene la wavelet hija muestreada
    psi_a_sT=1/sqrt(s)*motherWavelet(alpha,(t-T)/s);
return;


%psi_a=motherWavelet(alpha,t)
%Evalua el wavelet madre con forma alpha para los tiempos marcados por t
function psi_a=motherWavelet(alpha,t);
    t(find(t<0))=0;
    psi_a=1/prod(1:alpha-1)*t.^(alpha-1).*exp(-t).*cos(2*pi*t);
return;
```

createData.m

```
%createData(path)
%
%Recibe:
%   path    - Ruta hacia el directorio que contiene los datos de entrenamiento
```

```
%
%Utiliza los archivos:
%   frameDuration.txt       - Duracion en segundos de cada trama
%   windowDuration.txt      - Longitud de la ventana en segundos
%   method.txt              - Metodo para la extraccion de caracteristicas
%   supportThreshold.txt    - Umbral para el soporte numerico del wavelet
%
%Escribe los archivos:
%   nInputs.txt     - Numero de caracteristicas de entrada para la red neuronal
%   speechIn.txt    - Vectores de parametros extraidos de los archivos de
%                     entrenamiento para la red neuronal
%   speechOut.txt   - Salida esperada para cada vector de parametros obtenido de
%                     los archivos de entrenamiento

function createData(path);
    if(nargin<1)
        path='audio/trainingData';
    end;
    path(find(path=='\'))='/';

    %Obtiene los parametros para la extraccion de caracteristicas
    frameDuration=load('data/frameDuration.txt');
    windowDuration=load('data/windowDuration.txt');
    method=char(load('data/method.txt'));
    supportThreshold=load('data/supportThreshold.txt');

    %Inicializa las variables de salida
    nInputs=0;
    speechIn=[];
    speechOut=[];
    fprintf('Extrayendo datos de entrenamiento de archivos de audio...\n');
    n=0;
```

```
    %Busca archivos de audio adecuados para el entrenamiento

    while((exist([path '/a (' num2str(n) ').wav'],'file')) | (exist([path '/e ('
num2str(n) ').wav'],'file'))...

            | (exist([path '/i (' num2str(n) ').wav'],'file')) | (exist([path '/o
(' num2str(n) ').wav'],'file'))...

            | (exist([path '/u (' num2str(n) ').wav'],'file'))...

            | (exist([path '/x (' num2str(n) ').wav'],'file')))


        if(exist([path '/a (' num2str(n) ').wav'],'file'))

            %Procesa un archivo que contiene solo alofonos de la "A" dentro de
sus segmentos vocalizados

            fprintf(['\tProcesando archivo ' path '/a (' num2str(n)
').wav''...\n\t']);

            [data,samplingFrequency]=wavRead([path '/a (' num2str(n) ').wav']);


            if(nInputs==0)

nInputs=2*(floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)
))+1);

            end;


            %Acondicionamiento y deteccion de segmentos vocalizados

            [data,bVoiced]=isVoiced(data,samplingFrequency);


            %Genera un eje temporal para las secciones vocalizadas

            tVoiced=1:length(data);

            for(i=1:length(bVoiced))

                if(~bVoiced(i))

                    tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;

                end;

            end;


            %Extraccion de caracteristicas de la señal
```

```
featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,1,tVoiced);


            %Actualizacion de los datos de salida con los nuevos datos de
entrenamiento

            for(i=1:length(bVoiced))

                if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))

                    speechIn=[speechIn featureVector(:,i)];

                    %Codigo de la vocal A

                    speechOut=[speechOut [1 0 0 0 0]'];

                end;

            end;

            fprintf('\t\t%d segmentos procesados\n',length(find(bVoiced)));

        end;


        if(exist([path '/e (' num2str(n) ').wav'],'file'))

            %Procesa un archivo que contiene solo alofonos de la "E" dentro de
sus segmentos vocalizados

            fprintf(['\tProcesando archivo ' path '/e (' num2str(n)
').wav''...\n\t']);

            [data,samplingFrequency]=wavRead([path '/e (' num2str(n) ').wav']);


            if(nInputs==0)

nInputs=2*(floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)
))+1);

            end;


            %Acondicionamiento y deteccion de segmentos vocalizados

            [data,bVoiced]=isVoiced(data,samplingFrequency);


            %Genera un eje temporal para las secciones vocalizadas

            tVoiced=1:length(data);

            for(i=1:length(bVoiced))
```

```
                if(~bVoiced(i))

                    tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;

                end;

            end;


            %Extraccion de caracteristicas de la señal


featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,1,tVoiced);


            %Actualizacion de los datos de salida con los nuevos datos de
entrenamiento

            for(i=1:length(bVoiced))

                if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))

                    speechIn=[speechIn featureVector(:,i)];

                    %Codigo de la vocal E

                    speechOut=[speechOut [0 1 0 0 0]'];

                end;

            end;

            fprintf('\t\t%d segmentos procesados\n',length(find(bVoiced)));

        end;


        if(exist([path '/i (' num2str(n) ').wav'],'file'))

            %Procesa un archivo que contiene solo alofonos de la "I" dentro de
sus segmentos vocalizados

            fprintf(['\tProcesando archivo ' path '/i (' num2str(n)
').wav''...\n\t']);

            [data,samplingFrequency]=wavRead([path '/i (' num2str(n) ').wav']);


            if(nInputs==0)


nInputs=2*(floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)
))+1);

            end;
```

```
            %Acondicionamiento y deteccion de segmentos vocalizados
            [data,bVoiced]=isVoiced(data,samplingFrequency);

            %Genera un eje temporal para las secciones vocalizadas
            tVoiced=1:length(data);
            for(i=1:length(bVoiced))
                if(~bVoiced(i))
                    tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
                end;
            end;

            %Extraccion de caracteristicas de la señal

featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,1,tVoiced);

            %Actualizacion de los datos de salida con los nuevos datos de
entrenamiento
            for(i=1:length(bVoiced))
                if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))
                    speechIn=[speechIn featureVector(:,i)];
                    %Codigo de la vocal I
                    speechOut=[speechOut [0 0 1 0 0]'];
                end;
            end;
            fprintf('\t\t%d segmentos procesados\n',length(find(bVoiced)));
        end;

        if(exist([path '/o (' num2str(n) ').wav'],'file'))
            %Procesa un archivo que contiene solo alofonos de la "O" dentro de
sus segmentos vocalizados
```

```
            fprintf(['\tProcesando archivo ' path '/o (' num2str(n)
').wav''...\n\t']);

            [data,samplingFrequency]=wavRead([path '/o (' num2str(n) ').wav']);


            if(nInputs==0)

nInputs=2*(floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)
))+1);

            end;


            %Acondicionamiento y deteccion de segmentos vocalizados

            [data,bVoiced]=isVoiced(data,samplingFrequency);


            %Genera un eje temporal para las secciones vocalizadas

            tVoiced=1:length(data);

            for(i=1:length(bVoiced))

                if(~bVoiced(i))

                    tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;

                end;

            end;


            %Extraccion de caracteristicas de la señal

featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,1,tVoiced);


            %Actualizacion de los datos de salida con los nuevos datos de
entrenamiento

            for(i=1:length(bVoiced))

                if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))

                    speechIn=[speechIn featureVector(:,i)];

                    %Codigo de la vocal O

                    speechOut=[speechOut [0 0 0 1 0]'];

                end;
```

```
            end;
            fprintf('\t\t%d segmentos procesados\n',length(find(bVoiced)));
        end;

        if(exist([path '/u (' num2str(n) ').wav'],'file'))
            %Procesa un archivo que contiene solo alofonos de la "U" dentro de
sus segmentos vocalizados
            fprintf(['\tProcesando archivo ' path '/u (' num2str(n)
').wav''...\n\t']);
            [data,samplingFrequency]=wavRead([path '/u (' num2str(n) ').wav']);

            if(nInputs==0)

nInputs=2*(floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)
))+1);
            end;

            %Acondicionamiento y deteccion de segmentos vocalizados
            [data,bVoiced]=isVoiced(data,samplingFrequency);

            %Genera un eje temporal para las secciones vocalizadas
            tVoiced=1:length(data);
            for(i=1:length(bVoiced))
                if(~bVoiced(i))
                    tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
                end;
            end;

            %Extraccion de caracteristicas de la señal

featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,1,tVoiced);
```

```
            %Actualizacion de los datos de salida con los nuevos datos de
entrenamiento
            for(i=1:length(bVoiced))
                if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))
                    speechIn=[speechIn featureVector(:,i)];
                    %Codigo de la vocal U
                    speechOut=[speechOut [0 0 0 0 1]'];
                end;
            end;
            fprintf('\t\t%d segmentos procesados\n',length(find(bVoiced)));
        end;

        if(exist([path '/x (' num2str(n) ').wav'],'file'))
            %Procesa un archivo que contiene solo consonantes dentro de sus
segmentos vocalizados
            fprintf(['\tProcesando archivo ' path '/x (' num2str(n)
').wav''...\n\t']);
            [data,samplingFrequency]=wavRead([path '/x (' num2str(n) ').wav']);

            if(nInputs==0)

nInputs=2*(floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)
))+1);
            end;

            %Acondicionamiento y deteccion de segmentos vocalizados
            [data,bVoiced]=isVoiced(data,samplingFrequency);

            %Genera un eje temporal para las secciones vocalizadas
            tVoiced=1:length(data);
            for(i=1:length(bVoiced))
                if(~bVoiced(i))
                    tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
```

```
            end;
        end;

        %Extraccion de caracteristicas de la señal

featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,1,tVoiced);

        %Actualizacion de los datos de salida con los nuevos datos de
entrenamiento
        for(i=1:length(bVoiced))
            if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))
                speechIn=[speechIn featureVector(:,i)];
                %Codigo de las consonantes vocalizadas
                speechOut=[speechOut [0 0 0 0 0]'];
            end;
        end;
        fprintf('\t\t%d segmentos procesados\n',length(find(bVoiced)));
    end;

    n=n+1;
end;
fprintf('Se han obtenido un total de %d datos de
entrenamiento\n',round(length(speechIn(:))/nInputs));

%Almacena los resultados en archivos de datos
save data/nInputs.txt nInputs -ascii;
save data/speechIn.txt speechIn -ascii;
save data/speechOut.txt speechOut -ascii;
return;
```

init.m

```
%init(X,Y,W1r,W2r)
%
%Recibe:
```

```
%   X       - Datos de entrada que se utilizaran durante el entrenamiento de la
%             red neuronal
%   Y       - Datos de salidas esperadas para cada dato de entrada definido en X
%   W1r     - Numero de nodos en la primera capa oculta de la red neuronal
%   W2r     - Numero de nodos en la segunda capa oculta
%
%Escribe los archivos:
%   Xmean.txt   - Media de los datos de entrada que se utilizaran durante el
%                 entrenamiento de la red neuronal
%   Xstd.txt    - Desviacion estandar de los datos de entrada que se utilizaran
%                 durante el entrenamiento de la red neuronal
%   Ymean.txt   - Media de los datos de salidas esperadas que se utilizaran
%                 durante el entrenamiento de la red neuronal
%   Ystd.txt    - Desviacion estandar de los datos de salidas esperadas que se
%                 utilizaran durante el entrenamiento de la red neuronal
%   W1.txt      - Pesos aleatorios para la primer capa oculta de la red neuronal
%   W2.txt      - Pesos aleatorios para la segunda capa oculta de la red neuronal
%   W3.txt      - Pesos aleatorios para la capa de salidas de la red neuronal
%   B1.txt      - Desplazamientos aleatorios para la primer capa oculta de la red
neuronal
%   B2.txt      - Desplazamientos aleatorios para la segunda capa oculta de la
red neuronal
%   B3.txt      - Desplazamientos aleatorios para la capa de salidas de la red
neuronal


function init(X,Y,W1r,W2r);
    W3r=size(Y,1);
    W1c=size(X,1);
    W2c=W1r;
    W3c=W2r;

    %Los pesos y desplazamientos iniciales tomaran valores entre minInitialValue
y maxInitialValue
    minInitialValue=0.1;
```

```
    maxInitialValue=minInitialValue+0.001;

    %Calcula la media y desviacion estandar de los datos de entrenamiento
    Xmean=mean(X')';
    Xstd=std(X')';
    Ymean=mean(Y')';
    Ystd=std(Y')';

    %Genera los pesos y desplazamientos aleatorios
    W1=minInitialValue+randn(W1r,W1c)*(maxInitialValue-minInitialValue);
    B1=minInitialValue+randn(W1r,1)*(maxInitialValue-minInitialValue);
    W2=minInitialValue+randn(W2r,W2c)*(maxInitialValue-minInitialValue);
    B2=minInitialValue+randn(W2r,1)*(maxInitialValue-minInitialValue);
    W3=minInitialValue+randn(W3r,W3c)*(maxInitialValue-minInitialValue);
    B3=minInitialValue+randn(W3r,1)*(maxInitialValue-minInitialValue);

    %Almacena los resultados en archivos de datos
    save data/Xmean.txt Xmean -ascii;
    save data/Xstd.txt Xstd   -ascii;
    save data/Ymean.txt Ymean -ascii;
    save data/Ystd.txt Ystd   -ascii;
    save data/W1.txt W1 -ascii;
    save data/W2.txt W2 -ascii;
    save data/W3.txt W3 -ascii;
    save data/B1.txt B1 -ascii;
    save data/B2.txt B2 -ascii;
    save data/B3.txt B3 -ascii;
return;
```

train.m

```
%train(input,output,eMax,nStat)
%
%Recibe:
```

```
%   input    - Vectores de entrada para el entrenamiento de la red neuronal
%   output   - Vectores de salidas esperadas correspondientes a cada vector
%              de entradas
%   eMax     - Numero maximo de iteraciones a efectuar durante el entrenamiento
%   nStat    - Numero de iteraciones a realizar antes de desplegar el progreso
%              en la consola de Matlab
%
%Utiliza los archivos:
%   Xmean.txt    - Media de los datos de entrada que se utilizaran durante el
%                  entrenamiento de la red neuronal
%   Xstd.txt     - Desviacion estandar de los datos de entrada que se utilizaran
%                  durante el entrenamiento de la red neuronal
%   Ymean.txt    - Media de los datos de salidas esperadas que se utilizaran
%                  durante el entrenamiento de la red neuronal
%   Ystd.txt     - Desviacion estandar de los datos de salidas esperadas que se
%                  utilizaran durante el entrenamiento de la red neuronal
%   W1.txt       - Pesos iniciales para la primer capa oculta de la red neuronal
%   W2.txt       - Pesos iniciales para la segunda capa oculta de la red neuronal
%   W3.txt       - Pesos iniciales para la capa de salidas de la red neuronal
%   B1.txt       - Desplazamientos iniciales para la primer capa oculta de la red
neuronal
%   B2.txt       - Desplazamientos iniciales para la segunda capa oculta de la red
neuronal
%   B3.txt       - Desplazamientos iniciales para la capa de salidas de la red
neuronal
%
%Escribe los archivos:
%   W1.txt  - Pesos para la primer capa oculta de la red neuronal despues del
%             entrenamiento
%   W2.txt  - Pesos para la segunda capa oculta de la red neuronal despues del
%             entrenamiento
%   W3.txt  - Pesos para la capa de salidas de la red neuronal despues del
%             entrenamiento
%   B1.txt  - Desplazamientos para la primer capa oculta de la red neuronal
```

```
%              despues del entrenamiento
%   B2.txt  - Desplazamientos para la segunda capa oculta de la red neuronal
%              despues del entrenamiento
%   B3.txt  - Desplazamientos para la capa de salidas de la red neuronal despues
%              del entrenamiento


function train(input,output,eMax,nStat);
    %Error cuadratico medio deseado
    errorMin=0.1;
    %Factor de aprendizaje
    learningRatio=0.01;
    %Coeficiente de incremento para el factor de aprendizaje en caso de reducir
el error
    learningRatioI=1.05;
    %Coeficiente de decremento para el factor de aprendizaje en caso de
    %aumentar el error mas alla de lo tolerado
    learningRatioD=0.7;
    %Peso dado al cambio anterior en los pesos y desplazamientos para
    %calcular el cambio siguiente en los mismos parametros (Solo en caso de
    %que el error no aumente mas de lo tolerado)
    delayRatioI=0.95;
    %Maximo incremento en el error tolerado
    errorRatio=1.04;

    if(nargin==3)
        nStat=0;
    end;

    %Obtiene la media y desviacion estandar observadas en las entradas y
    %salidas durante el entrenamiento
    Xmean=load('data/Xmean.txt');
    Xstd=load('data/Xstd.txt');
    Ymean=load('data/Ymean.txt');
```

```
Ystd=load('data/Ystd.txt');

%Normaliza las entradas y salidas esperadas para la red
%Esto evita que valores relativamente grandes para alguna entrada o
%salida afecten el desempeño general del entrenamiento
nIn=size(input,2);
for(i=1:nIn)
    input(:,i)=(input(:,i)-Xmean)./Xstd;
end;
nOut=size(output,1);
for(i=1:nIn)
    output(:,i)=(output(:,i)-Ymean)./Ystd;
end;

%Valor cuadratico medio de las salidas deseadas
meanSquareOutput=meanSquare(output);

%Lee de archivos de datos los pesos y desplazamientos actuales
W1=load('data/W1.txt');
W2=load('data/W2.txt');
W3=load('data/W3.txt');
B1=load('data/B1.txt');
B2=load('data/B2.txt');
B3=load('data/B3.txt');

%Inicializa los cambios en pesos y desplazamientos
dW1=zeros(size(W1));
dB1=zeros(size(B1));
dW2=zeros(size(W2));
dB2=zeros(size(B2));
dW3=zeros(size(W3));
dB3=zeros(size(B3));
```

```
    %No existe un cambio en los parametros previo a la primera iteracion
    delayRatio=0;

    %Realiza una clasificacion inicial de las entradas y determina el error como
la
    %diferencia entre esta clasificacion y la esperada
    y1=tanh(W1*input+B1*ones(1,nIn));
    y2=tanh(W2*y1+B2*ones(1,nIn));
    y3=W3*y2+B3*ones(1,nIn);
    E=output-y3;

    %Calcula la suma de los errores iniciales al cuadrado
    sumSquareError=sum(sum(E.* E));

    %Determina la diferencia entre la salida inicial y la deseada para cada
    %nodo de la red. La derivada de la funcion de activacion esta dada como:
    %diff(tanh(x)) = 1-tanh(x)^2
    D3=E;
    D2=(1-(y2.*y2)).*(W3'*D3);
    D1=(1-(y1.* y1)).*(W2'*D2);

    %Comienza el entrenamiento
    for(epoch=1:eMax)
        %Obtiene el error cuadratico medio para los parametros actuales
        meanSquareError=sumSquareError/(nOut*nIn);

        %Cada nStat iteraciones...
        if(mod(epoch,nStat)==0)
            %Despliega el estado del entrenamiento
            fprintf('Iteracion: %d   Error cuadratico medio:
%f\n',epoch,100*meanSquareError/meanSquareOutput);
            %Almacena los parametros obtenidos hasta el momento
            save data/W1.txt W1 -ascii;
```

```
    save data/W2.txt W2 -ascii;
    save data/W3.txt W3 -ascii;
    save data/B1.txt B1 -ascii;
    save data/B2.txt B2 -ascii;
    save data/B3.txt B3 -ascii;
end;

if(meanSquareError/meanSquareOutput<=errorMin)
    %Si se alcanza el error deseado, termina el entrenamiento
    epoch=epoch-1;
    break;
end;

%Determina el cambio para los pesos y desplazamientos de acuerdo con
%el factor de aprendizaje y la salida deseada para cada nodo
h=(1-delayRatio)*learningRatio;
dW1=delayRatio*dW1+h*(D1*input');
dB1=delayRatio*dB1+h*(D1*ones(nIn,1));
dW2=delayRatio*dW2+h*(D2*y1');
dB2=delayRatio*dB2+h*(D2*ones(nIn,1));
dW3=delayRatio*dW3+h*(D3*y2');
dB3=delayRatio*dB3+h*(D3*ones(nIn,1));

%Obtiene los nuevos pesos y desplazamientos
W1Tmp=W1+dW1;
B1Tmp=B1+dB1;
W2Tmp=W2+dW2;
B2Tmp=B2+dB2;
W3Tmp=W3+dW3;
B3Tmp=B3+dB3;

%Realiza una clasificacion con los nuevos pesos y desplazamientos
y1Tmp=tanh(W1Tmp*input+B1Tmp*ones(1,nIn));
```

```
y2Tmp=tanh(W2Tmp*y1Tmp+B2Tmp*ones(1,nIn));
y3Tmp=W3Tmp*y2Tmp+B3Tmp*ones(1,nIn);

%Determina el error cometido con los nuevos pesos y desplazamientos
ETmp=output-y3Tmp;
sumSquareErrorTmp=sum(sum(ETmp.*ETmp));

%Ajusta el factor de aprendizaje de acuerdo con la variacion del error
if(sumSquareErrorTmp>sumSquareError*errorRatio)
    learningRatio=learningRatio*learningRatioD;
    delayRatio=0;
else
    delayRatio=delayRatioI;
    if(sumSquareErrorTmp<sumSquareError)
        learningRatio=learningRatio*learningRatioI;
    end;

    %Si el error disminuye, actualiza los pesos y desplazamientos
    W1=W1Tmp;
    B1=B1Tmp;
    W2=W2Tmp;
    B2=B2Tmp;
    W3=W3Tmp;
    B3=B3Tmp;
    %y los parametros de entrenamiento
    y1=y1Tmp;
    y2=y2Tmp;
    E=ETmp;
    sumSquareError=sumSquareErrorTmp;
    D3=E;
    D2=(1-(y2.*y2)).*(W3'*D3);
    D1=(1-(y1.*y1)).*(W2'*D2);
end;
```

```
    end;

    %Al alcanzar un critero de paro (numero maximo de iteraciones o error
    %deseado) guarda los nuevos pesos y desplazamientos en archivos de datos
    save data/W1.txt W1 -ascii;
    save data/W2.txt W2 -ascii;
    save data/W3.txt W3 -ascii;
    save data/B1.txt B1 -ascii;
    save data/B2.txt B2 -ascii;
    save data/B3.txt B3 -ascii;
return;


%r=meanSquare(x)
%Obtiene el valor cuadratico medio de la señal x
function r=meanSquare(x);
    x=x(:);
    r=(x'*x)/length(x);
return;
```

trainSpeech.m

```
%trainSpeech(path)
%
%Recibe:
%   path    - Ruta del directorio que contiene los datos de entrenamiento
%
%Utiliza los archivos:
%   nInputs.txt     - Numero de caracteristicas de entrada para la red neuronal
%   speechIn.txt    - Datos de entrada para el entrenamiento de la red
%   speechOut.txt   - Salidas esperadas para cada dato de entrenamiento
%   umbrales.txt    - Numero de nodos en cada capa oculta y numero maximo
%                     de iteraciones para el entrenamiento


function trainSpeech(path);
```

```
    input=[];

    output=[];


    %Obtiene los datos de configuracion para la red neuronal

    netParams=load('data/umbrales.txt');

    n1=netParams(1);

    n2=netParams(2);

    eMax=netParams(3);


    %Si no existen datos de entrenamiento previos, llama a createData

    if(~(exist('data/speechIn.txt','file') & exist('data/speechOut.txt','file') &
exist('data/nInputs.txt','file')))

        createData(path);

    end;

    %Lee los datos de entrenamiento

    nInputs=load('data/nInputs.txt');

    input=load('data/speechIn.txt');

    output=load('data/speechOut.txt');


    %Si la red no ha sido entrenada, la inicializa

    if(~(...

            exist('data/W1.txt','file') & exist('data/W2.txt','file') &
exist('data/W3.txt','file') &...

            exist('data/B1.txt','file') & exist('data/B2.txt','file') &
exist('data/B3.txt','file')))

        init(input,output,n1,n2);

    end;


    %Determina el error cuadratico medio inicial

    errorInicial=meanSquare(output-forward(input))/meanSquare(output);


    %Inicia el entrenamiento

    fprintf('Comenzando entrenamiento de la red...\n');

    train(input,output,eMax,20);
```

```
    %Determina el error cuadratico medio final
    errorFinal=meanSquare(output-forward(input))/meanSquare(output);

    fprintf('\nSe ha completado el entrenamiento de la red.\n\tError inicial: %g,
Error despues del entrenamiento: %g\n',100*errorInicial,100*errorFinal);
return;

%r=meanSquare(x)
%Obtiene el valor cuadratico medio de la señal x
function r=meanSquare(x);
    x=x(:);
    r=(x'*x)/length(x);
return;
```

forward.m

```
%output=forward(input)
%
%Recibe:
%   input - Vector(es) de entradas para la red neuronal
%
%Utiliza los archivos:
%   Xmean.txt   - Media de los datos de entrada que se utilizaron durante el
%                 entrenamiento de la red neuronal
%   Xstd.txt    - Desviacion estandar de los datos de entrada que se utilizaron
%                 durante el entrenamiento de la red neuronal
%   Ymean.txt   - Media de los datos de salidas esperadas que se utilizaron
%                 durante el entrenamiento de la red neuronal
%   Ystd.txt    - Desviacion estandar de los datos de salidas esperadas que se
%                 utilizaron durante el entrenamiento de la red neuronal
%   W1.txt      - Pesos para la primer capa oculta de la red neuronal
%   W2.txt      - Pesos para la segunda capa oculta de la red neuronal
%   W3.txt      - Pesos para la capa de salidas de la red neuronal
%   B1.txt      - Desplazamientos para la primer capa oculta de la red neuronal
```

```
%   B2.txt      - Desplazamientos para la segunda capa oculta de la red neuronal
%   B3.txt      - Desplazamientos para la capa de salidas de la red neuronal
%
%Regresa:
%   output - Vector(es) de salidas que indican la(s) clase(s) a la(s) que
pertenece(n)
%            el (los) vector(es) de entradas


function output=forward(input);
    %Lee los pesos y desplazamientos obtenidos luego del entrenamiento
    W1=load('data/W1.txt');
    W2=load('data/W2.txt');
    W3=load('data/W3.txt');
    B1=load('data/B1.txt');
    B2=load('data/B2.txt');
    B3=load('data/B3.txt');

    %Obtiene la media y desviacion estandar observadas en las entradas y
    %salidas durante el entrenamiento
    Xmean=load('data/Xmean.txt');
    Xstd=load('data/Xstd.txt');
    Ymean=load('data/Ymean.txt');
    Ystd=load('data/Ystd.txt');

    nIn=size(input,2);
    %Normalizacion de los datos de entrada
    for(i=1:nIn)
        input(:,i)=(input(:,i)-Xmean)./Xstd;
    end;

    %Salidas de la primer capa oculta
    Z=W1*input+B1*ones(1,nIn);
    output=tanh(Z);
```

```
    %Salidas de la segunda capa oculta
    Z=W2*output+B2*ones(1,nIn);
    output=tanh(Z);

    %Salidas de la red neuronal
    output=W3*output+B3*ones(1,nIn);

    %Escalamiento de los datos de salida
    for(i=1:nIn)
        output(:,i)=output(:,i).*Ystd+Ymean;
    end;
return;
```

FIR.m

```
%h=FIR(samplingFrequency,passBandFrequency,stopBandFrequency,order,attenuation,ty
pe)
%
%Recibe:
%   samplingFrequency   - Frecuencia de muestreo de la señal a filtrar
%   passBandFrequency   - Frecuencia de la transicion de banda pasante a banda de
rechazo
%   stopBandFrequency   - Frecuencia de la transicion de banda de rechazo a banda
pasante
%   order               - Orden deseado para el filtro
%   attenuation         - Atenuacion deseada para la banda de rechazo del filtro
%   type                - 'low' o 'high', indicando si se desea un filtro pasa-
bajas o pasa-altas respectivamente
%
%Regresa:
%   h   - Respuesta al impulso para el filtro solicitado

function
h=FIR(samplingFrequency,passBandFrequency,stopBandFrequency,order,attenuation,typ
e);
    %Ajusta el orden a su siguiente valor impar
```

```
    if(mod(order,2)==1)
        N=order;
    else
        N=order+1;
    end;
    M=(N-1)/2;

    %Ajusta las frecuencias de transicion deseadas en caso de solicitarse
    %un pasa-altas
    if(strcmp(type,'high'))
        passBandFrequency=samplingFrequency/2-passBandFrequency;
        stopBandFrequency=samplingFrequency/2-stopBandFrequency;
    end;

    %Calcula las frecuencias de transicion normalizadas
    wpass=2*pi*(passBandFrequency/samplingFrequency);
    wstop=2*pi*(stopBandFrequency/samplingFrequency);
    fpass=wpass/pi;
    fstop=wstop/pi;

    %Obtiene la respuesta al impulso del filtro cuya respuesta en frecuencia
    %presente un error cuadratico medio minimo con respecto a los parametros
solicitados
    q=[fpass+attenuation*(1-fstop) fpass*sa(fpass*[1:2*M])-
attenuation*fstop*sa(fstop*[1:2*M])];
    Q1=hermitianToeplitzMatrix(q([0:M]+1));
    Q2=hankelMatrix(q([0:M]+1),q([M:2*M]+1));
    Q=(Q1+Q2)/2;
    b=fpass*sa(fpass*[0:M]');
    a=inv(Q)*b;
    h=[a(M+1:-1:2)/2;a(1);a(2:M+1)/2];

    %Cambia el signo de los coeficientes impares en caso de solicitarse un pasa-
altas
```

```
    if(strcmp(type,'high'))
        for(n=1:length(h))
            h(n)=(-1)^(n-1)*h(n);
        end;
    end;
return;

%y=sa(x)
%Computa la funcion sa(pi*x)
function y=sa(x);
    y=ones(size(x));
    i=find(x);
    y(i)=sin(pi*x(i))./(pi*x(i));
return;

%t=hermitianToeplitzMatrix(r)
%Recibe:
%   r - Tamaño deseado para la matriz
%Regresa:
%   t - Matriz cuadrada de rxr que cumple con las condiciones:
%       t=[a_ij] --> a_ij = conj(a_ji)
%       t=[a_ij] --> A a_ij = a_k, k=-r+1,...,-1,0,1,...,r-1
%                    & t(1,:) = [a_0;a_1;...;a_r-1] -->
%                      t(n,:) = [a_-n+1;t(n-1,1:end-1)], n=2,3,...,r
function t=hermitianToeplitzMatrix(r);
    p=length(r);
    r=transpose(r);
    x=[r(p:-1:2);r];
    cidx=transpose((0:p-1));
    ridx=p:-1:1;
    t=cidx(:,ones(p,1))+ridx(ones(p,1),:);
    t=x(t);
return;
```

```
%H=hankelMatrix(c,r)
%Recibe:
%   c - Primer vector columna para la matriz resultante
%   r - Ultimo vector columna para la matriz resultante
%Regresa:
%   H - Matriz cuadrada de length(c)xlength(c) que cumple con la condicion:
%       H=a_ij --> a_ij = B(i+j-1), B = [c;r(2:end)]
function H=hankelMatrix(c,r);
    r=transpose(r);
    c=transpose(c);
    nr=length(r);
    nc=length(c);
    x=[c;r((2:nr)')];
    cidx=(1:nc)';
    ridx=0:(nr-1);
    H=cidx(:,ones(nr,1))+ridx(ones(nc,1),:);
    H=x(H);
return;
```

record.m

```
%function data=record(duration,samplingFrequency)
%
%Recibe:
%   duration            - Tiempo en segundos que durara la grabacion
%   samplingFrequency   - Frecuencia de muestreo en Hertz a utilizar para
%                         la grabacion
%
%Regresa:
%   data    - Vector de muestras obtenidas desde una entrada analogica

function data=record(duration,samplingFrequency);
    %Obtiene la referencia a una entrada analogica
```

```
[realSamplingFrequency,hAudioInput]=initAudioInput(duration,samplingFrequency);

    %Comienza la grabacion
    start(hAudioInput);
    try
        %Almacena los datos adquiridos
        data=getData(hAudioInput);
        %De ser necesario ajusta la frecuencia de muestreo a la deseada
data=resample(data,round(samplingFrequency),round(realSamplingFrequency));
        %Normaliza la señal
        data=0.99*data/max(abs(data));
    catch
        %Si hubo un error, regresa -1
        data=-1;
    end;

    %Libera la entrada analogica
    stop(hAudioInput);
    delete(hAudioInput);
return;

%hAnalogInput=initAudioInput(tLength,fs)
%Obtiene y configura una entrada analogica
function [fs,hAnalogInput]=initAudioInput(tLength,fs);
    %Determina la version de Matlab
    vesion=ver;
    name={version.Name};
    mIndex=find(strcmp(name,'MATLAB'));
    if(isempty(mIndex))
        mIndex=find(strcmp(name,'MATLAB Toolbox'));
    end;
```

```
vNumber=str2num(version(nIndex).Version);

%Crea una entrada analogica asociada a la tarjeta de sonido
hAnalogInput=analoginput('winsound');

%Grabacion monoaural
addchannel(hAnalogInput,1);

if((vNumber==6.1)|(vNumber==6.5))
    %Permite el uso de frecuencias de muestreo no estandar
    set(hAnalogInput,'StandardSampleRates','Off');
end;
%Configura la frecuencia de muestreo deseada
set(hAnalogInput,'SampleRate',fs);
%Lee el valor de la frecuencia de muestreo que se utilizara
%Dependiendo del hardware, esta podria no ser la deseada
fs=get(hAnalogInput,'SampleRate');

%Configura la condicion requerida para comenzar a grabar
set(hAnalogInput,'TriggerChannel',hAnalogInput.Channel(1));
set(hAnalogInput,'TriggerType','software');
set(hAnalogInput,'TriggerCondition','Rising');
set(hAnalogInput,'TriggerConditionValue',0.01);
set(hAnalogInput,'TriggerDelay',-0.1);
set(hAnalogInput,'TriggerDelayUnits','seconds');

%Se realizara solo una grabacion
set(hAnalogInput,'TriggerRepeat',0);

%Configura el numero de muestras a obtener
set(hAnalogInput,'SamplesPerTrigger',fs*tLength);

%Tiempo maximo de silencio antes de indicar un error
```

```
    set(hAnalogInput,'TimeOut',15);
return;
```

interfaz.m

```
%Funciones creadas por la herramienta GUIDE de matlab
function varargout = interfaz(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @interfaz_OpeningFcn, ...
                   'gui_OutputFcn',  @interfaz_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


function interfaz_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);


function varargout = interfaz_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;


% pushbutton1_Callback(hObject, eventdata, handles)
% Guarda el archivo de audio
function pushbutton1_Callback(hObject, eventdata, handles)
```

```
    %Abre ventana
    [filename,pathname]=uiputfile('*.wav','Ingresa el nombre de archivo');
    %Guarda el archivo
    sound=load('data/ui/sound.txt');
    if(filename~=0)
       wavwrite(sound,8000,[pathname filename]);
   end;
return;

% pushbutton2_Callback(hObject, eventdata, handles)
% Realiza la grabacion de audio.
function pushbutton2_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    %Obtiene la duracion
    duration=get(handles.edit2,'String');
    %Guarda el archivo de audio en una variable
    sound=record(str2num(duration),8000);
    if sound==-1
        errordlg('Error en deteccion de sonidos','Error','modal');
    end
    %Guarda la grabacion en un archivo de texto
    save data/ui/sound.txt sound -ascii;
return;

% pushbutton3_Callback(hObject, eventdata, handles)
% Elige un directorio
function pushbutton3_Callback(hObject, eventdata, handles)
    %Abre una ventana
    dname = uigetdir('C:\');
    %Guarda en un archivo de texto la ruta
    save data/ui/dname.txt dname -ascii;
return;
```

```
% pushbutton4_Callback(hObject, eventdata, handles)
% Inicia el entrenamiento de la red neuronal artificial
function pushbutton4_Callback(hObject, eventdata, handles)
     dname=load('data/ui/dname.txt');
     cd data;
     delete *.txt;
     cd ..;

     if(~exist('data/frameDuration.txt','file') |
~exist('data/windowDuration.txt','file') | ~exist('data/method.txt','file') |
~exist('data/supportThreshold.txt','file'))
         signalParams;
     end;

    if(~exist('data/umbrales.txt','file'))
        initThresholds;
    end;

     %Carga archivos de texto a variables
     frameDuration=load('data/frameDuration.txt');
     windowDuration=load('data/windowDuration.txt');
     method=char(load('data/method.txt'));
     supportThreshold=load('data/supportThreshold.txt');

     trainSpeech(dname);
 return;

% pushbutton5_Callback(hObject, eventdata, handles)
% Selecciona un archivo con extension .wav
function pushbutton5_Callback(hObject, eventdata, handles)
    %Guarda la ruta y el nombre del archivo
    [filename,pathname] = uigetfile('*.wav','Select wav file');
    file=[pathname filename];
```

```
    %Obtiene parametros del archivo de audio
    [data,samplingFrequency,nBits]=wavRead(file);
    %Guarda las variables en archivos de texto
    save data/ui/data.txt data -ascii;
    save data/ui/samplingFrequency.txt samplingFrequency -ascii;
    save data/ui/nBits.txt nBits -ascii;
return;


% pushbutton6_Callback(hObject, eventdata, handles)
% Inicia el reconocimiento de un archivo de audio
function pushbutton6_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);


    %Si no existen los archivos de configuracion, los genera con valores por
defecto
    if(~exist('data/frameDuration.txt','file') |
~exist('data/windowDuration.txt','file') | ~exist('data/method.txt','file') |
~exist('data/supportThreshold.txt','file'))
        signalParams;
    end;
    if(~exist('data/umbrales.txt','file'))
        initThresholds;
    end;


    %Si la red neuronal no ha sido entrenada, llama a trainSpeech
    if(~(exist('data/W1.txt','file') & exist('data/W2.txt','file') &
exist('data/W3.txt','file')...
            & exist('data/B1.txt','file') & exist('data/B2.txt','file') &
exist('data/B3.txt','file') & exist('data/nInputs.txt','file')))
        trainSpeech('audio/trainingData');
    end;


    %Obtiene los parametros para la segmentacion
    frameDuration=load('data/frameDuration.txt');
    windowDuration=load('data/windowDuration.txt');
```

```
    %Obtiene los parametros para la extraccion de caracteristicas
    method=char(load('data/method.txt'));
    supportThreshold=load('data/supportThreshold.txt');
    %Obtiene los parametros para el reconocimiento
    nInputs=load('data/nInputs.txt');
    thresholds=load('data/umbrales.txt');
    activationTH=thresholds(10);
    differentiationTH=thresholds(11);

    data=load('data/ui/data.txt');
    samplingFrequency=load('data/ui/samplingFrequency.txt');
    nBits=load('data/ui/nBits.txt');

    %Acondiciona la señal e identifica sus tramas vocalizadas
    [data,bVoiced]=isVoiced(data,samplingFrequency);

    %Guarda en un archivo de texto la variable data
    save data/ui/fData.txt data -ascii;

    %Genera un eje de tiempo para las componentes vocalizadas
    tVoiced=1:length(data);
    for(i=1:length(bVoiced))
        if(~bVoiced(i))
            tVoiced((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
        end;
    end;

    %Inicializa las variables para la generacion de la cadena resultante
    vowels=repmat('_',size(bVoiced));
    code=['a' 'e' 'i' 'o' 'u'];
```

```
    %Obtiene vectores de parametros para cada trama de la señal

featureVector=featureExtraction(data,samplingFrequency,frameDuration,windowDurati
on,supportThreshold,nInputs,method,0,tVoiced);


    for(i=1:length(bVoiced))
        if(bVoiced(i) & featureVector(:,i)~=repmat(-1,nInputs,1))
            %Solo se busca un fonema para las tramas vocalizadas
            pattern=[(1:length(code))' forward(featureVector(:,i))];
            %Se buscan los dos valores mas grandes
            pattern=flipud(sortrows(pattern,2));
            if(~(pattern(1,2)<activationTH | pattern(2,2)>differentiationTH))
                %Si el mayor excede el umbral de activacion y el siguiente no
excede el umbral de diferenciacion,
                %se agrega el simbolo correspondiente al fonema
                vowels(i)=code(pattern(1,1));
            end;
        end;
    end;

    %Guarda en un archivo de texto la variable vowels
    save data/ui/vowels.txt vowels -ascii;

    %Se empleara un filtrado modal con una ventana de 10 simbolos
    filterSize=10;

    %Elimina tramas erroneamente clasificadas como no vocalizadas
    vowels=lowPass(vowels,'_',filterSize-4);

    %Busca solo las componentes vocalizadas
    vowelsIndex=find(vowels~='_');
    if(isempty(vowelsIndex))
        %Si no existen, termina
        word='_';
```

```
        return;
    end;

    %Busca los limites entre grupos de vocales consecutivas
    vowelsLimits=[0 find(vowelsIndex(1:end-1)~=vowelsIndex(2:end)-1)
length(vowelsIndex)];

    %Determina la longitud del grupo de vocales mas largo
    maxLength=max(vowelsLimits(2:end)-vowelsLimits(1:end-1));

    %Genera una matriz con los grupos de vocales ordenados por filas
    prevLimit=vowelsLimits(1);
    vowelsGroups=[];
    for(i=2:length(vowelsLimits))

currentGroup=vowels(vowelsIndex(prevLimit+1):vowelsIndex(vowelsLimits(i)));
        prevLimit=vowelsLimits(i);
        %Si el grupo es menor a la longitud maxima, agrega simbolos 'x' al final
        vowelsGroups=[vowelsGroups;[currentGroup repmat('x',1,maxLength-
length(currentGroup))]];
    end;

    %Realiza un filtrado modal de cada grupo de vocales
    vowels=[];
    for(i=1:length(vowelsGroups(:,1)))
        %Se eliminan los simbolos que pudieron ser agregados
        currentS=vowelsGroups(i,find(vowelsGroups(i,:)~='x'));
        filteredS=[];
        %Si el grupo es menor al tamaño del filtro, se obtiene su moda
        if(length(currentS)<=filterSize)
            filteredS=findMode(currentS);
            if(length(filteredS)>1)
                filteredS=filteredS(1);
            end;
```

```
        else
            %De lo contrario, se obtiene la moda de cada grupo de filterSize
simbolos
            for(i=ceil(filterSize/2):length(currentS)-floor(filterSize/2))
                currentVowel=findMode(currentS(i-floor((filterSize-
1)/2):i+ceil((filterSize-1)/2)));
                if(length(currentVowel)>1)
                    %Si mas de un simbolo comparten el numero maximo de
apariciones se toma el mas alejado de los bordes
                    if(i<length(currentS)/2)
                        currentVowel=currentVowel(end);
                    else
                        currentVowel=currentVowel(1);
                    end;
                end;
                filteredS=[filteredS currentVowel];
            end;
        end;
        %El resultado se agrega a la cadena resultante
        vowels=[vowels '_' filteredS '_'];
    end;

    %Se eliminan los simbolos identicos a su predecesor
    word=vowels(1);
    for(i=2:length(vowels))
        if(vowels(i)~=word(end))
            word=[word vowels(i)];
        end;
    end;

    %Guarda en archivos de texto variables
    save data/ui/tVoiced.txt tVoiced -ascii;
    save data/ui/featureVector.txt featureVector -ascii;
```

```
    %Modifica el contenido del cuadro de texto text4
    set(handles.text4,'String',word);
return;

% listbox1_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto listbox1
function listbox1_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% listbox1_Callback(hObject, eventdata, handles)
% Se ejecuta cuando existe un cambio en listbox1
function listbox1_Callback(hObject, eventdata, handles)

% pushbutton7_Callback(hObject, eventdata, handles)
% Genera las graficas
function pushbutton7_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);

    %Carga archivos de texto a variables
    frameDuration=load('data/frameDuration.txt');
    samplingFrequency=load('data/ui/samplingFrequency.txt');
    data=load('data/ui/data.txt');
    fData=load('data/ui/fData.txt');
    tVoiced=load('data/ui/tVoiced.txt');
    featureVector=load('data/ui/featureVector.txt');

    %Obtiene el valor del listbox1
    value=get(handles.listbox1,'Value');
```

```
    if value==1
    %grafica la señal original
        if ishold
            hold;
        end

time=1/samplingFrequency:1/samplingFrequency:length(data)/samplingFrequency;
        plot(time,data,'Color', [ 0.502, 0.502, 0.753 ] );
        xlabel('Tiempo');
        ylabel('Amplitud');
        axis([0 length(data)/samplingFrequency min(data) max(data)]);
    elseif value==2
    %grafica la señal despues del preprocesamiento
        if ishold
            hold;
        end

time=1/samplingFrequency:1/samplingFrequency:length(fData)/samplingFrequency;
        plot(time,fData,'Color', [ 0.502, 0.502, 0.753 ] );
        xlabel('Tiempo');
        ylabel('Amplitud');
        axis([0 length(fData)/samplingFrequency min(fData) max(fData)]);
    elseif value==3
    %grafica la señal que muestra segmentos vocalizados y no vocalizados
        if ishold
            hold;
        end


time=1/samplingFrequency:1/samplingFrequency:length(fData)/samplingFrequency;
        dataVoiced=fData;
        dataVoiced(find(tVoiced==0))=0;
        dataUnvoiced=fData;
        dataUnvoiced(find(tVoiced~=0))=0;
```

```
        plot(time,dataVoiced,'Color', [ 0.643, 1, 0.4 ] );
        xlabel('Tiempo');
        ylabel('Amplitud');
        if ~ishold
            hold;
        end
        plot(time,dataUnvoiced,'Color', [ 0.502, 0.502, 0.753 ] );
        axis([0 length(fData)/samplingFrequency min(fData) max(fData)]);
    elseif value==4
    %grafica la transformada wavelet de la señal
        if ishold
            hold;
        end
        data=fData;
        fs=samplingFrequency;
        waveletDecomposition=[];

n=floor(7*log(samplingFrequency/1300+sqrt((samplingFrequency/1300).^2+1)));
        for(Z=1:n)
            fc=325*(exp(2*Z/7)-1)./exp(Z/7);

waveletDecomposition=[waveletDecomposition;earDWT(data,fs,fc,3,0.03,0.001,1)'];
        end


        if(length(waveletDecomposition(:,1))<n)
           waveletDecomposition=[waveletDecomposition;zeros(n-
length(waveletDecomposition(:,1)),length(waveletDecomposition(1,:)))];
        end
        waveletDecomposition=abs(flipud(waveletDecomposition));
        waveletDecomposition=flipud(waveletDecomposition);
        cm=colormap('jet');
        cm=sqrt(cm(:,1).^2+cm(:,2).^2+cm(:,3).^2)';
        cm=cm/max(cm);
```

```
        ampSC=min(min(waveletDecomposition)):(max(max(waveletDecomposition))-
min(min(waveletDecomposition)))/(length(cm)-1):max(max(waveletDecomposition));
        imagesc(ampSC,0,cm);
        imagesc(waveletDecomposition);
        xlabel('Traslacion');
        ylabel('Escala');
        set(gca,'YDir','normal');

    elseif value==5
    %grafica el vector de parametros
        if ishold
            hold;
        end
        r=0;
        g=0;
        b=0;
        for i=1:length(featureVector(:,1))
            r=r+.015;
            g=g+.010;
            b=b+.025;
            plot(featureVector(i,:),'Color',[r,g,b]);
            if ~ishold
                hold;
            end
        end
        axis tight;
        xlabel('Vector');
        ylabel('Intensidad');
    else
    %grafica los componentes vocalicos de la señal
        if ishold
            hold;
        end
```

```
        vowels=load('data/ui/vowels.txt');

time=1/samplingFrequency:1/samplingFrequency:length(fData)/samplingFrequency;

        %Inicia filtrado de la cadena de caracteres a la salida de la red
        %neuronal

        %Se empleara un filtrado modal con una ventana de 10 simbolos
        filterSize=10;

        %Busca solo las componentes vocalizadas
        vowelsIndex=find(vowels~='_');
        if(isempty(vowelsIndex))
            %Si no existen, termina
            word='_';
            return;
        end;

        %Busca los limites entre grupos de vocales consecutivas
        vowelsLimits=[0 find(vowelsIndex(1:end-1)~=vowelsIndex(2:end)-1)
length(vowelsIndex)];

        %Determina la longitud del grupo de vocales mas largo
        maxLength=max(vowelsLimits(2:end)-vowelsLimits(1:end-1));

        %Genera una matriz con los grupos de vocales ordenados por filas
        prevLimit=vowelsLimits(1);
        vowelsGroups=[];
        for(i=2:length(vowelsLimits))

currentGroup=vowels(vowelsIndex(prevLimit+1):vowelsIndex(vowelsLimits(i)));
            prevLimit=vowelsLimits(i);
            %Si el grupo es menor a la longitud maxima, agrega simbolos 'x' al
final
```

```
            vowelsGroups=[vowelsGroups;[currentGroup repmat('x',1,maxLength-
length(currentGroup))]];
        end;

        %Realiza un filtrado modal de cada grupo de vocales
        vowels=[];
        for(i=1:length(vowelsGroups(:,1)))
            %Se eliminan los simbolos que pudieron ser agregados
            currentS=vowelsGroups(i,find(vowelsGroups(i,:)~='x'));
            filteredS=[];
            %Si el grupo es menor al tamaño del filtro, se obtiene su moda
            if(length(currentS)<=filterSize)
                filteredS=findMode(currentS);
                if(length(filteredS)>1)
                    filteredS=filteredS(1);
                end;
            else
                %De lo contrario, se obtiene la moda de cada grupo de filterSize
simbolos
                for(i=ceil(filterSize/2):length(currentS)-floor(filterSize/2))
                    currentVowel=findMode(currentS(i-floor((filterSize-
1)/2):i+ceil((filterSize-1)/2)));
                    if(length(currentVowel)>1)
                        %Si mas de un simbolo comparten el numero maximo de
apariciones se toma el mas alejado de los bordes
                        if(i<length(currentS)/2)
                            currentVowel=currentVowel(end);
                        else
                            currentVowel=currentVowel(1);
                        end;
                    end;
                    filteredS=[filteredS currentVowel];
                end;
            end;
```

```
            %El resultado se agrega a la cadena resultante
            vowels=[vowels '_' filteredS '_'];
        end;

        %Se eliminan los simbolos identicos a su predecesor
        word=vowels(1);
        for(i=2:length(vowels))
            if(vowels(i)~=word(end))
                word=[word vowels(i)];
            end;
        end;

        %Carga el archivo vowels.txt a la variable vowels
        vowels=load('data/ui/vowels.txt');

        tFilteredVowels=word(find(word~='_'));
        prevLimit=vowelsLimits(1);
        for(i=2:length(vowelsLimits))
vowels(vowelsIndex(prevLimit+1):vowelsIndex(vowelsLimits(i)))=tFilteredVowels(i-
1);
            prevLimit=vowelsLimits(i);
        end;

        vowelA=fData;
        vowelE=fData;
        vowelI=fData;
        vowelO=fData;
        vowelU=fData;
        plot(time,fData,'Color', [ 1, 1, 0.502 ] );
        if ~ishold
            hold;
        end
```

```
        for(i=1:length(vowels))
            if(vowels(i)~='a')
                vowelA((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
            end
        end
        plot(time,vowelA,'Color', [ .702, .702, 0.851 ] );
        if ~ishold
            hold;
        end
        for(i=1:length(vowels))
            if(vowels(i)~='e')
                vowelE((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
            end
        end
        plot(time,vowelE,'Color', [ .452, .452, 0.725 ]  );
        if ~ishold
            hold;
        end
        for(i=1:length(vowels))
            if(vowels(i)~='i')
                vowelI((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;
            end
        end
        plot(time,vowelI,'Color', [ .251, .251, 0.502 ]  );
        if ~ishold
            hold;
        end
        for(i=1:length(vowels))
            if(vowels(i)~='o')
```

```
                vowelO((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;

            end

        end

        plot(time,vowelO,'Color', [ .145, .145, 0.286 ]  );

        if ~ishold

            hold;

        end

        for(i=1:length(vowels))

            if(vowels(i)~='u')

                vowelU((i-
1)*round(frameDuration*samplingFrequency)+1:i*round(frameDuration*samplingFrequen
cy))=0;

            end

        end

        plot(time,vowelU,'Color', [ 0, 0, 0 ]  );

        if ~ishold

            hold;

        end

        axis tight;

        xlabel('Tiempo');

        ylabel('Amplitud');

    end

return;


% pushbutton8_Callback(hObject, eventdata, handles)

% Reproduce el archivo de audio

function pushbutton8_Callback(hObject, eventdata, handles)

    %Carga archivos de texto en variables

    data=load('data/ui/data.txt');

    samplingFrequency=load('data/ui/samplingFrequency.txt');

    nBits=load('data/ui/nBits.txt');

    %Reproduce los datos
```

```
    sound(data,samplingFrequency,nBits);
return;

% edit2_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto listbox1
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% edit2_Callback(hObject, eventdata, handles)
% Se ejecuta cuando cambia el objeto edit2
function edit2_Callback(hObject, eventdata, handles)

%dataMode=findMode(data)
%Obtiene la moda del conjunto data
function dataMode=findMode(data);
    %Agrupa simbolos iguales
    sortedData=[];
    for(i=1:length(data))
        if(isempty(find(sortedData==data(i))))
            %Si es la primera aparicion del simbolo, agrega su clase
            sortedData=[sortedData;data(i) 1];
        else
            %De lo contrario, incrementa la cuenta del simbolo
            [x,y]=find(sortedData==data(i));
            sortedData(x,2)=sortedData(x,2)+1;
        end;
    end;
    %Busca el (los) simbolo(s) con un mayor numero de apariciones
    dataMode=sortedData(find(sortedData(:,2)==max(sortedData(:,2))),1);
return;
```

```
%s=lowPass(data,symbol,windowSize)
%Filtrado modal de las apariciones de symbol dentro del vector data
function s=lowPass(data,symbol,windowSize);
    s=data;
    %Busca las apariciones de symbol
    sIndex=find(data==symbol);
    for(i=1:length(sIndex))
        %Filtrado modal con una ventana de windowSize elementos
        if(length(find(data(max(1,sIndex(i)-floor((windowSize-
1)/2)):min(length(data),sIndex(i)+ceil((windowSize-
1)/2)))~=symbol))>=ceil(windowSize/2))
            s(sIndex(i))=0;
        end;
    end;
    %Elimina de la cadena final los simbolos indeseados
    s=s(find(s~=0));
return;
```

configuracion.m

```
%Funciones creadas por la herramienta GUIDE de matlab
function varargout = configuracion(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @configuracion_OpeningFcn, ...
                   'gui_OutputFcn',  @configuracion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


function configuracion_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);


function varargout = configuracion_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;


% slider1_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider1 y guarda los valores por
% defecto de cada uno de los criterios
function slider1_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
% Localiza a slider1 en el valor 18
set(handles.slider1,'Value',18);
x=zeros(11,1);
x(1)=54;
x(2)=10;
x(3)=5000;
x(4)=0.05;
x(5)=0.33;
x(6)=0.17;
x(7)=0.33;
```

```
x(8)=1200;
x(9)=5000;
x(10)=0.5;
x(11)=0.33;
%Guarda las variables en un archivo de texto
save data/umbrales.txt x -ascii;


% slider1_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider1 y guarda ese valor en el archivo de texto
function slider1_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider1,'Value');
    value=round(value)+36;
    set(handles.text8,'String',value);
    x=load('data/umbrales.txt');
    x(1)=value;
    save data/umbrales.txt x -ascii;


% slider2_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider2
function slider2_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider2,'Value',0);


% slider2_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider2 y guarda ese valor en el archivo de texto
function slider2_Callback(hObject, eventdata, handles)
```

```
    handles=guihandles(hObject);
    value=get(handles.slider2,'Value');
    value=round(value)+10;
    set(handles.text10,'String',value);
    x=load('data/umbrales.txt');
    x(2)=value;
    save data/umbrales.txt x -ascii;

% slider3_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider3
function slider3_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider3,'Value',4900);

% slider3_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider3 y guarda ese valor en el archivo de texto
function slider3_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider3,'Value');
    value=round(value)+100;
    set(handles.text13,'String',value);
    x=load('data/umbrales.txt');
    x(3)=value;
    save data/umbrales.txt x -ascii;

% slider5_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider5
```

```
function slider5_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider5,'Value',0.05);

% slider5_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider5 y guarda ese valor en el archivo de texto
function slider5_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider5,'Value');
    set(handles.text18,'String',value);
    x=load('data/umbrales.txt');
    x(4)=value;
    save data/umbrales.txt x -ascii;

% slider6_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider6
function slider6_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider6,'Value',0.03);

% slider6_Callback(hObject, eventdata, handles)
```

```
% Actualiza el valor del slider6 y guarda ese valor en el archivo de texto
function slider6_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider6,'Value');
    value=value+0.3;
    set(handles.text20,'String',value);
    x=load('data/umbrales.txt');
    x(5)=value;
    save data/umbrales.txt x -ascii;

% slider7_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider7
function slider7_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider7,'Value',0.07);

% slider7_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider7 y guarda ese valor en el archivo de texto
function slider7_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider7,'Value');
    value=value+0.1;
    set(handles.text22,'String',value);
    x=load('data/umbrales.txt');
    x(6)=value;
    save data/umbrales.txt x -ascii;
```

```
% slider8_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider8
function slider8_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider8,'Value',0.13);

% slider8_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider8 y guarda ese valor en el archivo de texto
function slider8_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider8,'Value');
    value=value+0.2;
    set(handles.text24,'String',value);
    x=load('data/umbrales.txt');
    x(7)=value;
    save data/umbrales.txt x -ascii;

% slider12_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider12
function slider12_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
```

```
set(handles.slider12,'Value',900);


% slider12_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider12 y guarda ese valor en el archivo de texto
function slider12_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider12,'Value');
    value=round(value)+300;
    set(handles.text35,'String',value);
    x=load('data/umbrales.txt');
    x(8)=value;
    save data/umbrales.txt x -ascii;


% slider13_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider13
function slider13_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider13,'Value',3500);


% slider13_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider13 y guarda ese valor en el archivo de texto
function slider13_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider13,'Value');
    value=round(value)+1500;
    set(handles.text37,'String',value);
    x=load('data/umbrales.txt');
```

```
    x(9)=value;
    save data/umbrales.txt x -ascii;

% slider16_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider16
function slider16_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider16,'Value',0.3);

% slider16_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider16 y guarda ese valor en el archivo de texto
function slider16_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider16,'Value');
    value=value+0.2;
    set(handles.text44,'String',value);
    x=load('data/umbrales.txt');
    x(10)=value;
    save data/umbrales.txt x -ascii;

% slider17_CreateFcn(hObject, eventdata, handles)
% Establece las propiedades del objeto slider17
function slider17_CreateFcn(hObject, eventdata, handles)
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
```

```
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
handles=guihandles(hObject);
set(handles.slider17,'Value',0.23);


% slider17_Callback(hObject, eventdata, handles)
% Actualiza el valor del slider17 y guarda ese valor en el archivo de texto
function slider17_Callback(hObject, eventdata, handles)
    handles=guihandles(hObject);
    value=get(handles.slider17,'Value');
    value=value+0.1;
    set(handles.text46,'String',value);
    x=load('data/umbrales.txt');
    x(11)=value;
    save data/umbrales.txt x -ascii;
```

Tabla de constantes utilizadas en el modelo del oído de Zhang, et.al.

Parameters	Description	Values
Basilar membrane tuning filter and control path		
ω_{CF}	characteristic frequency of the fiber (rad/s)	
a	delay of the onset tone responses for cat (s)	See Eq. (3)
A_D	coefficient for traveling wave delay (ms)	8.13
A_L	length constant for traveling wave delay (mm)	6.49
x_{CF}	distance from apex of basilar membrane (mm)	
$\tau(t)$	output of the control path	
τ_{narrow}	estimated time constant at low sound level	See Eq. (4)
τ_{wide}	estimated time constant at high sound level	See Eq. (6)
γ_{cp}	order of the wide bandpass filter in control path	3
ω_{cp}	center frequency of the wide bandpass filter	1.2 mm basal to fiber CF
K	ratio of time constant in control path to that in signal path	$0.2+0.8\ \tau_{wide}/\tau_{narrow}$
A_{cp}	parameter in logarithmic nonlinearity	970
B_{cp}	parameter in logarithmic nonlinearity	2.75
C_{cp}	parameter in logarithmic nonlinearity	0.69
$x0_{cp}$	parameter in Boltzman function	7.6
$s0_{cp}$	parameter in Boltzman function	12
$x1_{cp}$	parameter in Boltzman function	5
$s1_{cp}$	parameter in Boltzman function	5
shift_{cp}	parameter in Boltzman function	0.125
cut_{cp}	cutoff frequency of control-path low-pass filter (Hz)	800
k_{cp}	order of control-path low-pass filter	3
dc	estimated dc shift of CP low-pass filter output at high level	0.37
R_0	ratio of τ_{LB} (lower bound of τ_{LP}) to τ_{narrow} [see Eq. (13)]	0.05
Inner hair cell model		
A_{ihc0}	scalar in IHC nonlinear function [see Eq. (16)]	0.1
B_{ihc}	parameter in IHC nonlinear function [see Eq. (15)]	2000
C_{ihc}	parameter in IHC nonlinear function [see Eq. (16)]	1.74
D_{ihc}	parameter in IHC nonlinear function [see Eq. (16)]	$6.87e$-9
cut_{ihc}	cutoff frequency of IHC low-pass filter (Hz)	3800
k_{ihc}	order of IHC low-pass filter	7
p_1	parameter in V_{ihc} rectifying function	0.0143
p_2	parameter in V_{ihc} rectifying function	See Eq. (18)
Synapse		
$spont$	spontaneous rate of fiber (spikes/s)	50
A_{ss}	steady state rate (spikes/s)	350
τ_{ST}	short-term time constant (ms)	60
τ_R	rapid time constant (ms)	2
$A_{R/ST}$	rapid response amplitude to short-term response amplitude ratio	6
PTS	peak to steady state ratio	3.6
$P_{1\,max}$	permeability at high sound level	0.6
Spike generator and refractoriness		
$c0$	parameter for relative refractoriness	0.5
$c1$	parameter for relative refractoriness	0.5
$s0$	parameter for relative refractoriness (ms)	1.0
$s1$	parameter for relative refractoriness (ms)	12.5
R_A	absolute refractory period (ms)	0.75

Bibliografía

[1] Howard, David, et.al., "ACOUSTICS AND PSYCHOACOUSTICS". Ed: Focal Press. Second Edition, 2001

[2] Xuedong Zhang, et.al., "A PHENOMENOLOGICAL MODEL FOR THE RESPONSES OF AUDITORY-NERVE FIBERS: I. NONLINEAR TUNING WITH COMPRESSION AND SUPPRESSION", Acoustical Society of America, 2001.

[3] Greenwood, Mark, et.al., "SUVing: AUTOMATIC SILENCE/UNVOICED/VOICED CLASSIFICATION OF SPEECH", Department of Computer Science, University of Sheffield. (http://plaza.ufl.edu/dongsoo7/speech1.pdf)

[4] Schroeder MR, et.al., "OBJECTIVE MEASURE OF CERTAIN SPEECH SIGNAL DEGRADATIONS BASED ON MASKING PROPERTIES OF THE HUMAN AUDITORY PERCEPTION", Frontiers of Speech Communication Research, Academic Press, 1979.

[5] http://mathworld.wolfram.com/GammaDistribution.html

[6] http://mathworld.wolfram.com/GammaFunction.html

[7] Chapman, Stephen. "MATLAB PROGRAMMING FOR ENGINEERS", Ed: Thomson Learning. Second Edition.

[8] Courant & Hilbert, "METHODS OF MATHEMATICAL PHYSICS VOLUME I", Ed: Wiley-Interscience, 1989.

[9] Freeman, James, et.al., "NEURAL NETWORKS, ALGORITHMS, APPLICATIONS, AND PROGRAMMING TECHNIQUES", Ed: Addison-Wesley.

[10] Guyton, Arthur, et.al. "TRATADO DE FISIOLOGÍA MÉDICA", Ed: McGraw Hill.

[11] Haykin, Simon, "NEURAL NETWORKS, A COMPREHENSIVE FOUNDATION", Ed: Prentice Hall.

[12] Lokenath Debnath, "WAVELET TRANSFORMS & TIME-FREQUENCY SIGNAL ANALYSIS", Ed: Birkhäuser, 2001.

[13] Tierney, Lawrence, et.al., "CURRENT MEDICAL DIAGNOSIS & TREATMENT", Ed: Mc Graw Hill.

[14] Walker, James S., "A PRIMER ON WAVELETS AND THEIR SCIENTIFIC APPLICATIONS", Ed: Chapman & Hall/CRC, 1999.

[15] Ziemer, Rodger E., et.al., "SIGNALS AND SYSTEMS: CONTINUOUS AND DISCRETE", Ed: Prentice Hall, 1998.

[16] Hynek Hermansky, "STOCHASTIC TECHNIQUES IN DERIVING PERCEPTUAL KNOWLEDGE", IDIAP Research Institute.
(http://journal.speech.cs.cmu.edu/SAPA2004/papers/136.pdf)

[17] Mestre, Antonio Ríos, "FONEMAS Y ALÓFONOS DEL ESPAÑOL", La Trascripción Fonética Automática del Diccionario Electrónico de Formas Simples Flexivas del Español: Estudio Fonológico en el Léxico. Departamento de Filología Española, Universitad Autónoma de Barcelona, 1999. (http://elies.rediris.es/elies4/Cap4.htm)

[18] Scherer, Ronald, "A BASIC OVERVIEW OF VOICE PRODUCTION"
(www.voicefoundation.org/library/voiceproduction.pdf)

[19] Schniter, Phil, "DISCRETE WAVELET TRANSFORM: MAIN CONCEPTS", (http://cnx.rice.edu/content/m10436/latest/)

[20] Smith, Julius O. III, "BARK AND ERB BILINEAR TRANSFORMS", Center for Computer Research in Music and Acoustics, Stanford University. (http://ccrma.stanford.edu/~jos/bbt/bbt.html)

[22] Stergiou, Christos; Siganos Dimitrios, "NEURAL NETWORKS" (http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html)

[23] http://cslu.cse.ogi.edu/tutordemos/SpectrogramReading/ipa/formants.html

[24] http://hyperphysics.phy-astr.gsu.edu/hbase/sound/timbre.html

[25] http://www.cs.usyd.edu.au/~irena/ai01/nn/4.html

[26] http://www.hains.net/communication/studying.html

[27] http://www.health.state.ny.us/nysdoh/antibiotic/ear.gif

[28] http://www.larynxlink.com/Library/faqs/FAQ16.htm

[29] http://www.neoxi.com/NNR/

[30] http://www.voiceproblem.org/anatomy/understanding.asp