ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

PROCESADO DIGITAL DE VOZ PARA EL RECONOCIMIENTO DEL HABLANTE APLICADO A DISPOSITIVOS MÓVILES

Daniel Moral Bárcena

Jesús Villadangos Alonso

Pamplona, Fecha de defensa

Resumen

Este proyecto hace un estudio sobre la posibilidad de implantar una aplicación de autentificación mediante datos biométricos en dispositivos móviles. Se ha realizado un breve repaso a los métodos de reconocimiento biométrico existentes en la actualidad, así como a las posibilidades que nos ofrecen tablets y smartphones en esta área. Se ha optado por utilizar el micrófono que incorporan los dispositivos para recoger y analizar la señal digital de voz del usuario, utilizando esta señal como identificador biométrico. Se ha desarrollado un sistema completo de verificación del locutor con la herramienta informática Matlab. El sistema se puede dividir en dos bloques. El primero procesa la señal de audio en varias fases: acondicionamiento de señal, enventanado, segmentado de tramas sonoras y extracción de formantes entre otras operaciones. Se han utilizado dos tipos de coeficientes para caracterizar la voz, por un lado coeficientes LPCC que se basan en filtros de predicción lineal; y por otro coeficientes MFCC que utilizan la transformada de Fourier y la transformada discreta del coseno. El segundo bloque se encarga de decidir si los anteriores coeficientes pertenecen a un determinado locutor o no. Para ello se ha utilizado un algoritmo de alineamiento temporal dinámico con el propósito de salvar los desajustes temporales entre repeticiones y calcular las diferencias entre coeficientes. Estas diferencias serán las que permitan discernir si un usuario es auténtico o se trata de un impostor. Se ha obtenido el valor óptimo para varios parámetros en la etapa de procesado de audio. Se ha propuesto un sistema de entrenamiento del modelo del locutor, se han sacado conclusiones sobre los resultados obtenidos y acerca de la posibilidad real de desarrollar una aplicación comercial de este tipo. Por último se sugieren nuevos métodos de análisis y líneas futuras de investigación.

Abstract

This project aims to do a study on the possibility of implementing an authentication application using biometrics in mobile devices. It has been carried out a brief review of existing methods of biometric recognition today and the possibilities offered by tablets and smartphones in this area. It was decided to use the built-in microphone all devices to collect and analyze the digital voice signal of the user and use this signal as a biometric identifier. It has developed a complete system of speaker verification with Matlab software tool. The system can be divided in two blocks. The first, processes audio signal in several stages: signal conditioning, windowing, segmenting voiced frames and formant extracting among other operations. Have been used two types of coefficients to characterize the speech. For side LPCC coefficients based on linear prediction filters, and secondly MFCC coefficients using the Fourier Transform and the Discrete Cosine Transform. The second block is responsible for deciding if the above coefficients belong to a particular speaker or not. For this we used a dynamic time alignment algorithm for the purpose of saving temporary mismatches between replicates and calculate the differences between coefficients. These differences are those that allow to know if a user is genuine or an impostor. It has obtained the optimal value for several parameters in processing audio step. Has been proposed a speaker model training system, have been obtained conclusions about the results and about the real possibility of developing a commercial application of this type. Finally new analytical methods and future research are suggested.

Palabras clave

Verificación del locutor, MFCC, LPCC, DTW, FFT, DCT, SIFT, CMVN, coeficientes delta, predicción lineal, formantes, ZCR, segmentado de tramas sonoras, escala Mel, banco de filtros triangulares.

Agradecimientos

A mis padres por su paciencia y apoyo, por haberme inculcado los valores del sacrificio y la superación, por haber realizado un gran esfuerzo económico todos estos años.

A la Universidad Pública de Navarra por haberme proporcionado los conocimientos en comunicaciones, programación y procesado de señal.

Al tutor de este proyecto, Jesús Villadangos, por sus consejos y ayuda.

Al profesor Miroslav Zivanovic por haberme transmitido el interés en el procesado digital de audio.

A todas las grandes personas que he podido conocer y que han colaborado en mi autoconocimiento durante esta etapa en Pamplona, con las que he compartido experiencias, buenos y malos ratos.

A los compañeros de residencia, de piso, a la gente del camino. Sin vosotros no sería quien soy. Gracias a todos.

Índice

1. INTRODUCCIÓN 1
1.1. Motivación 1
1.2. Control de acceso 2
1.2.1. Parámetros biométricos 3
1.3. Procesado digital de audio 5
1.4. Objetivo del proyecto 6
1.4.1. Definiciones 7
1.- Etapas de un sistema RAL: entrenamiento y test 7
2.- Identificación vs verificación 7
3.- Texto dependiente vs texto independiente 8
4.- Medidas de robustez y comportamiento del sistema: curvas ROC 8
1.4.2. Consideraciones acerca del entorno de utilización de los dispositivos móviles 11
1.4.3. Aparato fonador humano 12
1.- Cavidades infraglóticas 12
2.- Cavidad glótica 13
3.- Cavidades supraglóticas 13
1.4.4. Antecedentes en verificación del locutor 15
1.5. Hardware y Software empleado 19
2. EXTRACCIÓN DE PARÁMETROS PARA LA CARACTERIZACIÓN 20
2.1. Adquisición de la señal 20
2.2. Acondicionamiento de la señal 21
2.2.1. Quitar componente continua 21
2.2.2. Filtrado paso alto 21
2.2.3. Filtrado de pre-énfasis 22
2.2.4. Enventanado 24
2.2.5. Segmentado de tramas sonoras 26
2.3. Extracción de coeficientes cepstrales 28
2.3.1. Mel Frequency Cepstral Coefficients 28
1.-Transformada de Fourier de la señal 28
2.-Banco de filtros en escala Mel 29
3.-Operador no lineal 31

4.- Discrete Cosine Transform (DCT) 32
2.3.2. Linear Prediction Cepstral Coefficients 33
1.- Cálculo de los coeficientes LPC 33
2.- Transformación al dominio cepstral 33
2.3.3. Coeficientes delta 34
2.3.4. Cepstral Mean Variance Normalization 35
3. MÉTODOS DE COMPARACIÓN Y DECISIÓN: DTW 36
3.1. Dynamic Time Warping 36
4. ALGORITMOS DESARROLLADOS 42
4.1. Scripts y funciones para el análisis MFCC 42
4.2. Scripts y funciones para el análisis LPCC 44
5. DISPOSITIVO EXPERIMENTAL 47
5.1. Entrenamiento 47
5.2. Umbral 48
5.3. Test 49
5.4. Planteamiento del testeo 49
6. RESULTADOS 51
5.1. Optimización de los parámetros de análisis 51
5.2. Respuesta del sistema para coeficientes MFCC 54
5.3. Respuesta del sistema para coeficientes LPCC 56
7. CONCLUSIONES 58
8. TRABAJOS FUTUROS 60
9. BIBLIOGRAFÍA 61
10. GLOSARIO 63
Anexo 1 – Código Matlab MFCC 64
Anexo 2 – Código Matlab LPCC 84

Capítulo 1

INTRODUCCIÓN

1.1. Motivación

En la sociedad actual cada vez existe más dependencia de Internet. Los usuarios de redes sociales no hacen más que crecer día a día, de igual forma cada vez más servicios son ofrecidos exclusivamente a través de la nube: la administración de cuentas bancarias, operaciones en mercados bursátiles, matrículas de estudios, instancias a la administración etc. Es claro que la afiliación a organizaciones que ofrecen servicios a través de internet cada vez es mayor. Muchos de estos servicios necesitan identificar al usuario para que su funcionamiento sea correcto.

La manera típica de identificación ha sido siempre por nombre de usuario y contraseña, sin embargo debido a la gran cantidad de sitios web a los que puede estar subscrito un usuario medio, la cantidad de nombres y passwords a memorizar puede ser muy elevada y es fácil olvidarse de algún dato. Una buena alternativa para la identificación de cada usuario es la basada en datos biométricos. Reciben este nombre los indicadores que pueden ser extraídos del cuerpo humano. Estos parámetros pueden tener diferentes orígenes como la voz, las huellas dactilares, la retina, el iris, el tejido vascular etc. Resultan ser muy fiables e intransferibles puesto que no existen dos personas iguales.

El uso generalizado de smartphones y la gran cantidad de servicios sobre internet que nos ofrecen estos dispositivos, hace que sea interesante implementar en ellos un sistema de identificación mediante parámetros biométricos. Actualmente los sensores incorporados en esta clase de dispositivos, capaces de captar datos biométricos son básicamente la cámara y el micrófono.

En referencia a la cámara, ya existen numerosas aplicaciones de detección de rostro, algunas de ellas incluidas en sistemas operativos de renombre como Android, a partir de su versión 4.0 - Ice Cream Sandwich o iOS 5 en adelante. Sin embargo la precisión de los algoritmos está limitada por la baja calidad de las cámaras frontales equipadas en los dispositivos que tienen resolución VGA habitualmente, de la alta variabilidad de la imagen (diferentes condiciones de luz, distinto peinado, posibilidad de barba/maquillaje…) y de la necesidad de un enfoque correcto.

Sin embargo el reconocimiento del usuario basado en parámetros biométricos extraídos de la señal de voz es un campo poco explotado en dispositivos móviles, ya que aunque existen aplicaciones de autenticación a partir de análisis digital de audio, estos algoritmos simplemente detectan palabras o frases, y las comparan con una muestra inicial, si ambas coinciden la autenticación es satisfactoria. Este método obliga igualmente a memorizar una contraseña, es decir no consiste en identificación del locutor sino en identificación del mensaje.

1.2. Control de acceso

Hay gran cantidad de aplicaciones y sistemas que necesitan estar provistos de un control de acceso para que desempeñen su cometido de forma adecuada. El objetivo de este proceso es evitar un uso inadecuado o fraudulento. Un paso clave en los sistemas de control de acceso es la etapa de autenticación, la cual aplicada al mundo de las máquinas se puede definir como el proceso mediante el cual se intenta verificar la identidad digital de una persona que intenta acceder a un determinado servicio, es decir, se pretende que el usuario que intenta acceder al citado servicio sea

el que esté autorizado específicamente a ello y no otro. Los métodos de autenticación se pueden clasificar por la forma mediante la cual se lleva a cabo la verificación:

- Métodos basados en algo conocido: solo un determinado usuario debe de conocer unos datos de acceso que básicamente consiste en una contraseña, acompañada en ocasiones de un nombre de usuario. Es la forma más barata y extendida de autenticación. El pin del teléfono, el número del cajero o los datos de acceso a páginas web son buenos ejemplos de este método.
- Métodos basados en algo poseído: son métodos en los que se debe de tener un determinado objeto que es habitualmente una tarjeta inteligente o chip con información digital almacenada. Este es el caso de gran cantidad de operadores de televisión por satélite, en los que se necesita una Smart-card para que funcione el decodificador. También lo encontramos como métodos de entrada a diversas instalaciones en forma de tarjetas de plástico o en la tarjeta SIM de los teléfonos móviles, la cual identifica al propietario como usuario único de ese número telefónico.
- Métodos basados en parámetros biométricos: el cuerpo posee determinadas características que pueden ser de distinta naturaleza y son involuntarias al individuo que las posee. Se pueden utilizar dichas características mediante sistemas digitales de detección para autenticar al usuario.

1.2.1. Parámetros biométricos

El cuerpo de cada individuo es único y por ello se pueden extraer de él parámetros que identifiquen a cada usuario de forma unívoca mediante el uso de las herramientas analíticas adecuadas. Estos parámetros pueden estar basados en cualquier característica intrínseca al cuerpo humano, desde el ADN al olor corporal o la manera de caminar. Típicamente, por ser menos complejas de detectar y gozar de mayor estabilidad se han utilizado cinco [1] características, las cuales están detalladas en la tabla 1 [2][3] .

Como consideraciones generales se puede decir que los análisis de iris y de retina son los métodos más fiables, sin embargo el coste del dispositivo detector es muy elevado e imposible de implantar en dispositivos móviles a día de hoy.

A la anterior tabla hay que añadir los últimos avances en procesado digital de imagen aplicado a la detección de rostros, tanto 2D analizando una imagen normal, como 3D interpolando varias imágenes para obtener un volumen. Como ya se ha comentado, si bien existe un puñado de aplicaciones de detección facial, la baja calidad de las cámaras frontales de los dispositivos móviles así como la imposibilidad de un enfoque preciso, hace inviable el desarrollo de un algoritmo robusto de procesado 2D de imagen. El análisis 3D se realiza a partir de varias fotografías 2D con lo que un enfoque preciso es requerido igualmente. Un sistema de este tipo también queda descartado.

Según la tabla 1 la voz no parece una característica demasiado fiable como indicador biométrico, por ser poco estable. No obstante hay que puntualizar que la tabla recoge datos anteriores a 1997 y que en la última década han surgido nuevos algoritmos y mejoras de los ya existentes, consiguiendo que la autenticación mediante el reconocimiento del hablante por análisis de la voz, sin llegar a la robustez del iris o la retina, sea un método de gran seguridad.

	Ojo - Iris	Ojo - Retina	Huellas dactilares	Geometría de la mano	Escritura Firma	Voz
Fiabilidad	Muy alta	Muy alta	Alta	Alta	Alta	Alta
Facilidad de uso	Media	Baja	Alta	Alta	Alta	Alta
Prevención de ataques	Muy Alta	Muy alta	Alta	Alta	Media	Media
Aceptación	Media	Media	Media	Alta	Muy alta	Alta
Estabilidad	Alta	Alta	Alta	Media	Media	Media
Identificación y autenticación	Ambas	Ambas	Ambas	Aut.	Ambas	Aut.
Estándars	-	-	ANSI/NIST FBI	-	-	SVAPI
Interferencias	Gafas	Irritaciones	Suciedad, asperezas ...	Artritis, reumatismo ...	Firmas fáciles o cambiantes	Ruido, resfriados ...
Utilización	Instalaciones nucleares,centros penitenciarios	Instalaciones nucleares, centros penitenciarios	Policía, industrial	General	Industrial	Accesos remotos en bancos
Precio por nodo en 1997 (USD)	5000	5000	1200	2100	1000	1200

Tabla 1

1.3. Procesado digital de audio

El PDVA es una rama de la ingeniería que se ha desarrollado enormemente en las últimas décadas, tanto como ha permitido el incremento en la capacidad de cómputo de los ordenadores y la aparición de herramientas de desarrollo software de fácil acceso como Matlab. Por todo ello ha sido posible la implementación de algoritmos cada vez más complejos y robustos en tiempos de ejecución razonables y

en dispositivos portables permitiendo el desarrollo de aplicaciones de gran utilidad como Shazam. El PDVA se puede clasificar a grandes rasgos según la figura 1.1 En verde se resalta la rama del procesado de audio objeto de este trabajo.

Figura 1.1

1.4. Objetivo del proyecto

El objetivo de este trabajo es dar un repaso al estado del arte en técnicas de verificación del locutor y desarrollar un sistema de este tipo con el software Matlab que pueda funcionar en las condiciones de uso habituales de los dispositivos móviles. Se pretende estudiar el comportamiento del sistema ante cambios en determinados parámetros del procesado de señal, estableciendo las técnicas y los valores óptimos para que la autenticación sea lo más robusta posible.

1.4.1. Definiciones [4]

1.- Etapas de un sistema RAL: entrenamiento y test

Cualquier sistema de reconocimiento automático del locutor precisa obligatoriamente de dos etapas: entrenamiento y test.

- Entrenamiento: esta etapa tiene como fin generar un modelo que represente a un determinado locutor. Para construir este modelo es necesario extraer de la voz ciertos parámetros que caractericen la señal. En el caso de sistemas dependientes del texto, esta caracterización se lleva a cabo además para un conjunto fonético concreto consistente en una palabra, frase o texto en general.
- Test: es la etapa en la que un sujeto accede al sistema para verificar su identidad ó en el caso de un impostor, para suplantar la identidad de otro usuario. La decisión de aceptación, cuando el sujeto es quien dice ser, o rechazo se lleva a cabo mediante la comparación de los parámetros previamente entrenados para ese sujeto y los extraídos en el test.

En ambas etapas se realiza el mismo procesado digital, que persigue extraer de la señal los parámetros que mejor caractericen al sujeto; pronunciando una determinada palabra cuando se trata de sistemas dependientes del texto.

2.- Identificación vs verificación

El campo del reconocimiento del locutor se puede dividir a su vez en dos ramas: identificación y verificación del hablante.

- La identificación consiste en averiguar a quién pertenece un segmento de voz a priori desconocido. Esto se consigue comparando estadísticamente los parámetros extraídos en el test de este usuario con los parámetros de un número N de usuarios, que han sido entrenados y registrados previamente en una base de datos. El vector de la base de datos que presente el mayor parecido con los parámetros extraídos en el test y supere un umbral de correlación con estos será el usuario identificado.

- La verificación es la comprobación de que los parámetros extraídos en el test realizado por un usuario pertenezcan o no a dicho usuario. Se comparan los parámetros del test con una sola entrada de la base de datos, la que se corresponde al supuesto usuario que realiza el test. La salida del sistema es binaria, de aceptación o rechazo.

Este trabajo pretende estudiar la posibilidad de autenticación mediante análisis de la voz y por tanto se centra en la verificación del hablante.

3.- Texto dependiente vs texto independiente

Los sistemas de verificación del hablante se pueden clasificar a su vez en sistemas dependientes de una pronunciación fonética determinada y en sistemas independientes. En los sistemas dependientes, el texto con el que es realizado el test debe de ser el mismo que se ha pronunciado en la evaluación. El entrenamiento puede estar formado por letras, palabras, frases o textos más extensos. Los sistemas independientes del texto no tienen esta premisa y son capaces de realizar la verificación a partir de un segmento de audio desconocido. Las tasas de error son sin embargo notablemente superiores en estos últimos sistemas y precisan de tiempos de entrenamiento y test muy superiores.

En este trabajo se va centrar en los algoritmos dependientes del texto por las ventajas que presenta.

4.- Medidas de robustez y comportamiento del sistema: curvas ROC

Un sistema de verificación del locutor, como ya hemos visto puede tener dos posibles salidas: aceptación o rechazo. Por ello también puede tener dos tipos de errores:

- Falsa aceptación: también llamado error de falsa alarma, es cuando un impostor se hace pasar por otra persona y el sistema, siendo engañado le acepta.

- Falso rechazo: se produce cuando un usuario auténtico, que es quien dice ser, es rechazado por el sistema.

Por tanto la calidad de un sistema se puede medir calculando las probabilidades de falsa aceptación, o de falso rechazo. Además existe otro parámetro indicativo de la precisión de un sistema llamado EER, representado en la figura 1.2, que no es más que la probabilidad de error del sistema cuándo la probabilidad de falsa aceptación es igual a la de falso rechazo.

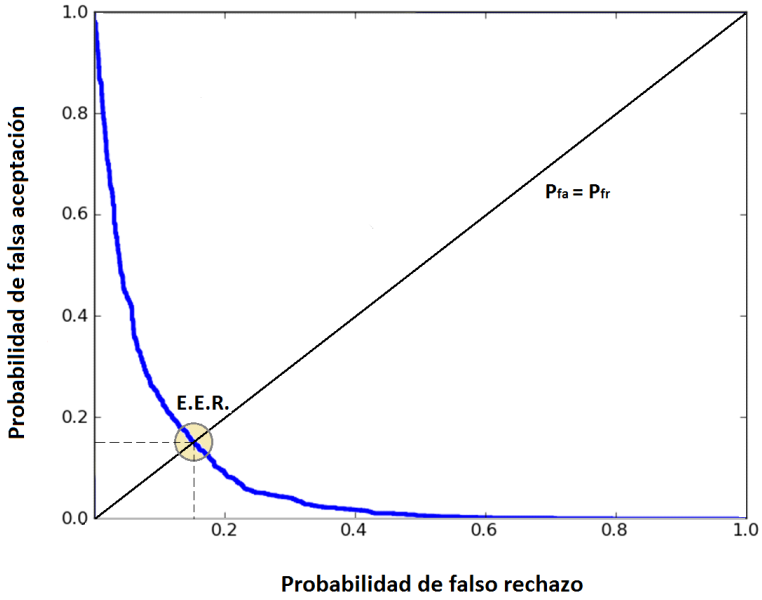


Figura 1.2

Un sistema ideal de verificación sería aquel en el que las distribuciones de densidad de probabilidad de puntuación para usuarios genuinos e impostores, están separadas. Utópicamente se tendría una fiabilidad del 100% puesto que el umbral de decisión se pondría en una puntuación intercalada entre las dos distribuciones. En la realidad esto no es posible y los sistemas implementados presentan un comportamiento como el descrito en la figura 1.3 Ambas distribuciones se solapan de una forma más o menos acusada y hay que elegir un umbral de decisión. En una aplicación real normalmente es más critico el error de falsa aceptación, es decir, que un usuario no autorizado tenga acceso al sistema, aplicación, recinto etc. Por ello se suele reducir este tipo de error a costa de aumentar el de falso rechazo, es decir, mover el umbral de la figura

hacia la derecha. El error de falso rechazo no es tan grave porque el usuario siempre tiene la opción de volver a intentar la autenticación un número de veces prudencial y prestablecido.

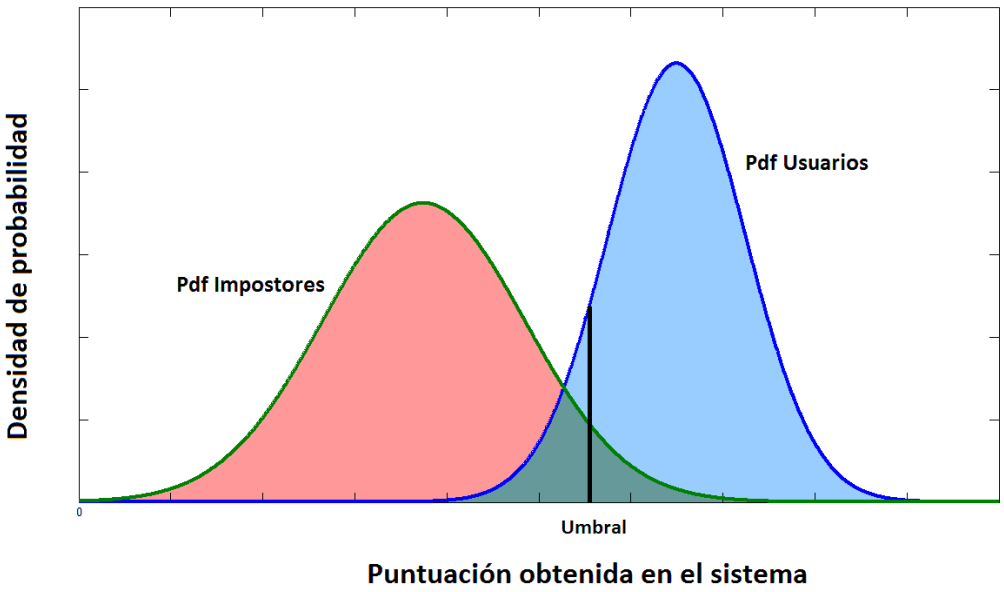


Figura 1.3

Las curvas ROC como la de la figura 1.2 representan muy bien la calidad de un sistema RAL ya que relaciona los dos errores posibles. Se va ha explicar brevemente como se confecciona una de estas curvas.

- Se deben de hacer dos tipos de tests. Por un lado tests auténticos en los que el entrenamiento y el test los ha realizado el mismo locutor y por otro tests impostores en los que el modelo del locutor entrenado se enfrenta a otros locutores.
- Es conveniente realizar gran cantidad de tests de cada tipo para obtener tasas de respuesta fiables. En este trabajo se han realizado 100 tests auténticos y otros 100 impostores.
- Los tests auténticos dan una cantidad de verdaderos aceptados, VA, y otra de falsos aceptados FR; con los tests impostores se obtiene un numero de verdaderos rechazados, VA, y otro de falsos aceptados, FA.

- Se define:

$$Sensibilidad = TVA = \frac{VA}{(VA + FR)}$$

$$Especificidad = 1 - TFA = \frac{VR}{(VR + FA)}$$

$$1 - Especificidad = TFA$$

El eje *y* de la curva ROC representa la sensibilidad del sistema o la tasa de verdaderos aceptados y el eje *x* representa 1 menos la especificidad, es decir la tasa del error de falsa aceptación.

1.4.2. Consideraciones acerca del entorno de utilización de los dispositivos móviles

El sistema está pensado para ser utilizado en dispositivos móviles, esto obliga a:

- Mínima carga computacional: el rendimiento hardware, (procesador fundamentalmente) de smartphones y tablets no puede ser comparado todavía al que incorporan ordenadores portátiles o de sobremesa, si bien cada vez se está igualando más, debido fundamentalmente a la mejor gestión de recursos realizada por sistemas operativos móviles y a las grandes inversiones privadas en esta tecnología. La aplicación debe de perseguir la ejecución a tiempo real, es decir reducir los tiempos de entrenamiento y test.
- Máxima inmunidad al ruido: los dispositivos móviles se manejan habitualmente en ambientes ruidosos, por ello será necesario elaborar un sistema que si bien precise de un ambiente tranquilo para el entrenamiento, pueda realizar el test en ambientes moderadamente ruidosos.

Sobre el primer punto, en general, se puede afirmar que un sistema de reconocimiento del locutor es mas robusto y preciso conforme aumenta la información de entrada, que en este caso es la longitud del texto. Por el

contrario es más inexacto y vulnerable cuanto menor es el contenido fonético del audio empleado. Por otro lado cuanta más longitud tenga el texto también será mayor el propio tiempo de pronunciación y la posterior etapa de procesado.

Por todo ello es de vital importancia desarrollar un algoritmo que maximice la eficiencia del sistema, consiguiendo unas buenas tasas de reconocimiento a costa de un tiempo de pronunciación y procesado de audio no demasiado largo, ya que se busca la ejecución a tiempo real. Se tendrá en cuenta este compromiso en la elaboración del sistema.

1.4.3. Aparato fonador humano [5]

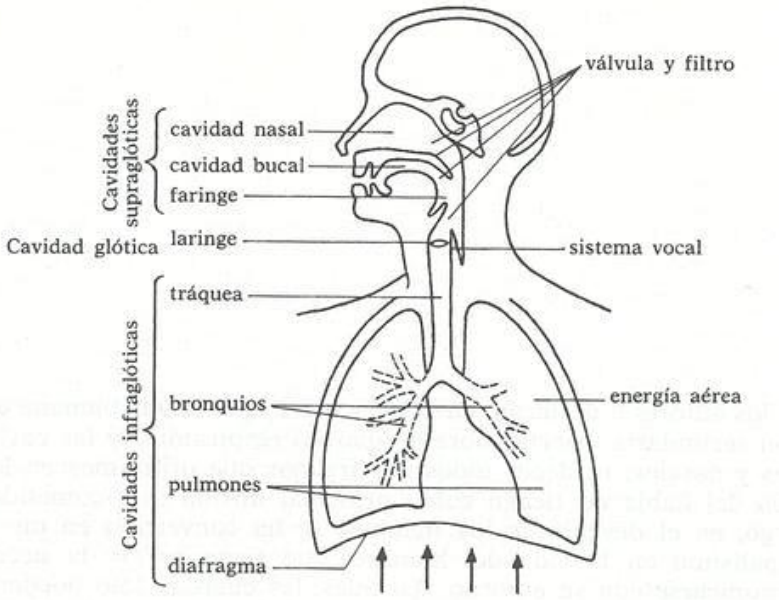


Figura 1.4: esquema del aparato fonador humano

Como se puede ver, el aparato fonador, está integrado en el sistema respiratorio y se puede dividir, de la zona inferior a la superior en:

- Cavidades infraglóticas: Son todas las que se encuentran situadas por debajo de la laringe. Estas cavidades alojan el diafragma, los pulmones, los bronquios y la tráquea. Aquí se produce el flujo de aire que más tarde es modulado. El principal responsable de este flujo es el diafragma, que es un

músculo situado bajo los pulmones y con forma convexa que se contrae y se relaja haciendo posible la respiración. En la fase de relajación se produce el flujo de aire desde los pulmones hacia el exterior. Los bronquios y la tráquea, desde el punto de vista de la producción de voz, solo son tuberías cartilaginosas que guían dicho flujo.

- Cavidad glótica: contiene básicamente la laringe y esta contiene a las cuerdas vocales que son dos fibras musculosas, situadas formando un triangulo en la laringe y cuya apertura (área del triángulo) puede variar a voluntad. Cuando el flujo de aire generado en los pulmones atraviesa las cuerdas vocales, produce en estas una vibración que puede variar en frecuencia e intensidad atendiendo a tres factores: masa, longitud y tensión de la glotis. La vibración se produce únicamente en determinados segmentos de voz llamados segmentos sonoros. Es aquí dónde la señal adquiere la propiedad de cuasi periodicidad, cuya frecuencia es igual a la frecuencia con la que vibran las cuerdas vocales, también llamada frecuencia fundamental o pitch.

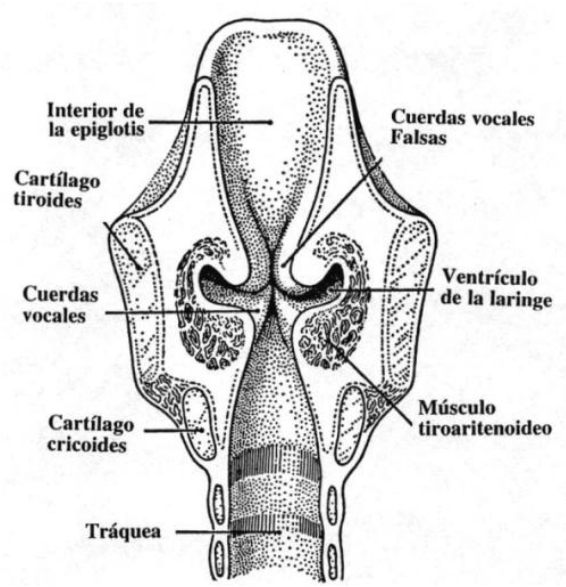


Figura 1.5: Glotis

- Cavidades supraglóticas: el conjunto de estas cavidades es el llamado tracto vocal, que actúa de caja de resonancia y modula la señal de voz. El tracto vocal está formado por la cavidad faríngea, la oral y la nasal.

- Faringe: es un tubo que tiene una longitud media de aproximadamente 17 cm y una sección transversal de entre 0 y 20
- Cavidad nasal: queda definido por el velo del paladar en su parte interna y los orificios nasales en la externa. Los sonidos nasales son fruto del acoplamiento acústico de las cavidades oral y nasal, producidas por la bajada de la úvula (velo).
- Cavidad oral: comprende la boca y todos sus elementos: dientes, lengua etc.

El aparato de fonación, visto así, es muy complejo y difícil de modelar, sin embargo se puede simplificar en tres bloques más sencillos: un generador de energía, un sistema vibrante que simula las cuerdas vocales y un sistema resonante que modela el tracto vocal y puede ser aproximado por el diagrama de la figura 1.6:

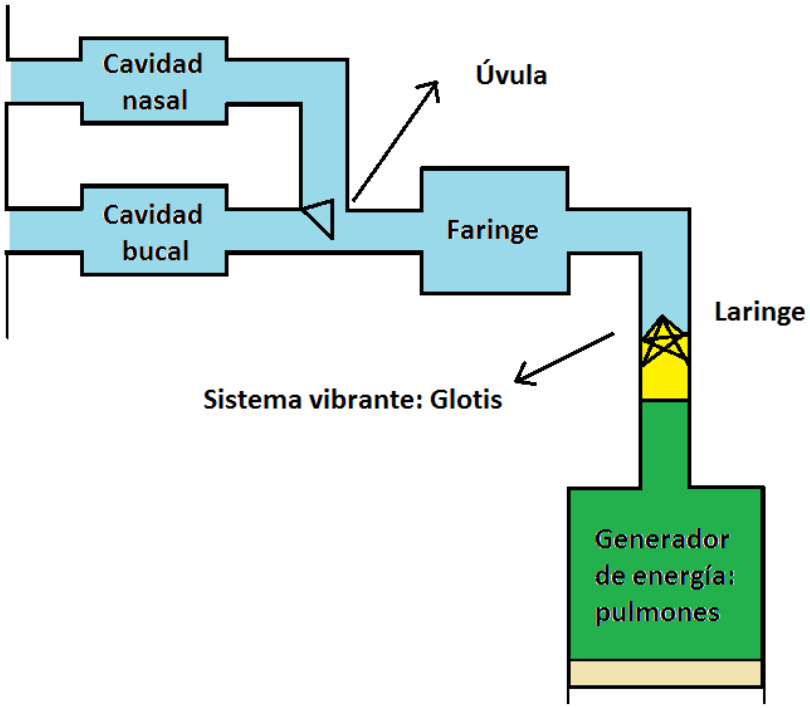


Figura 1.6: esquema simplificado

La forma de representación de la figura 1.7 permite modelar el aparato fonador como una secuencia de tubos resonantes concatenados. Las frecuencias de resonancia o los más popularmente llamados formantes son de especial interés porque dependen directamente de la forma y el tamaño del tracto vocal, el cual es distinto en cada persona. Por tanto conociendo los formantes y el pitch para un determinado fonema es posible caracterizar de una forma precisa a un determinado locutor.

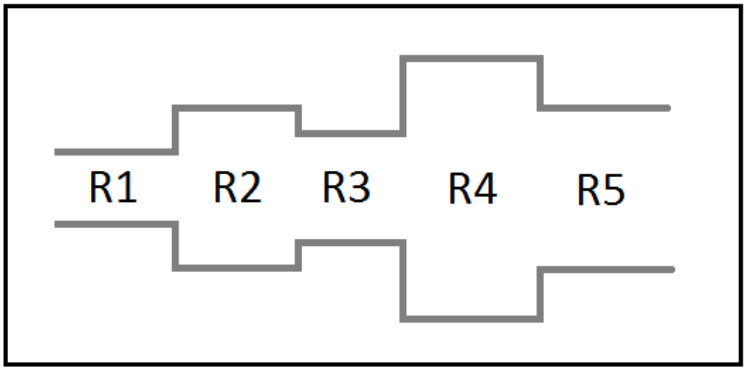


Figura 1.7: modelo de tracto vocal como tubos

Este modelo se comporta como un sistema lineal que contiene varios polos de magnitud finita y es la base para la extracción de los coeficientes de predicción lineal o LPC.

1.4.4. Antecedentes en verificación del locutor

Desde principios de los años 70 se llevan desarrollando sistemas de reconocimiento del locutor, tanto de verificación como de identificación; dependientes e independientes del texto. Un sistema RAL, tanto en la etapa de entrenamiento como en la de test, tiene habitualmente dos grandes bloques. El primero es el encargado de extraer de la voz una serie de parámetros espectrales que caractericen al locutor de la forma más precisa posible. Estos parámetros se concatenan para formar un vector que constituye una especie de código de barras de identificación del locutor. El segundo bloque evalúa y

decide si un determinado vector detectado corresponde a un determinado usuario registrado previamente en la base de datos.

Respecto a la manera de extraer los parámetros de la voz, existen dos grupos de técnicas con sus diversas variantes. El primer grupo es el análisis basado en procesado homomórfico del espectro de la señal, llamado también análisis cepstral que se fundamenta en la transformada de Fourier, por otra parte está el análisis de predicción lineal o LPC que se apoya en el modelado del tracto vocal del aparato fonador humano como una sucesión de tubos resonantes visto en la anterior sección. Esto se realiza mediante la implementación de un filtro FIR.

Cada uno de los métodos presenta ventajas e inconvenientes. Las líneas de investigación más recientes de reconocidos laboratorios como el HTK de la universidad de Cambridge se apoyan sobretodo en variantes del análisis cepstral, como por ejemplo el Mel-Cepstrum que combina el procesado homomórfico con un muestreo en la escala Mel y con el que se obtienen los famosos coeficientes MFCC. El análisis del cepstrum suele optimizarse añadiendo algoritmos que añaden robustez al sistema frente a diversos tipos de ruidos y variaciones.

Las técnicas más utilizadas de normalización de los coeficientes cepstrales son el RASTA *(RelAtive SpecTrA)* y el CMVN *(Cepstral Mean Variance Normalization).*

Sin embargo no hay que olvidar que el procesado homomórfico requiere de un operador no lineal para convertir la convolución temporal entre la señal de excitación y la respuesta del tracto vocal en una suma de ambas componentes en el dominio cepstral. El operador no lineal más usado para este propósito es el logaritmo, el cual hace al sistema más vulnerable al ruido y a pequeñas variaciones en el input.

El análisis LPC, para un mismo número de coeficientes es menos preciso y más mecánico, pero resalta muy bien los formantes y presumiblemente necesita menor relación señal a ruido para un estudio satisfactorio de la señal.

A esto hay que añadir que la carga computacional producida es menor al prescindir de las FFTs.

Respecto del segundo bloque, que se ocupa de crear un modelo de cada locutor y decidir la aceptación o el rechazo frente a un intento de test, existen varias técnicas que no son exclusivas del análisis de audio, utilizándose también en otras ciencias como la ingeniería genética por ejemplo. También este bloque se puede dividir en dos tipos

1. Por un lado están los modelos de plantillas, en los que el locutor queda almacenado en el sistema mediante un conjunto de vectores. En el test los vectores extraídos deberán tener un cierto parecido a la plantilla entrenada si el locutor es el mismo. El algoritmo más importante de este tipo es el alineamiento temporal dinámico (DTW).
2. En el otro bando está el modelado estadístico, en el que se crea un modelo a partir de los parámetros de la voz extraídos, o lo que es lo mismo se estudia la probabilidad de que cada coeficiente tome ciertos valores para un determinado locutor. Habitualmente el modelado estadístico da mejores resultados, si bien necesita un entrenamiento más largo y costoso computacionalmente. Como exponentes de este punto están los modelos de mezclas gaussianas (GMM-UVM), las redes neuronales artificiales (ANN) y los modelos ocultos de Markov (HMM).

Refiriéndonos únicamente al reconocimiento del locutor dependiente del texto las técnicas más utilizadas en la actualidad son el DTW, que intenta salvar los desajustes temporales en la pronunciación ajustando ambos vectores de características y los HMM, que modelan la voz como una transición entre estados, asignando una probabilidad de transición a cada uno de ellos.

Referencia	Organización responsable	Parámetros extraidos	Método utilizado para la decisión	Texto	Calidad de la señal de voz	Base de datos	Error
Atal 1974	AT&T	Cepstrum	Comparación de patrones	Dependiente	Laboratorio	10	2% (Id.) - 0.5 s 2% (verif.) - 1 s
Markel and Davis 1979	STI	LP	Términos estáticos largos	Independiente	Laboratorio	17	2% (Id.) - 39 s
Furui 1981	AT&T	Cepstrum normalizado	Comparación de patrones	Dependiente	Teléfono	10	0.2% (verif.) - 3 s
Schwartz, et al. 1982	BBN	LAR	PDF no paramétrico	Independiente	Teléfono	21	2.5% (id.) - 2 s
Li and Wrench 1983	ITT	LP-Cepstrum	Comparación de patrones	Independiente	Laboratorio	11	21% (id.) - 3 s 4% (id.) - 10 s
Doddington 1985	TI	Banco de filtros	Dinamic time warping	Dependiente	Laboratorio	200	0.8% (verif.) - 6 s
Soong, et al. 1985	AT&T	LP	VQ Verosimilitud de distorsión y de radio	10 dígitos aislados	Teléfono	100	5% (id.) - 1.5 s 1.5% (id.) - 3.5 s
Higgins and Wohlford 1986	ITT	Cepstrum	DTW probabilidad de resultados	Independiente	Laboratorio	11	10% (verif.) - 2.5 s 4.5% (verif.) - 10 s
Attili, et al. 1988	RPI	Cepstrum, LP, auto - correlación	Términos estáticos largos	Dependiente	Laboratorio	90	1% (verif.) - 3 s
Higgins et al. 1991	ITT	LAR, LP-Cepstrum	DTW probabilidad de resultados	Dependiente	Oficina	186	1.7% (verif.) - 10 s
Tishby 1991	AT&T	LP	HMM (Mezclas AR)	10 dígitos aislados	Teléfono	100	2.8% (verif.) - 1.5 s 0.8% (verif.) - 3.5 s
Reynolds and Carlson 1995	MIT-LL	Mel-Cepstrum	HMM (GMM)	Dependiente	Oficina	138	0.8% (id.) - 10 s 0.12% (verif.) - 10 s
Che and Lin 1995	Rutgers	Cepstrum	HMM	Dependiente	Oficina	138	0.56% (id.) - 2.5 s 0.14% (id.) - 10 s 0.62% (verif.)- 2. 5 s
Colombi, et al. 1996	AFIT	Cep, Eng dCep, ddCep	HMM monophone	Dependiente	Oficina	138	0.22% (id.) - 10 s 0.28% (verif.) - 10 s
Reynolds 1996	MIT-LL	MelCepstrum , Mel-dCepstrum	HMM (GMM)	Independiente	Teléfono	416	11% (verif.) - 3 s 6% (verif.) - 10 s 3% (verif.) - 30 s

Tabla 2

1.5. Hardware y Software empleado

Para la realización del proyecto se ha utilizado el siguiente equipo:

1. Hardware.
    - Ordenador portátil HP Pavilion g6, Intel core i5, 2,4 GHz, 4Gb DDR RAM.

    - Smartphone Alcatel One Touch 997.

    - Cable de datos USB

2. Software.
    - Matlab r2013b
    - DSP ToolBox para Matlab.
    - Aplicación de grabación para Android: Recforge
    - Algoritmo de DTW de código abierto para Matlab implementado por Pau Mic, Sven Mensing y otros.
    - Algoritmo para la creación de banco de filtros triangulares implementado por Kamil Wojcicki.
    - Adobe Photoshop cs4.
    - Microsoft Word 2010 .

Capítulo 2

EXTRACCIÓN DE PARÁMETROS PARA LA CARACTERIZACIÓN

El sistema implementado, simplificado en la figura 2.1, se ha dividido en tres bloques. El primero de ellos se ocupa de importar el archivo de audio en formato .wav y acondicionar la señal de forma que el procesado posterior sea más sencillo y fiable. El segundo bloque es el encargado de extraer los parámetros. Se han desarrollado las dos técnicas descritas anteriormente: análisis cepstral a partir de predicción lineal y mediante transformadas de Fourier, para poder compararlas con el objetivo de evaluar la mejor técnica de caracterizar la voz.

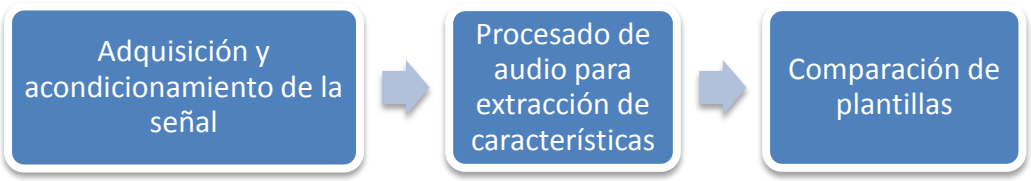


Figura 2.1

2.1. Adquisición de la señal

La señal que produce el sistema fonador es una onda mecánica, acústica y analógica. Esta onda hay que digitalizarla para poder tratarla. La conversión A/D se lleva a cabo muestreando la señal continua de entrada a una cierta velocidad o frecuencia de muestreo. Esta velocidad tiene que satisfacer el teorema de Nyquist, el cual obliga a que el muestreo se realice a una frecuencia igual o superior al doble de la

mayor frecuencia de la señal. Como los formantes superiores con energía significativa de la señal de voz alcanzan los 3000 Hz aproximadamente, es suficiente con muestrear la señal de entrada a 6000 Hz. Para dar un margen y ser un estándar del audio digital, el muestreo se ha realizado a 8 KHz. Por otro lado hay que decidir la precisión con la que se muestrea la señal, es decir, la cantidad de valores de amplitud que puede tomar una determinada muestra. Se ha optado por una codificación de 16 bits que da un total de $2^{16} = 65536$ valores posibles, lo cual es una precisión más que suficiente. Por último hay que decir que no se ha realizado ningún procesado de compresión de señal como MPEG layer 3 o AAC, siendo el formato de audio resultante .wav , 8KHz, 16 bits, señal mono.

2.2. Acondicionamiento de la señal [7]

2.2.1. Quitar componente continua

La componente continua es una desviación general de la señal, positiva o negativa debido principalmente a micrófonos de mala calidad o a campos magnéticos presentes. Esto afecta de forma crítica en etapas posteriores del procesado, como por ejemplo en el calculo de tasa de cruces por cero, dónde dicho coeficiente es mucho menor cuándo hay presencia de DC.

La componente continua se elimina restando la media de la señal a cada una de las muestras (2.1)

$$x_{DC} = x - \frac{1}{N}\sum_{i=1}^{N} x[i] \qquad (2.1)$$

2.2.2. Filtrado paso alto

Es posible que en la señal se hayan introducido componentes de baja frecuencia procedentes de distintas fuentes. Estas componentes no son de interés para el estudio de la señal de voz y por ello es mejor eliminarlas. La frecuencia de corte se ha establecido en 75 Hz que es la mínima a la que se encuentra la frecuencia fundamental (pitch) para la voz masculina [6].

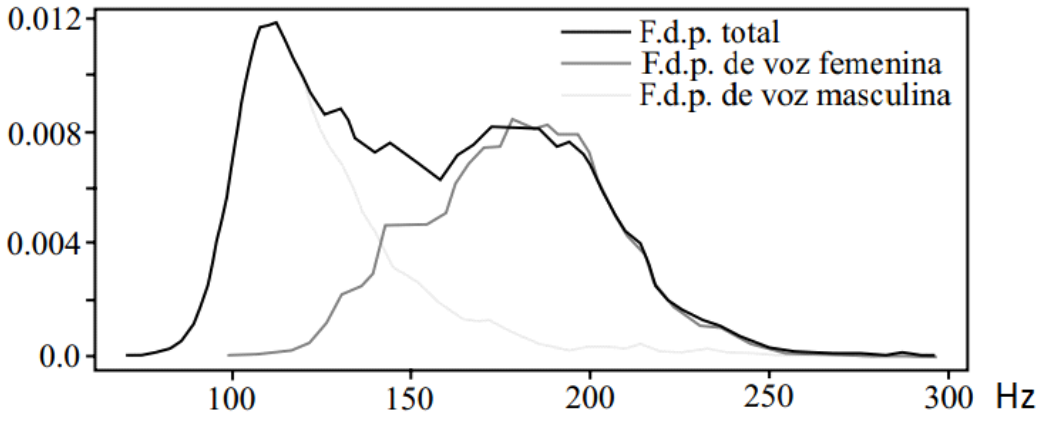


Figura 2.2: Pdf de las frecuencias fundamentales en hombres y en mujeres

2.2.3. Filtrado de pre-énfasis

El filtro de preénfasis pretende realzar la señal en altas frecuencias a razón de 6 dB por octava o lo que es lo mismo 20 dB por década. Se puede demostrar que el modelo de las cuerdas vocales produce un de-énfasis de 12 dB por octava para tramas sonoras de la señal. Los 6 dB por octava que enfatiza aproximadamente la radiación por los labios unidos a los 6 dB de este filtro compensan de alguna forma el modelo glotal. Se consigue con ello alisar el espectro de la señal y resaltar los formantes.

El filtro es de primer orden y en este proyecto se ha implementado cogiendo la expresión:

$$H_{pe} = 1 - \alpha_{pe} \, Z^{-1} \qquad (2.2)$$

Por tanto tiene un único coeficiente, que toma típicamente valores próximos a 1, en el trabajo realizado α_{pe}= 0,97. Su respuesta se puede ver en la figura 2.3.

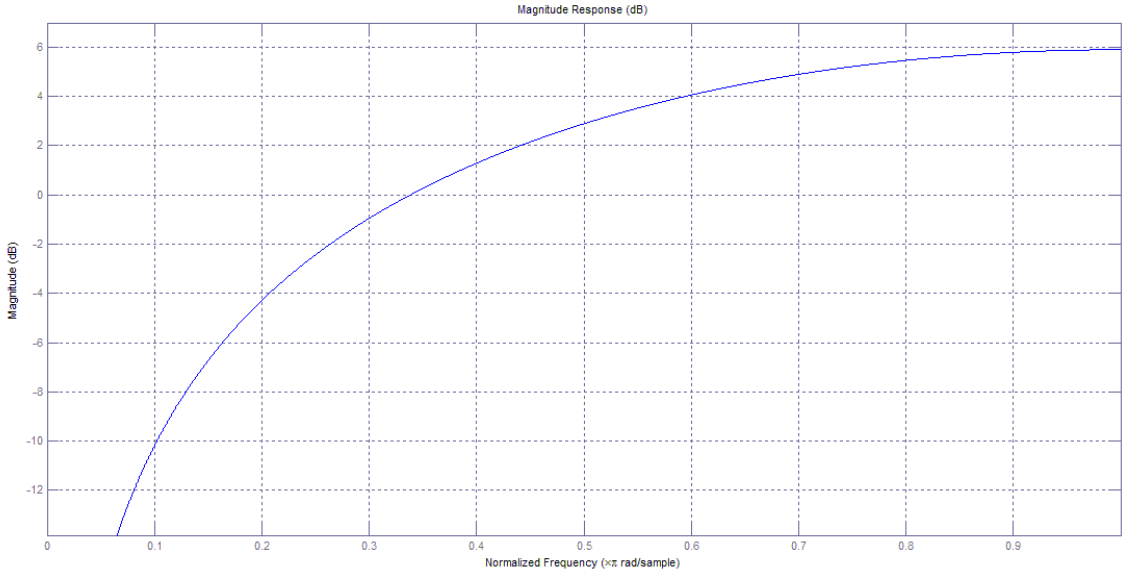


Figura 2.3

Como se puede ver en la respuesta en frecuencia, las altas frecuencias quedan enfatizadas.

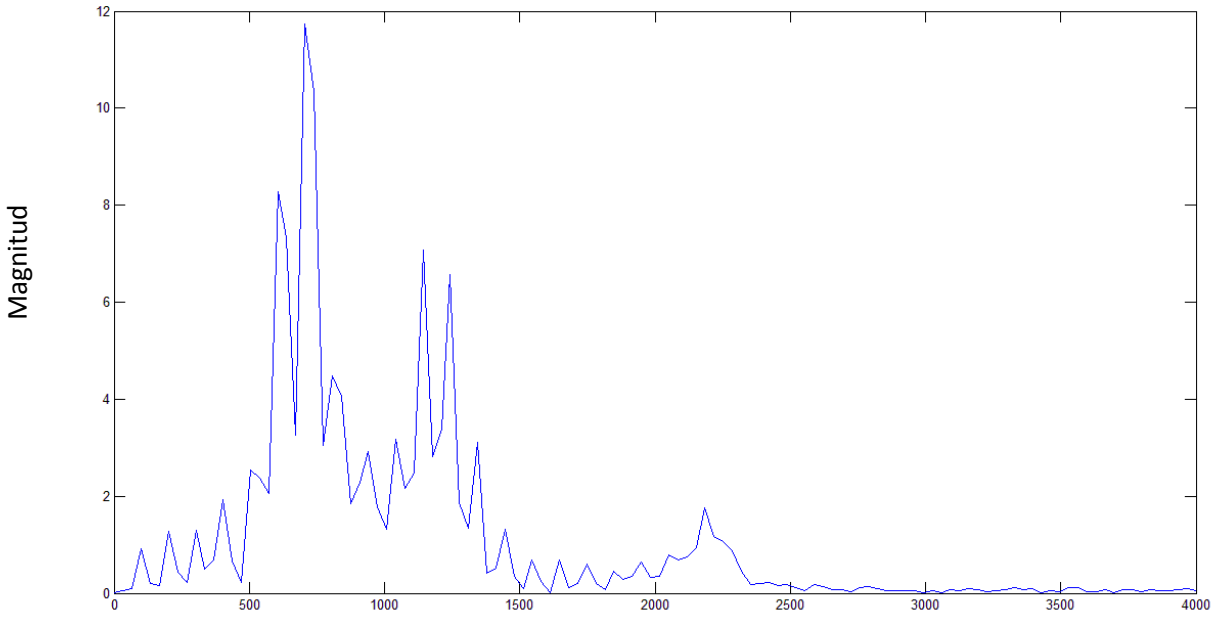


Figura 2.4

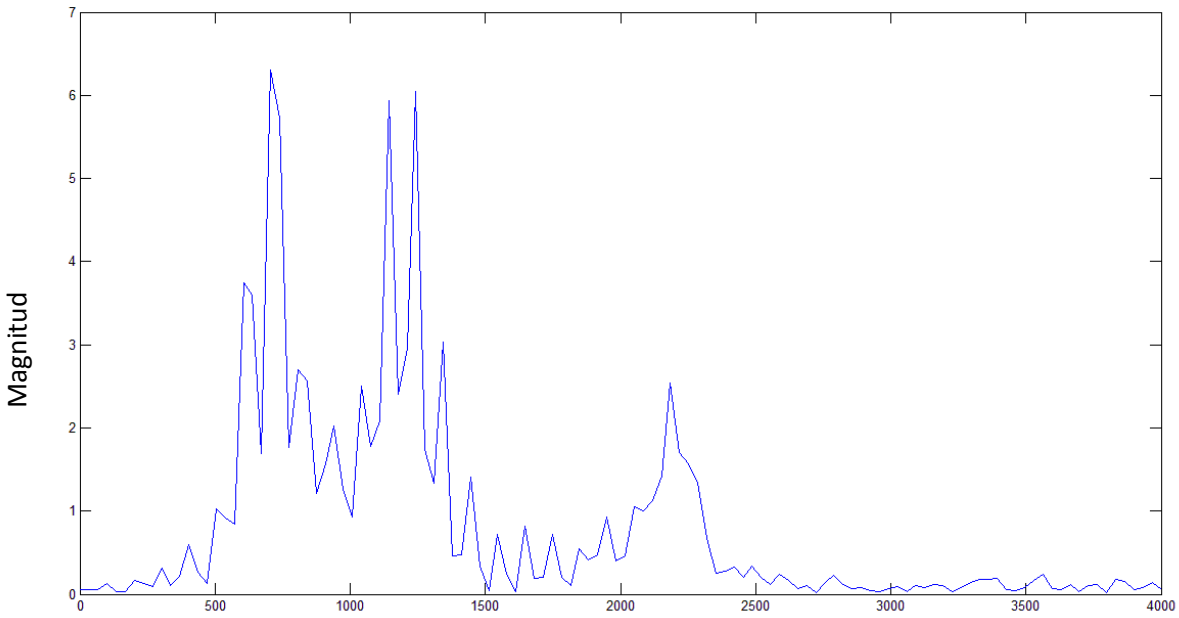


Figura 2.5

2.2.4. Enventanado

La señal de voz tiene naturaleza no estacionaria y es una señal estocástica, aun cuando el locutor pretende controlar lo que dice. Debido a esta variabilidad su estudio se hace muy complicado. Sin embargo en intervalos cortos de tiempo para fonemas concretos, la señal se estabiliza y se aproxima a ser estacionaria. Estos fonemas corresponden a los sonidos vocálicos y al ser pronunciados las cuerdas vocales vibran. Esta vibración confiere a la señal la propiedad de cuasiperiodicidad que permite extraer la frecuencia fundamental F0 o pitch y los primeros formantes, claves para caracterizar la voz.

Por ello es necesario dividir la señal en segmentos más pequeños que a partir de ahora denominaremos tramas. La duración de cada trama de análisis es un parámetro controvertido y depende fundamentalmente de la velocidad de articulación. Cuanto más rápido se pronuncie, será requerido un menor tamaño de ventana. Típicamente el tiempo puede ir desde 20 hasta 60

milisegundos. En el presente proyecto se estudiará el tamaño de ventana óptimo.

Otro tema fundamental a tratar es la superposición entre tramas. Habitualmente es recomendable que dos tramas consecutivas compartan muestras o se solapen entre sí. Esto se hace con dos objetivos, el primero es conseguir mayor correlación entre tramas adyacentes y suavizar por tanto la variación del espectro. El segundo es no perder información debido a que la ventana utilizada puede suavizar los bordes como se puede ver en la figura 2.6. El número de muestras superpuestas de nuevo depende de la velocidad de articulación. A mayor velocidad, mayor tiene que ser el overlap. De nuevo se realizará un estudio del mejor valor para este parámetro con el objetivo de disminuir los errores. En el proyecto el overlap está acotado en (0,1). Cero significa que el solapamiento es total y 1 que no hay solapamiento. Así si el overlap = 0.5 y la trama tiene una longitud de 240 muestras existe una superposición de 120 muestras.

Por último para realizar el enventanado hay que elegir la ventana a utilizar teniendo en cuenta que multiplicar una ventana con la señal en el dominio temporal, es convolucionar ambos espectros en el dominio frecuencial. Por tanto el espectro de la ventana a utilizar debe de intentar cumplir dos premisas:

- Que tenga una alta resolución frecuencial, lo que se traduce en un lóbulo central estrecho y alto.
- Que los lóbulos laterales decaigan rápidamente para que no afecten en exceso al espectro de la trama de voz.

Ambas características son disociadas, en la ventana rectangular por ejemplo se cumple muy bien la primera premisa pero no la segunda, por eso hay que buscar un compromiso entre las dos. Existen numerosas ventanas como la rectangular, Kaiser, Blackman, Barlett, hamming, hanning etc. En procesado de audio, actualmente se utiliza casi siempre la ventana de

hamming, representada en la figura 2.7 [4], por cumplir razonablemente bien las dos premisas citadas

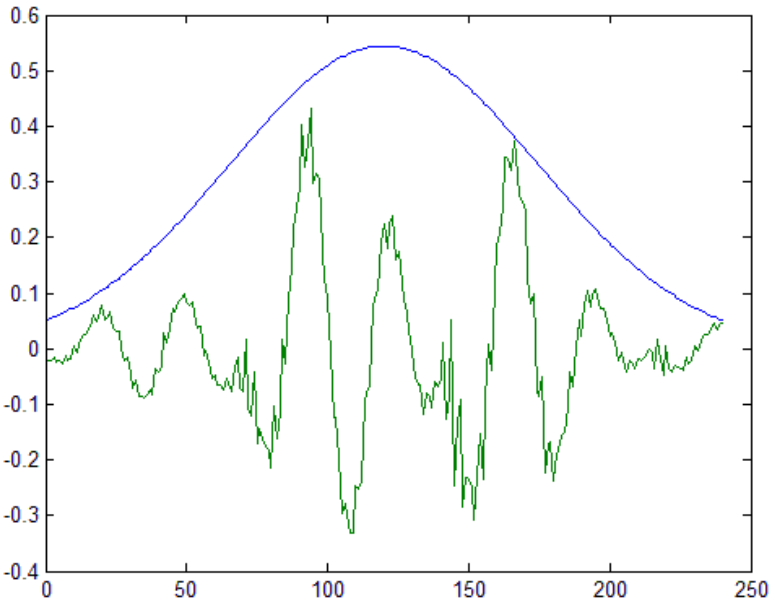


Figura 2.6

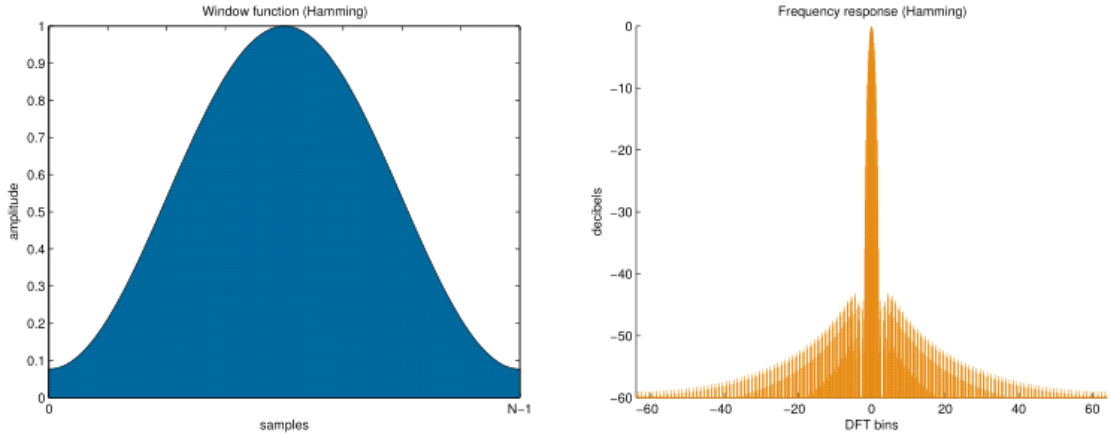


Figura 2.7

2.2.5. Segmentado de tramas sonoras

Como se ha explicado las tramas que contienen la información más relevante para la caracterización son las llamadas sonoras. Por ese motivo es

necesario establecer unos criterios que permitan aislar este tipo de tramas y desechar las pertenecientes a silencios o a sonidos sordos.

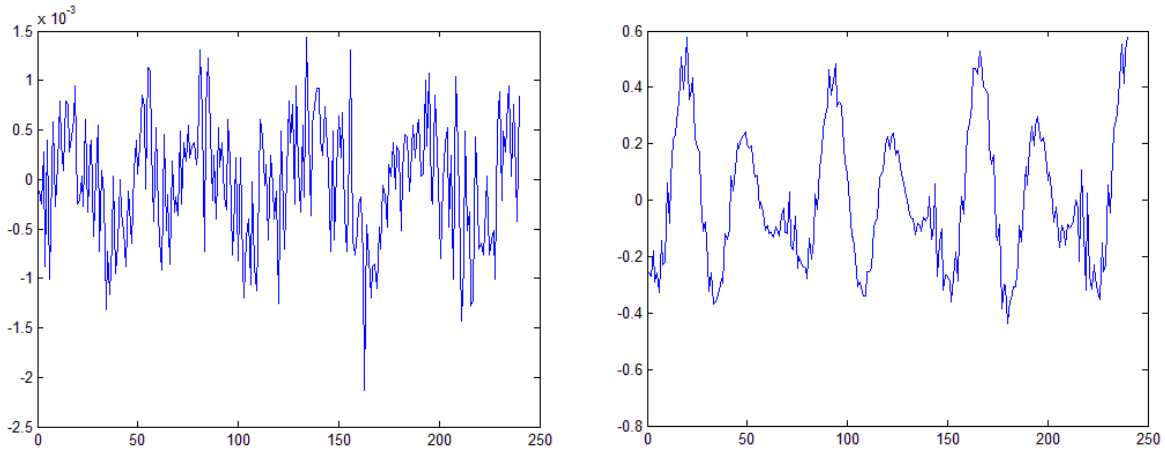
Para este fin existen tres métodos, el cálculo de la energía, la tasa de cruces por cero (ZCR), y el cálculo del pitch.

1. En primer lugar las tramas sonoras tienen mayor energía que las sordas o silenciosas. La energía de una determinada trama x se define en la expresión (2.3) :

$$E = \sum_{i=-\infty}^{\infty} x^2 [i] \qquad (2.3)$$

Es necesario definir un umbral de energía para discernir si un determinado coeficiente de energía pertenece o no a una trama sonora. Para dar estabilidad al cálculo del parámetro, además de eliminar la componente continua de la señal, también se ha normalizado entre 1 y -1.

2. Tasa de cruces por cero: La señal de audio, una vez eliminada la componente DC, pasa numerosas veces de ser positiva a negativa y viceversa. Sin embargo, el número de cruces es mucho mayor en las tramas sordas o en los silencios que en las tramas sonoras como se puede apreciar en la figura 2.8. El ZCR está acotado entre [0,1].



ZCR= 0,82 ZCR= 0,28

Figura 2.8: ZCR para trama sorda (izquierda) y sonora (derecha)

3. Cálculo del Pitch: Este método consiste en intentar extraer de la señal la frecuencia fundamental. Si esta frecuencia es detectada significa que la señal es cuasiperiódica y que por tanto pertenece a una trama sonora. El pitch se puede obtener de diversas formas, en este trabajo se ha realizado mediante el método SIFT [8] que se resume en filtrar la señal con un filtro LPC de orden 4, calcular la auto correlación del error de predicción y medir la distancia entre dos máximos consecutivos.

   En el presente proyecto se han desarrollado dos algoritmos de detección de tramas sonoras. El primero ha hecho uso de la tasa de cruces por cero y de la energía para aislar las tramas sonoras, los umbrales de decisión se han establecido empíricamente. El segundo algoritmo ha hecho uso del antes mencionado SIFT.

2.3. Extracción de coeficientes cepstrales

Una vez realizadas las operaciones de la sección 2.1, se obtienen de la señal únicamente las tramas sonoras (sonidos vocálicos), sin ruido en bajas frecuencias y con las altas frecuencias enfatizadas. A partir de aquí se van a explicar los pasos para obtener los coeficientes cepstrales mediante dos técnicas: utilizando análisis de Fourier (MFCC) y apoyándonos en los coeficientes de predicción lineal LPC que caracterizan el tracto vocal (LPCC).

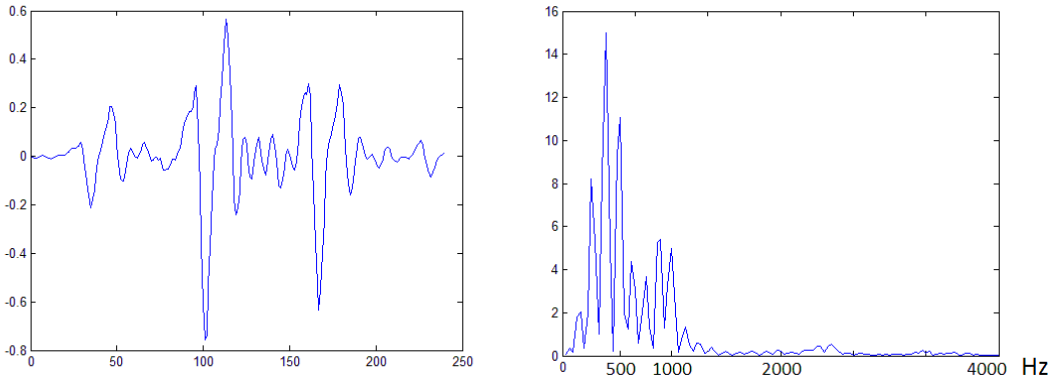
2.3.1. Mel Frequency Cepstral Coefficients [9]

1.-Transformada de Fourier de la señal

El primer paso es realizar la DFT expresada en (2.4) a la trama para obtener su contenido espectral, del que solo interesa la magnitud, descartando el plano complejo.

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk}; \qquad k = 0, \dots, N \qquad (2.4)$$

Para implementar la anterior expresión en Matlab se utiliza la FFT, que tiene menor complejidad computacional. En la figura 2.9, está representado el espectro en magnitud de una trama sonora



Trama sonora enventanada Magnitud del espectro

Figura 2.9

2.-Banco de filtros en escala Mel

El comportamiento del oído humano no es lineal con la frecuencia, por ello es efectivo muestrear el espectro de la señal siguiendo una escala que se aproxime a dicho comportamiento. La escala más utilizada para este propósito es la escala Mel, la cual tiene un comportamiento prácticamente lineal hasta los 1000 Hz y luego logarítmico de forma que los Hz reales son superiores a los Hz Mel, siguiendo la siguiente relación:

$$f_{MEL} = 2595\, log_{10}\left(\frac{1 + f_{Lineal}}{700}\right) \qquad (2.5)$$

Por tanto lo que se hace es realizar un muestreo del espectro con filtros triangulares de área unidad espaciados de acuerdo a la escala Mel. En Matlab se multiplica la magnitud del espectro por el banco de filtros representado en la figura 2.10.

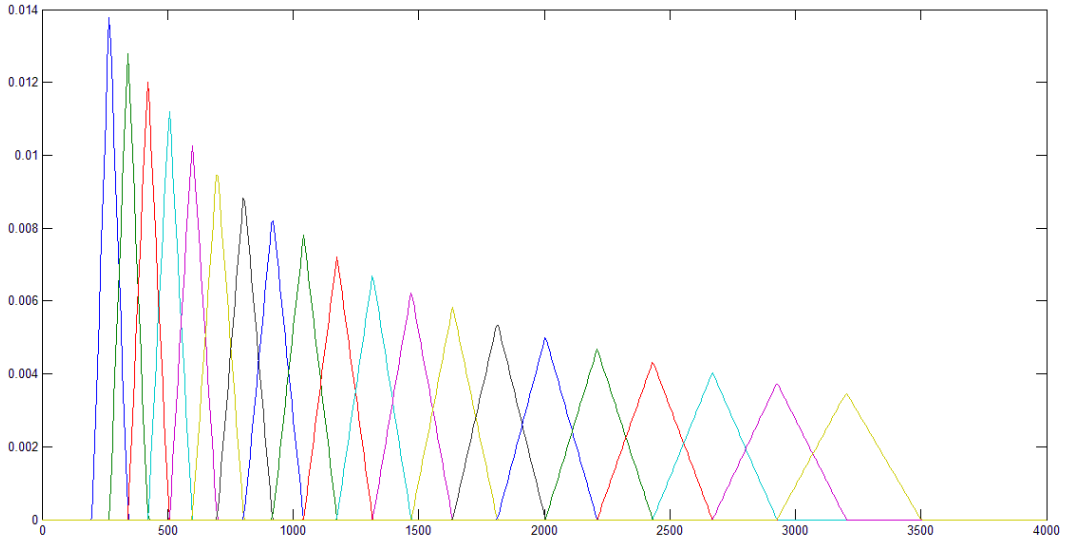


Figura 2.10: Banco de filtros triangulares implementado.

Un detalle importante del banco de filtros es que las frecuencias máximas y mínimas de cada filtro coinciden con la frecuencia central del filtro adyacente

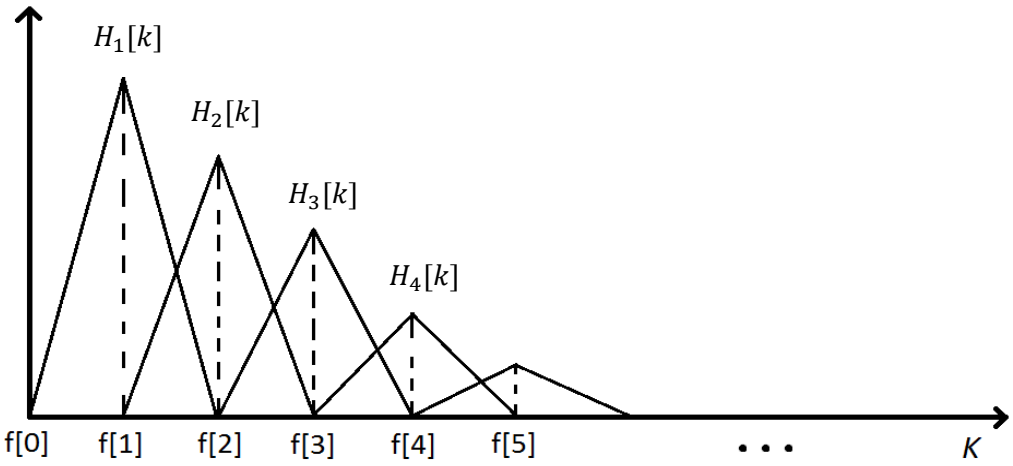


Figura 2.11

La forma triangular de los filtros está definida por:

$$H_m[k] = \begin{cases} 0; & k < f[m-1] \\ \frac{2(k-f[m-1])}{(f[m+1]-f[m-1])(f[m]-f[m-1])}; & f[m-1] \le k \le f[m] \\ \frac{2(f[m-1]-k)}{(f[m+1]-f[m-1])(f[m]-f[m-1])}; & f[m] \le k \le f[m+1] \\ 0; & k > f[m+1] \end{cases}$$

Los parámetros fundamentales del banco de filtros son las frecuencias de corte máxima y mínima y el número de filtros. Habitualmente el número de filtros está entre 20 y 40. El banco de filtros implementado tiene 28 filtros que están distribuidos en la escala mel, con una frecuencia mínima de 200 Hz y una máxima de 3500 Hz, si bien estos parámetros se pueden modificar.

Después de multiplicar la señal por el banco, se calcula la energía de cada uno de los filtros, de forma que obtenemos un número F de coeficientes.

$$E_m = \sum_{k=0}^{N-1} |X[k]|^2 H_m[k]; \qquad m = 1, \dots, F \qquad (2.6)$$

3.-Operador no lineal

La señal de excitación en tramas sonoras, que no es más que el pulso glotal, está convolucionada en el dominio temporal con la respuesta del tracto vocal. Al obtener el espectro mediante la transformada de Fourier, esta convolución se transforma en un producto que no nos permite separar ambas señales. Para poder separarlas de forma simple, es necesario un operador no lineal que transforme la multiplicación en una suma. Típicamente se ha utilizado el logaritmo como operador. En la actualidad hay debate sobre el operador más adecuado. En la universidad de Maryland por ejemplo, se realizó un estudio en el 2011 [10] utilizando un operador de naturaleza gaussiana, que consiguió mejorar la eficiencia en determinadas condiciones de ruido. Sin

embargo el sistema no fue más eficaz al introducir ruido de murmullo ("bubble noise"), el cuál está muy presente en el entorno de los dispositivos móviles. Por ello el operador elegido en el trabajo es el logaritmo.

$$E_{mlog} = log_{10}(E_m) \qquad (2.7)$$

4.- Discrete Cosine Transform (DCT)

Por último se realiza la transformada discreta del coseno para eliminar la alta correlación espectral entre coeficientes que tiene el logaritmo y pasar al dominio cepstral.

$$C_{MFCC}[m] = \sum_{k=0}^{N-1} E_{mlog}\, cos\left(m\left(k-\frac{1}{2}\right)\frac{\pi}{N}\right); \quad m = 1, \dots, F \qquad (2.8)$$

C_{MFCC} son los coeficientes cepstrales extraídos por el sistema, el número de coeficientes está dado por el número de filtros Mel aplicados en el espectro.

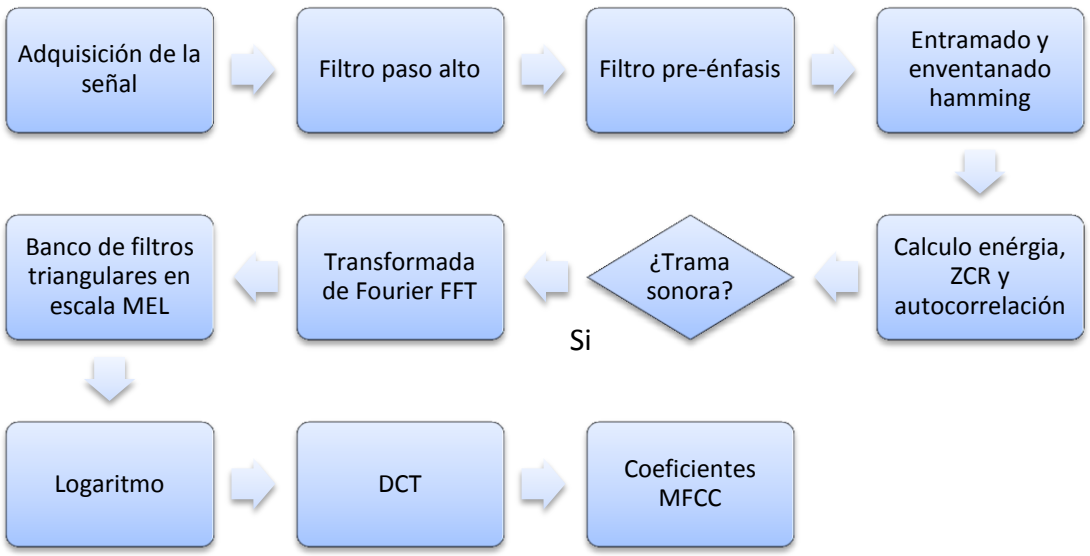


Figura 2.12: Proceso de extracción de los MFCC

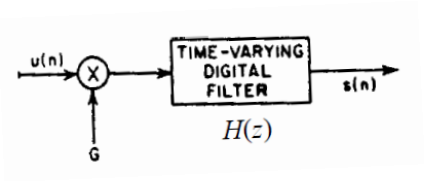
2.3.2. Linear Prediction Cepstral Coefficients

La adquisición, pre-procesamiento y enventanado de la señal es exactamente igual que para los MFCC. A partir de aquí:

1.- Cálculo de los coeficientes LPC

El primer paso es extraer los coeficientes a_k del filtro H(z) que modela el tracto vocal.

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \quad (2.9)$$



Aplicando la transformada Z inversa se llega a un sistema de N ecuaciones y N+P incógnitas, siendo N el número de muestras de una trama y P el orden del predictor. Este sistema es por tanto irresoluble.

La solución está en minimizar el error cuadrático medio de predicción que al final conduce a un sistema resoluble de P ecuaciones y P incógnitas. Todo este proceso esta bien documentado en la bibliografia [11]. En matlab la función para calcular los coeficientes viene ya implementada y resuelve el problema por el método de la autocorrelación, que consiste en hacer la derivada de la señal e igualar a 0 para llegar a las ecuaciones de Yule-Walker, este sistema es resuelto mediante el algoritmo recursivo de Levinson-Durbin que finalmente extrae los a_k .

2.- Transformación al dominio cepstral

Los coeficientes cepstrales pueden ser calculados a partir de los LPC, mediante el siguiente procedimiento recursivo [12]:

$$C_{LPCC}[1] = -a_1 \quad (2.10)$$

$$C_{LPCC}[m] = -a_k - \sum_{t=1}^{n-1} \left(1 - \frac{t}{m}\right) a_t \, C_{LPCC}[m-t]\,; \quad 1 < m \leq p \qquad (2.11)$$

$$C_{LPCC}[m] = -\sum_{t=1}^{p} \left(1 - \frac{t}{m}\right) a_t \, C_{LPCC}[m-t]\,; \quad p < m \qquad (2.12)$$

Como se puede ver el número m de coeficientes cepstrales habitualmente es mayor que el orden del filtro LPC, en un factor $Q = \left(\frac{P}{m}\right) \approx \frac{2}{3}$, en este trabajo el orden del filtro es 10 y la cantidad de coeficientes cepstrales 16, si bien puede ser variado.

El cálculo de los LPCC se ha implementado mediante el DSP Toolbox de Matlab.

2.3.3. Coeficientes delta

Además de los C_{MFCC} ó C_{LPCC} se incluyen otros coeficientes derivados de estos que permiten tener en cuenta la velocidad de coarticulación y en general la variabilidad entre pronunciaciones del locutor, por eso son llamados coeficientes dinámicos o delta. Los coeficientes delta, ΔC[13] se definen como la variación de los coeficientes cepstrales en un instante de tiempo. A su vez los coeficientes doble delta, $\Delta\Delta C$ se definen como la variación de los coeficientes delta en un instante de tiempo.

$$\Delta C_i[m] = \sum_{k=-l}^{l} \frac{k \, C_{i+k}[m]}{|k|} \qquad 1 \leq m \leq F \qquad (2.13)$$

$$\Delta\Delta C_i[m] = \sum_{k=-l}^{l} \frac{k \, \Delta C_{i+k}[m]}{|k|} \qquad 1 \leq m \leq F \qquad (2.14)$$

Dónde m representa el orden del coeficiente cepstral extraído en la trama i-ésima.

2.3.4. Cepstral Mean Variance Normalization

Para mejorar el sistema y hacerlo menos vulnerable al ruido y variabilidad en la grabación se ha utilizado una técnica llamada normalización cepstral de la media y la varianza [14]. Teniendo un total de T tramas sonoras, cada trama t tiene F coeficientes cepstrales. Se calcula la media y la varianza de cada vector representativo de la variación de un coeficiente Cepstral en el tiempo con un total de F vectores. Después se aplica la siguiente fórmula:

$$C_i^{CMVN}[m] = \frac{C_i[m] - \mu_m}{\sigma_m^2} \tag{2.15}$$

Dónde la media y la varianza están definidas:

$$\mu_m = \frac{1}{T}\sum_{i=1}^{T} C_i[m] \tag{2.16}$$

$$\sigma_m^2 = \frac{1}{T-1}\sum_{i=1}^{T} (C_i[m] - \mu_m)^2 \tag{2.17}$$

$$m = 1, \dots, F$$

Capítulo 3

MÉTODOS DE COMPARACIÓN Y DECISIÓN: DTW

El siguiente paso es evaluar hasta que punto el vector de parámetros extraído en un test puede pertenecer a un determinado usuario que previamente ha sido entrenado en el sistema. Para ello, en la fase de entrenamiento, que más tarde será explicada, se crea un modelo característico para cada locutor. La técnica empleada para generar este modelo tiene que tener en cuenta la alta variabilidad de la voz, tanto en dominio temporal, dónde cambia la velocidad de articulación, como en dominio frecuencial. Como se ha visto en la sección 1.2.3 existen varias técnicas de modelado, entre ellas se ha elegido el DTW por su buen compromiso entre simplicidad, robustez y carga computacional.

3.1. Dynamic Time Warping

La señal de voz por su naturaleza transitoria, no es igual en el entrenamiento y en el test. Uno de los mayores problemas es la poca correlación temporal entre ambas señales, es decir, la pronunciación en el entrenamiento se puede realizar en un tiempo t, mientras que el test siempre se realizará en un tiempo t ± α, además las velocidades de coarticulación entre fonemas también varían. El fenómeno del desajuste temporal es especialmente crítico cuándo el sistema de modelado y decisión se basa en la comparación de plantillas. Por ello es necesario la alineación temporal de ambas señales, esta alineación se hace en el dominio cepstral, sobre los vectores de

coeficientes, ya que hacerlo sobre la propia señal de audio en dominio temporal alteraría la información espectral de la voz.

La alineación se consigue mediante un algoritmo de programación dinámica conocido como Dynamic Time Warping o DTW por sus siglas en inglés [15]. El proceso seguido se describe a continuación:

1. Se tienen dos matrices de parámetros como la representada en la figura 3.1 de tamaño $i \times n$, la primera corresponde al entrenamiento y la segunda al test. El número de columnas "n" representa el número de parámetros diferentes ($C_1, \ldots, \Delta C_1, \ldots, \Delta\Delta C_n$) extraídos de cada trama, el numero de filas es la cantidad i de tramas enventanadas y analizadas de la señal.

Coeficiente / Trama	C_1	C_2	C_3	…	$\Delta\Delta C_N$
1					
2					
3					
4					
5					
6					
7					
8					
9					
…					
i M					

Figura 3.1

2. Cada columna forma por tanto un vector que corresponde a la evolución de un determinado coeficiente cepstral en el tiempo. Estos vectores deben de ser parecidos en ambas matrices si el locutor ha sido el mismo, pero es complicado que los valores coincidan en el eje temporal. Por ello para realizar una comparación adecuada hay que alinear ambos vectores estirando o contrayendo uno de ellos en el tiempo y calcular el coste o distancia que se ha empleado para alinearlos.
3. Para ello se confecciona una matriz distancia de tamaño $M \times N$, donde M es la longitud del vector v_1 correspondiente al entrenamiento y N es la

longitud de v_2, correspondiente al test. Se almacena en dicha matriz la distancia entre los posibles caminos.

$$d(m,n) = (v_1[m] - v_2[n])^2 \qquad (3.1)$$

4. Se busca un camino de menor distancia para llegar desde el punto (0,0) de la matriz hasta (M,N). Se establecen restricciones locales y globales. Las llamadas restricciones locales hacen referencia a las opciones de movimiento que existen desde un punto determinado, que en el ejemplo están definidas en la figura 3.2. La restricción global es el trazado de un paralelepípedo (superficie rayada) diagonal a la matriz que delimita la desviación máxima permitida.

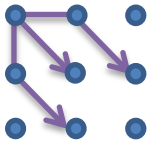


Figura 3.2: restricción local de movimiento

En la siguiente matriz se pueden ver las distancias calculadas y el camino óptimo para dos vectores de ejemplo $v_1 = \{1,2,3,4,3,2,1,0\}$ y $v_2 = \{1,1,2,3,4,4,3,3,2,0\}$

v_2 \ v_1	1	2	3	4	3	2	1	0
1	0	1	4	9	4	1	0	1
1	0	1	4	9	4	1	0	1
2	1	0	1	4	1	0	1	4
3	4	1	0	1	0	1	4	9
4	9	4	1	0	1	4	9	16
4	9	4	1	0	1	4	9	16
3	4	1	0	1	0	1	4	9
3	4	1	0	1	0	1	4	9
2	1	0	1	4	1	0	1	4
0	1	4	9	16	9	4	1	0

Figura 3.3

5. Se hace el sumatorio de todas las distancias por el camino más corto (línea azul). El resultado es un valor de distancia total D, que mide la diferencia entre un vector y otro.
6. Se repite el mismo alineamiento para todos los vectores que representan la evolución de cada uno de los coeficientes en el tiempo (figura 3.1) y se hace un sumatorio de todas las distancias obtenidas para obtener una distancia global. El valor D_{Global} será menor en el caso que el parámetro haya sido extraído del mismo locutor, y mayor si el entrenamiento y test ha sido realizado por diferentes locutores. Se establece un valor umbral a partir del cual el sistema acepta o rechaza el test.

A continuación se puede ver el alineamiento realizado para dos señales senoidales con distinta frecuencia y fase:

- $Sin_{azul} = \sin(x)$
- $Sin_{rojo} = \sin(1.1x + 3.8)$

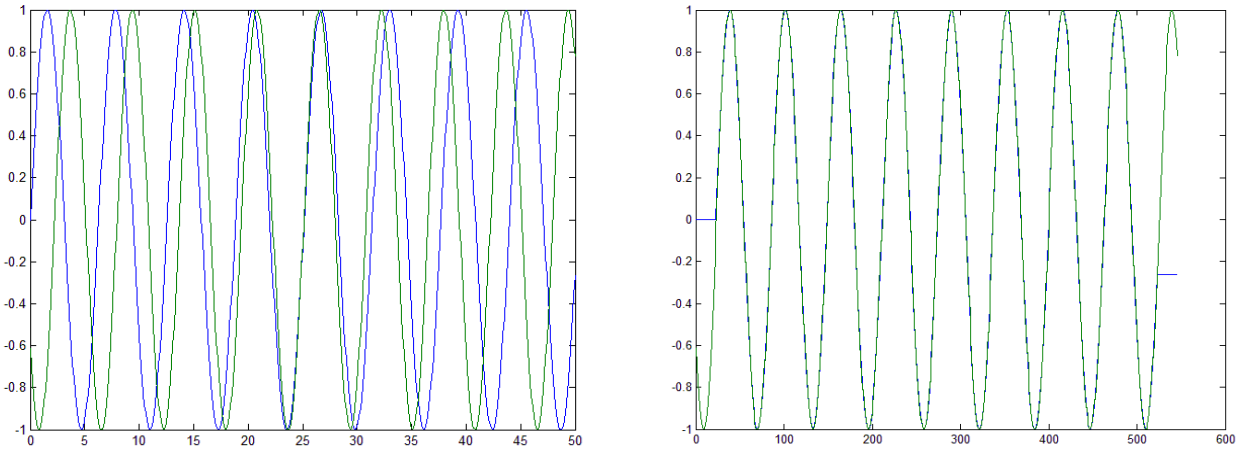


Figura 3.4: senoidales antes y después del algoritmo DTW

En la figura 3.4 se puede ver la función de alineamiento que hace el camino más corto entre los dos vectores. En este caso como la frecuencia del seno es constante (factor 1.1), la función de alineación es lineal (línea recta).

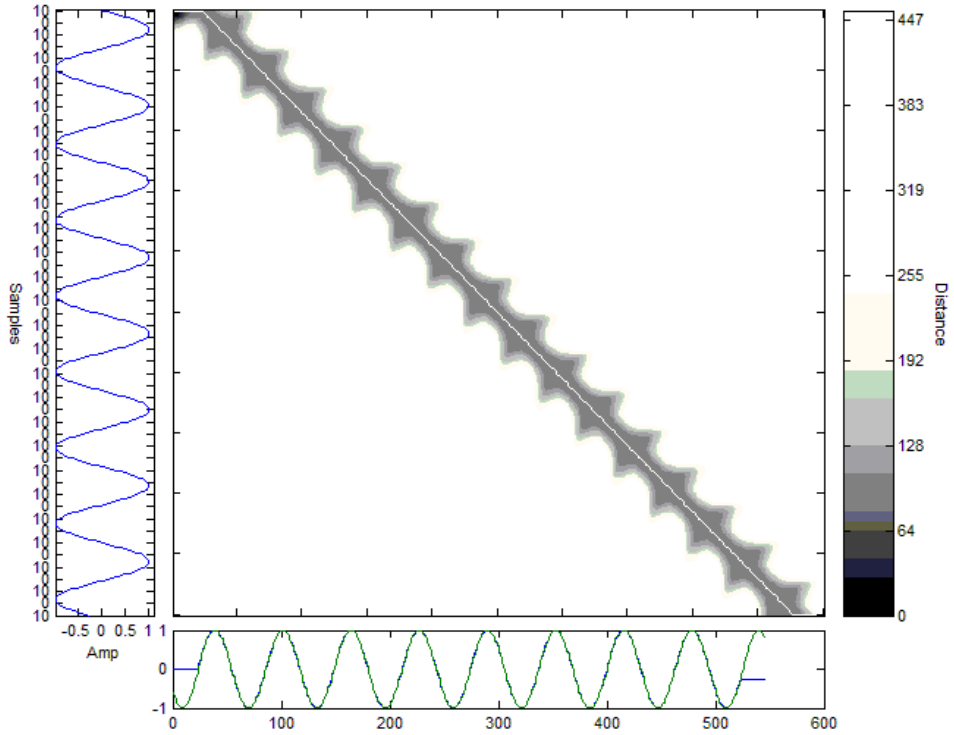


Figura 3.5: función de alineación

En la figura 3.6 se representa la alineación real del tercer coeficiente cepstral, para un entrenamiento y test de los dígitos “1,3,7”:

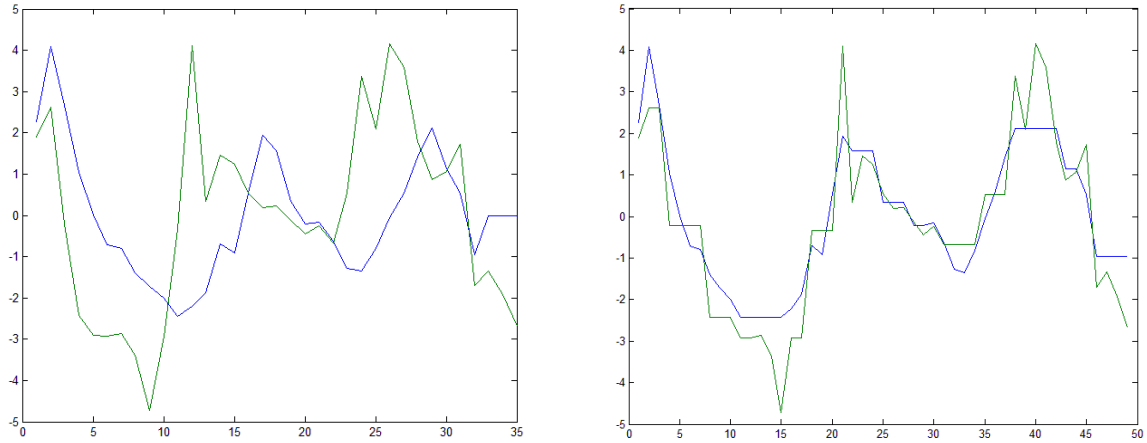


Figura 3.6

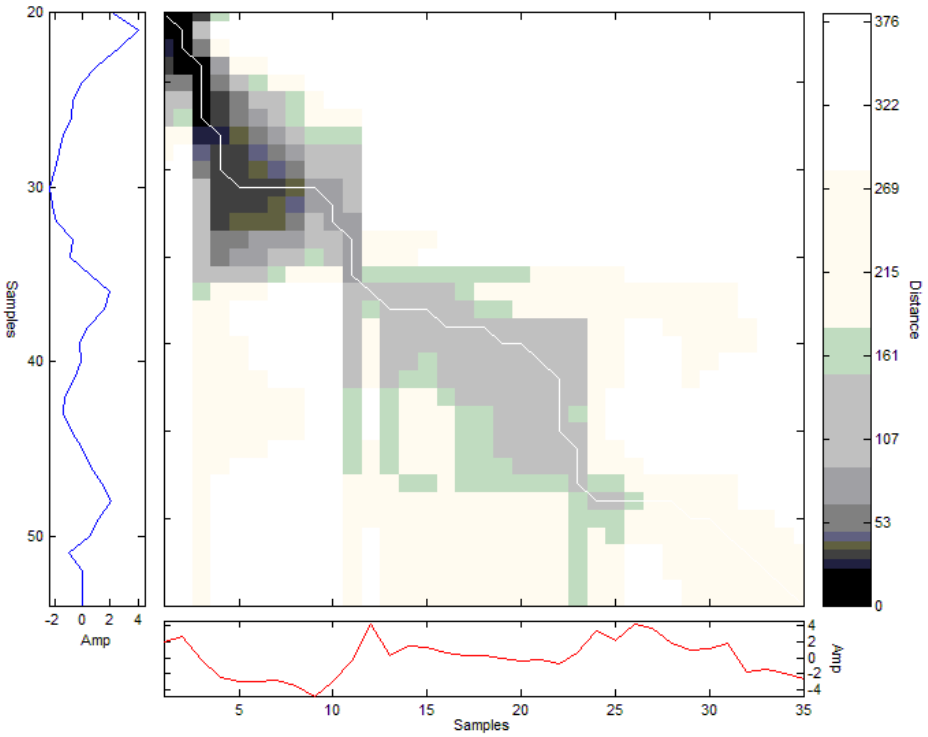


Figura 3.7: función de alineación para figura 3.6

Capítulo 4

ALGORITMOS DESARROLLADOS

Todo el sistema explicado en los dos capítulos anteriores se ha desarrollado con el software matemático Matlab en su última versión: r2013b. El sistema se ha querido entender como un grupo de funciones más sencillas relacionadas adecuadamente. Esto permite que el código sea más fácil de entender, corregir y modificar. Cada función desempeña un papel concreto que va ser explicado a continuación.

4.1. Scripts y funciones para el análisis MFCC

- ***Testeo_mfcc_bueno.m (Script a ejecutar)***

  Recorre los modelos de los 10 locutores comparando para cada uno de ellos su modelo con 10 tests genuinos (de ellos mismos) y 10 impostores (del resto de locutores). "bueno" hace referencia a que todas las comparaciones son con el código correcto

- ***Testeo_mfcc_codigoimpostor.m (Script a ejecutar)***

  Recorre los modelos de los 10 locutores comparando para cada uno de ellos su modelo con 10 tests de ellos mismos pronunciando el código correcto y con 10 impostores (del resto de locutores) pronunciando un código incorrecto.

- ***entrenamiento.m***

  Esta función entrena el modelo de un determinado locutor, determina el umbral adecuado y decide si un test pertenece o no a dicho modelo. Hay que introducir por tanto las 8 señales de audio para el entrenamiento y umbral y la señal de test. La salida es binaria: aceptación o rechazo.

- ***extraccion_MFCC.m***

  El objetivo de esta función es extraer la matriz de coeficientes para una determinada señal de audio. Se extraen tanto los coeficientes cepstrales normalizados como los deltas.

  - ***sonorapitch.m***

    Esta función extrae la señal únicamente con las tramas sonoras, el resto de tramas las sustituye por ceros, de forma que la longitud total de la señal es la misma. El método utilizado para segmentar las tramas es la detección del pitch por SIFT.

  - ***deteccciontsonora.m***

    Esta función extrae la señal únicamente con las tramas sonoras, el resto de tramas las sustituye por ceros, de forma que la longitud total de la señal es la misma. El método utilizado para segmentar las tramas es el de la energía y el ZCR, cuyos umbrales se han determinado empíricamente y se pueden cambiar desde la misma función.

  - ***MFCC.m***

    Son extraídos los coeficientes MFCC de la señal introducida. Estos coeficientes no están normalizados aún.

  - ***trifbank.m****

    Esta función se utiliza para crear el banco de filtros triángulares, se puede especificar el intervalo frecuencial deseado, el número de filtros deseado dentro de ese umbral, y otros parámetros como la normalización de los filtros triangulares a una altura unidad o a un área unidad.

  - ***CMVN2.m***

    Realiza la normalización cepstral de la media y la varianza de los coeficientes extraidos en MFCC.m

  - ***DIFMFCC.m***

    Calcula los coeficientes diferenciales deltas, de velocidad y aceleración a partir de los MFCC ya normalizados que devuelve la función CMVN2.

**Función implementada por Kamil Wojcick*

- ***vectorcarac.m***

  Concatena las matrices de coeficientes MFCC los deltas y los dobles deltas. Es el final de la función extraccion_MFCC.m.

- ***dtw2.m***

  Esta función se utiliza para entrenar el modelo del locutor y para establecer el umbral al que se aceptará a un usuario. Los parámetros de entrada son las matrices de coeficientes de los 8 segmentos de audio del locutor a entrenar.

- ***dtw4.m***

  Compara la señal del test con el modelo entrenado establecido en dtw2.m. La salida es una puntuación de parecido.

- ***dtw3.m****

  Es el algoritmo de DTW que ajusta los vectores y minimiza las distancias entre ellos. Esta función la utilizan dtw2.m y dtw4.m

4.2. Scripts y funciones para el análisis LPCC

- ***Testeo_lpcc_bueno.m (Script a ejecutar)***

  Recorre los modelos de los 10 locutores comparando para cada uno de ellos su modelo con 10 tests genuinos (de ellos mismos) y 10 impostores (del resto de locutores). “bueno” hace referencia a que todas las comparaciones son con el código correcto

- **Testeo_lpcc_codigoimpostor.m (Script a ejecutar)**

  Recorre los modelos de los 10 locutores comparando para cada uno de ellos su modelo con 10 tests de ellos mismos pronunciando el código correcto y con 10 impostores (del resto de locutores) pronunciando un código incorrecto.

- ***Entrenamiento_lpcc.m***

  Esta función entrena el modelo de un determinado locutor, determina el umbral adecuado y decide si un test pertenece o no a dicho modelo. Hay que introducir por tanto las 8 señales de audio para el entrenamiento y umbral y la señal de test. La salida es binaria: aceptación o rechazo.

- ***extraccion_LPCC.m***

El objetivo de esta función es extraer la matriz de coeficientes para una determinada señal de audio. Se extraen tanto los coeficientes cepstrales normalizados como los deltas.

- ***sonorapitch.m***

  Esta función extrae la señal únicamente con las tramas sonoras, el resto de tramas las sustituye por ceros, de forma que la longitud total de la señal es la misma. El método utilizado para segmentar las tramas es la detección del pitch por SIFT.

- ***detecciontsonora.m***

  Esta función extrae la señal únicamente con las tramas sonoras, el resto de tramas las sustituye por ceros, de forma que la longitud total de la señal es la misma. El método utilizado para segmentar las tramas es el de la energía y el ZCR, cuyos umbrales se han determinado empíricamente y se pueden cambiar desde la misma función.

- ***LPCC2.m***

  Son extraídos los coeficientes LPCC de la señal introducida. Estos coeficientes no están normalizados aún.

- ***CMVN2_lpcc.m***

  Realiza la normalización cepstral de la media y la varianza de los coeficientes extraidos en MFCC.m

- ***DIFMFCC.m***

  Calcula los coeficientes diferenciales deltas, de velocidad y aceleración a partir de los MFCC ya normalizados que devuelve la función CMVN2.

- ***vectorcarac.m***

  Concatena las matrices de coeficientes MFCC los deltas y los dobles deltas. Es el final de la función extraccion_MFCC.m.

- ***dtw2.m***

  Esta función se utiliza para entrenar el modelo del locutor y para establecer el umbral al que se aceptará a un usuario. Los parámetros de entrada son las matrices de coeficientes de los 8 segmentos de audio del locutor a entrenar.

- ***dtw4.m***

  Compara la señal del test con el modelo entrenado establecido en dtw2.m. La salida es una puntuación de parecido.

- ***dtw3.m****

  Es el algoritmo de DTW que ajusta los vectores y minimiza las distancias entre ellos. Esta función la utilizan dtw2.m y dtw4.m

Como se puede ver muchas funciones son llamadas en ambos métodos de análisis

**Función implementada por Pau Mic y otros.*

Capítulo 5

DISPOSITIVO EXPERIMENTAL

Como se ha comentado anteriormente es importante mejorar al máximo posible el modelo entrenado por el sistema. Para ello hay dos opciones: o bien la longitud del segmento de audio (frase, dígitos) es mayor, o el número de repeticiones es mayor. Como el sistema está enfocado a dispositivos móviles se ha optado por un código de 4 dígitos a modo de PIN code, dónde cada digito puede tomar valores del 0 al 9. La información fonética contenida en 4 dígitos es pobre y por tanto el entrenamiento se ha llevado a cabo introduciendo estos dígitos en más de una ocasión. Este capítulo pretende detallar como se ha realizado el entrenamiento, como se ha fijado el umbral de decisión y en general como se ha planteado el testeo de los algoritmos para obtener las tasas de error.

5.1. Entrenamiento

Para entrenar a un determinado usuario en el sistema desarrollado son necesarias 8 repeticiones del código PIN. Las primeras 4 sirven para hacer un promedio de todos los coeficientes extraídos y con ello crear un modelo más robusto, o por decirlo de otra forma que tenga más validez para distintas variaciones de la locución. Las otras 4 se utilizan para fijar el umbral, ya que fijar un umbral a partir de un modelo entrenado y las locuciones que han servido para definir ese modelo es altamente engañoso, tendiendo a resultar los umbrales de decisión en valores demasiado bajos.

Como hemos dicho el modelo se entrena a partir de cuatro repeticiones, las cuales se pueden relacionar entre ellas según el grafo de la figura 5.1

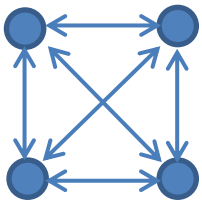


Figura 5.1

Para cada locución se forma su matriz de coeficientes variantes en el tiempo, y se comparan mediante DTW todas ellas, un total de 6 veces (líneas de la figura). Esto se hace así para cada coeficiente característico, es decir, si la matriz que caracteriza al locutor tiene 60 coeficientes entre cepstrales y deltas habría un total de 6*60= 360 accesos al algoritmo de DTW.

Para cada uno de estos accesos en los que se compara la evolución de pares de coeficientes en el tiempo, se almacena su vector medio y la distancia entre los vectores comparados. Por tanto obtenemos un total de 6*60=360 vectores de media y 360 distancias.

Por último para cada uno de los 60 coeficientes vamos viendo en cual de las 6 comparaciones la distancia ha sido menor y accedemos a su correspondiente vector medio, asignando este vector al coeficiente correspondiente del modelo final entrenado.

Se puede deducir que el modelo final está compuesto de las 6 locuciones, cada vector correspondiente a un coeficiente será el vector medio de comparación que tenga la mínima distancia entre las 6 posibles comparaciones para ese coeficiente.

5.2. Umbral

Las otras 4 locuciones se utilizan para fijar el umbral. Simplemente se trata de comparar el modelo anteriormente entrenado con cada una de estas cuatro nuevas locuciones para obtener sus correspondientes distancias totales. Estas distancias totales son el sumatorio de las distancias correspondientes a cada vector representativo de un coeficiente. En el ejemplo hay que sumar 60 distancias para

obtener la distancia total. Después se realiza la media aritmética y la desviación típica de las 4 distancias y se define el umbral como la suma de la media y la desviación típica. Cualquier puntuación por encima de ese valor será rechazado por el sistema.

$$media = \overline{x} = \frac{D_1 + D_2 + D_3 + D_4}{4} \tag{5.1}$$

$$Umbral = \overline{x} + \sqrt{\frac{1}{4-1}\sum_{i=1}^{4}(D_i - \overline{x})^2} \tag{5.2}$$

Para elaborar las curvas ROC es necesario probar el sistema ante distintos niveles de umbral

5.3. Test

Consiste sencillamente en enfrentar cada una de las locuciones de test al modelo entrenado y ver si supera el umbral o no lo hace.

5.4. Planteamiento del testeo

Se tienen un total de 10 sujetos, cada uno de ellos ha realizado un total de 28 locuciones. 8 de ellas para entrenar el modelo y fijar el umbral, pronunciando la contraseña válida de entrada al sistema que se ha establecido en “4-8-9-5”. 10 más pronunciando esta misma secuencia para realizar el test verídico, es decir el que el sistema tiene que aceptar. Y otros 10 pronunciando un PIN erróneo, en concreto “7-9-6-1”.

Se realizarán por tanto un total de 10*10=100 teses verídicos, en los que el locutor es quién dice ser. Por otro lado cada modelo de los 10 locutores se enfrentara a un test con la contraseña correcta de cada uno de los demás, es decir cada locutor será comparado con otros 9 teses impostores, se han cogido dos locuciones para uno

de los 9 sujetos a comparar con el objetivo de que haya 10 comparaciones por cada locutor para obtener un total de 100 teses impostores.

Para obtener las curvas ROC se han repetido las anteriores operaciones con un total de 10 umbrales diferentes. La apariencia contínua de las curvas se debe a que se ha realizado una interpolación mediante polinomios de Hermite introduciendo el método '*pchip*' en la función *interp1* ya implementada en Matlab.

```
FAA = interp1(1:1:12,FA(1,:),0.01:0.01:12,'pchip');
```

Por último se evaluará la robustez del sistema en el caso de que los impostores no sepan el PIN, comparando cada modelo con 10 teses impostores con PIN erróneo procedentes del resto de locutores.

Capítulo 6

RESULTADOS

5.1. Optimización de los parámetros de análisis

Hay ciertos parámetros que se utilizan en el análisis de audio de los dos métodos desarrollados, MFCC y LPCC. El objetivo de esta primera parte del testeo es determinar los valores óptimos que disminuyan la probabilidad de error.

Los parámetros a analizar van a ser:

- Overlap
- Tamaño de ventana
- Método de detección de trama sonora

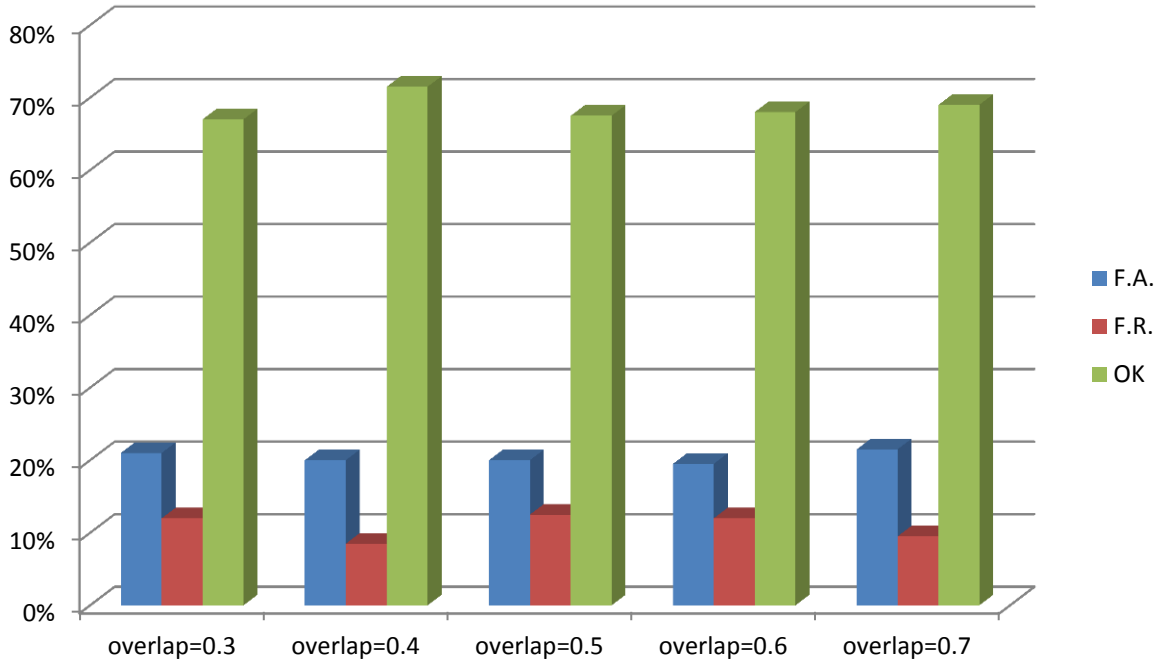


Figura 6.1

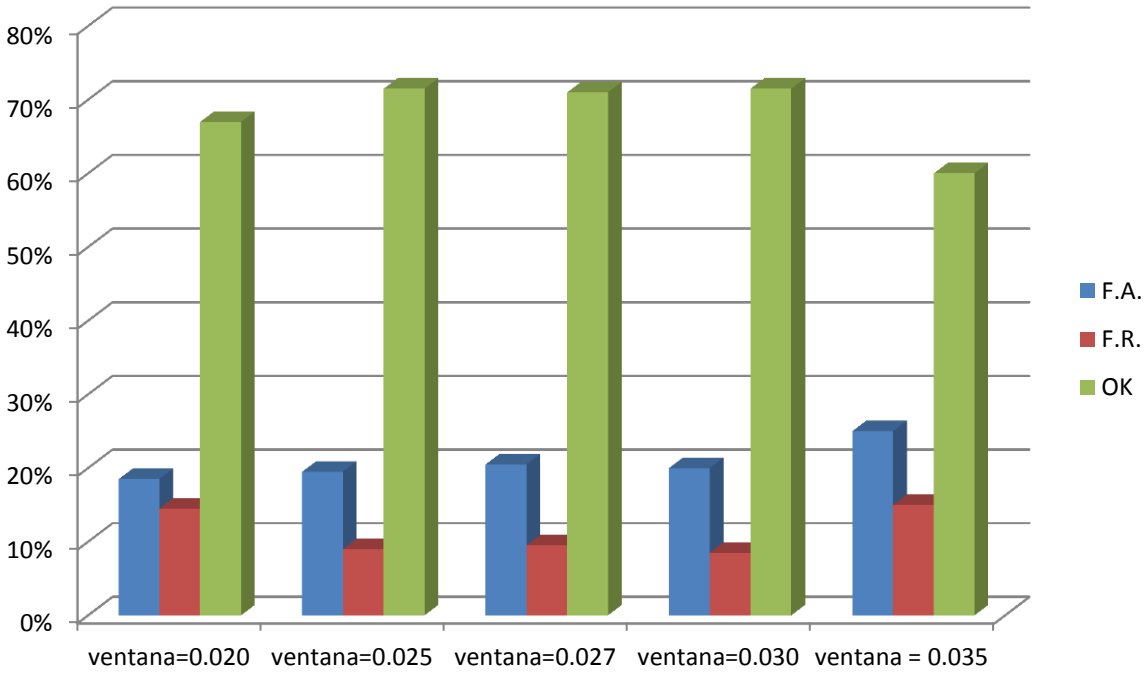


Figura 6.2

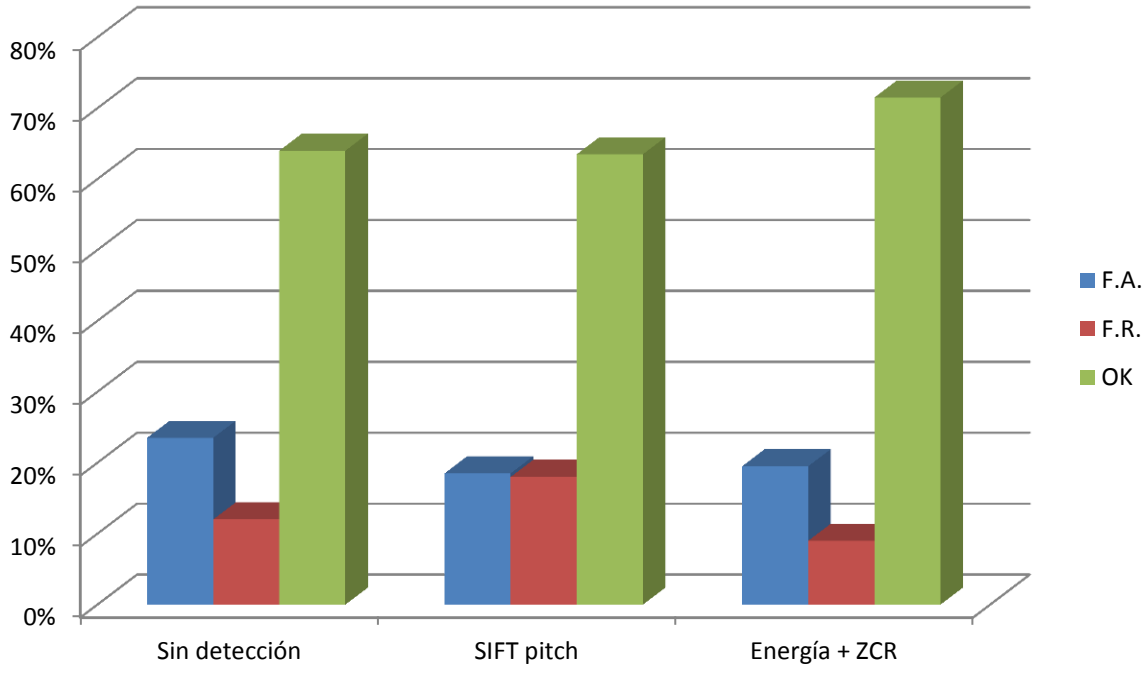


Figura 6.3

Se ve que el valor óptimo para el solapamiento entre ventanas de análisis es del 40%. Es decir si una ventana tiene una longitud de 240 muestras, la siguiente ventana a analizar empezaría en la muestra: 240*(1- 0.4)= 144 de la primera ventana. Esto como se ha dicho depende en gran medida de la velocidad de articulación.

La longitud de la ventana de análisis tiene una importancia más significativa, ya que si se establece una ventana temporal muy pequeña, se pierde demasiada resolución frecuencial y con ello la información espectral que nos permite caracterizar la señal de voz, por otro lado si se define una ventana demasiado grande se pierde la propiedad de cuasi-periodicidad de las tramas sonoras. Los tiempos de ventana óptimos se sitúan entre 25 y 30 ms.

En referencia al método de segmentación de tramas sonoras se confirma la hipótesis de que es necesario un segmentado de las tramas sonoras, ya que son estas las que contienen información relevante a nivel espectral para caracterizar una determinada señal de voz. Es destacable la diferencia entre los dos métodos probados, siendo mucho mejor el funcionamiento del algoritmo que utiliza la energia y el ZCR. El problema puede ser que no se haya definido bien el umbral de búsqueda del pitch en la función de autocorrelación del residuo, y como consecuencia la señal queda demasiado recortada:

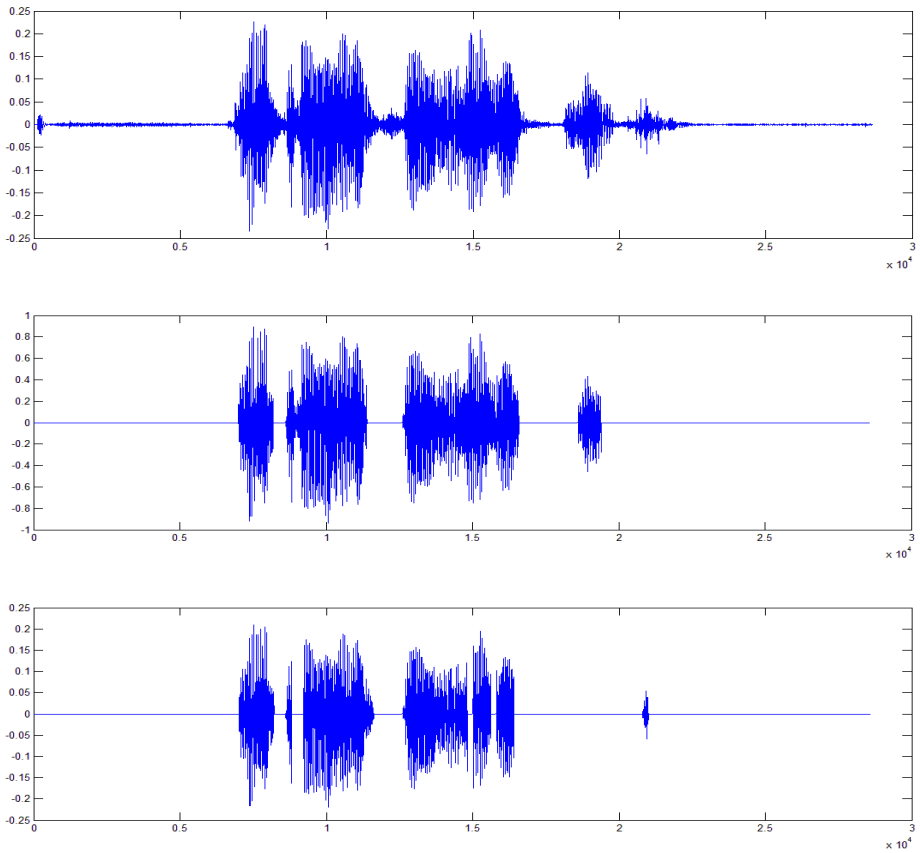


Figura 6.4: **Arriba** - señal sin segmentar; **Medio** – segmentado por análisis de energía y ZCR; **Abajo** – segmentado mediante detección de Pitch (SIFT).

5.2. Respuesta del sistema para coeficientes MFCC

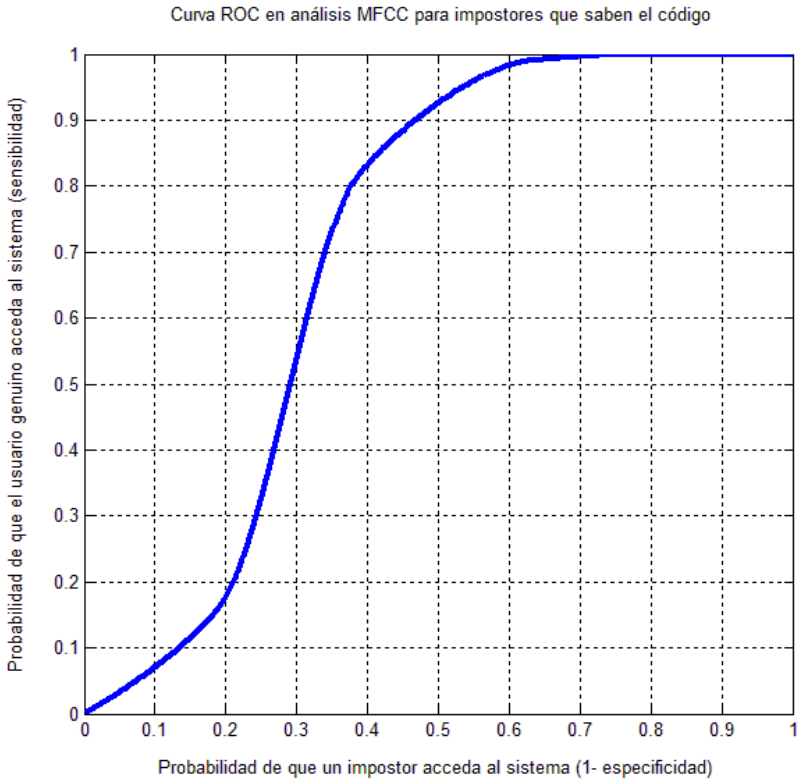


Figura 6.5

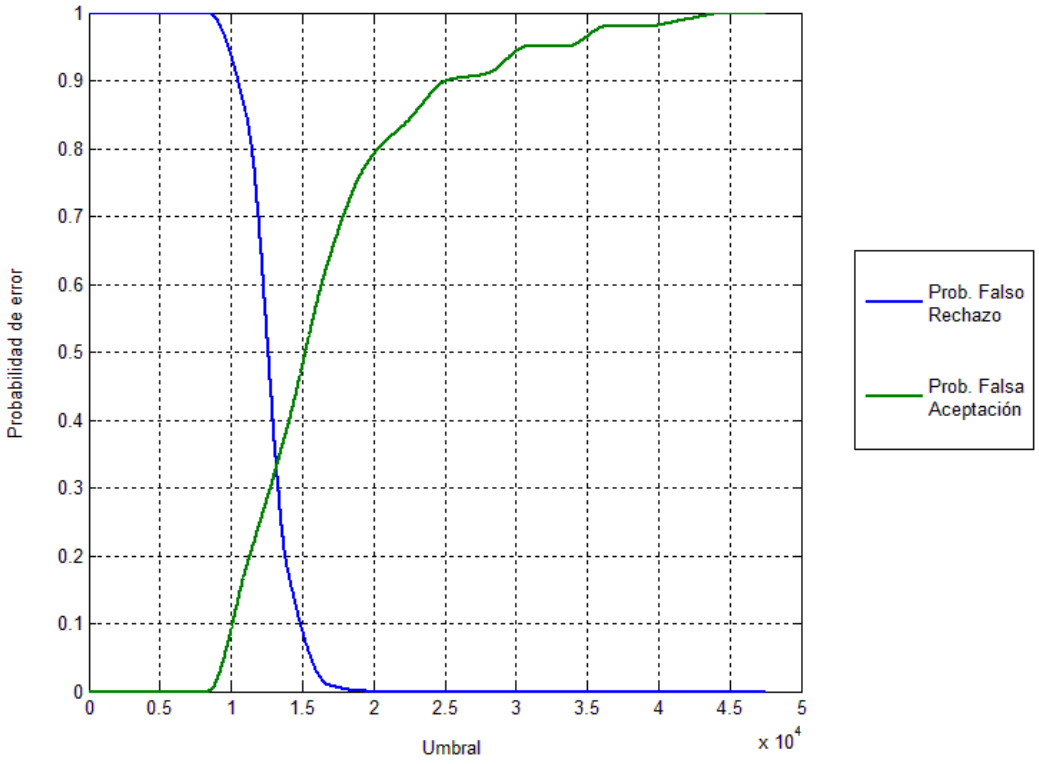


Figura 6.6

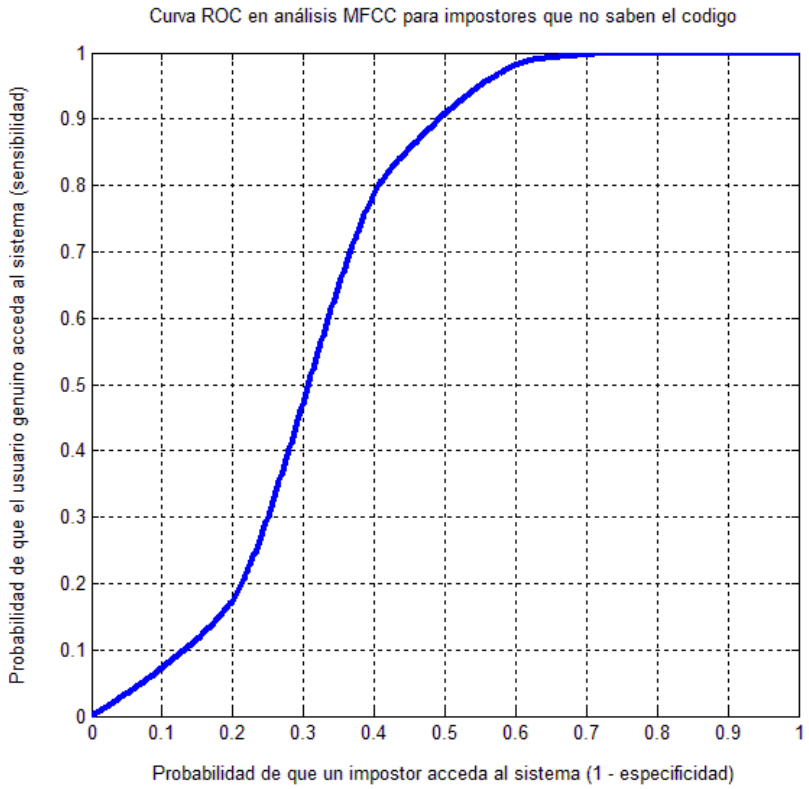


Figura 6.7

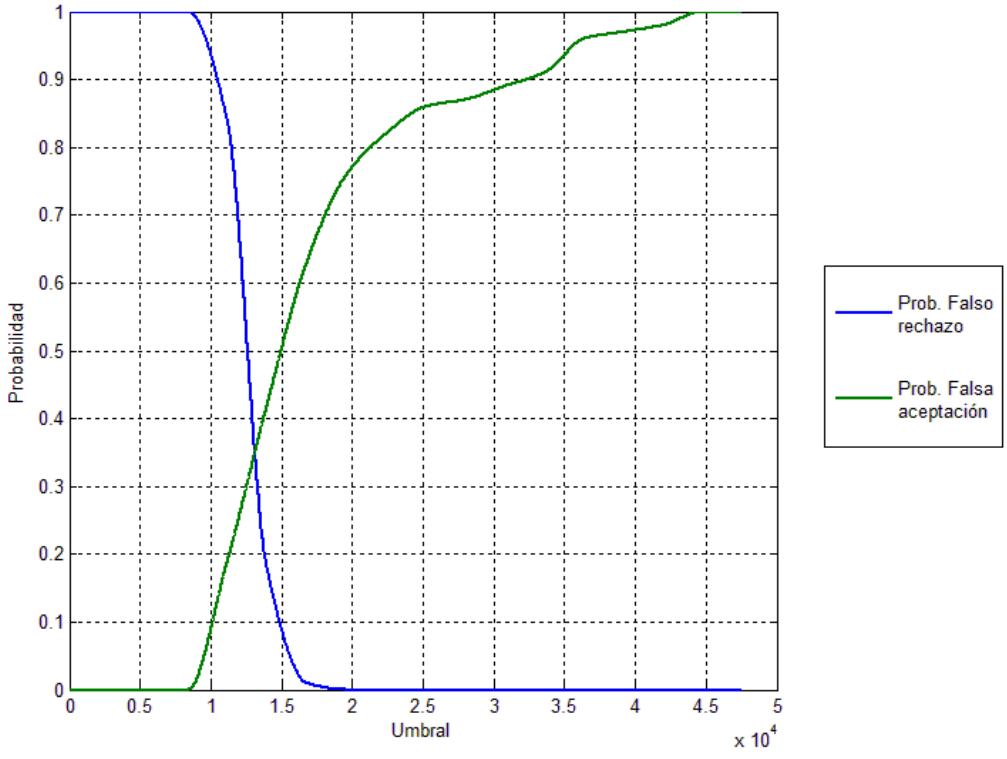


Figura 6.8

5.3. Respuesta del sistema para coeficientes LPCC

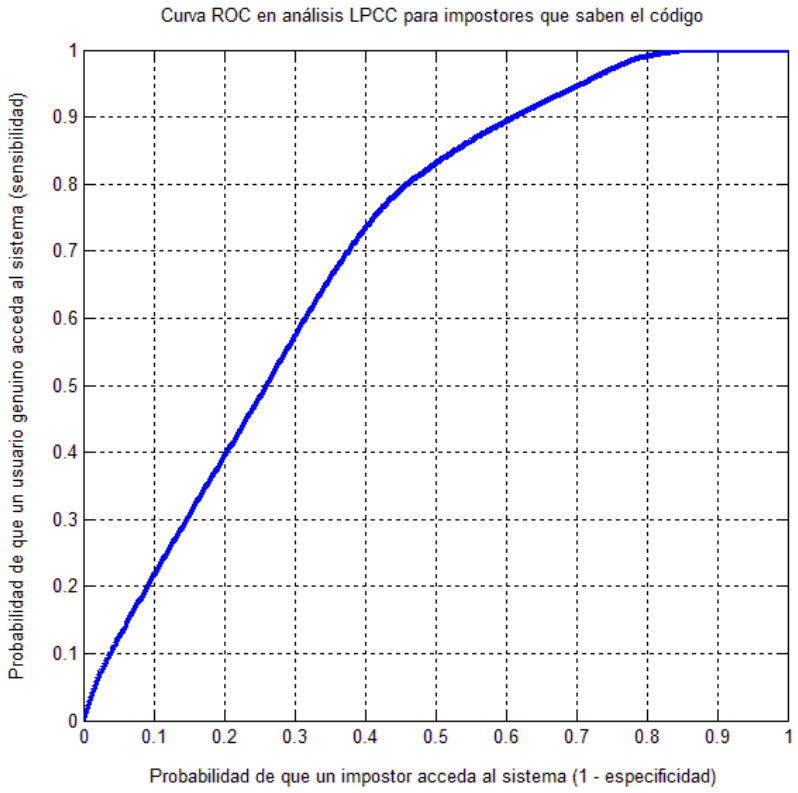


Figura 6.9

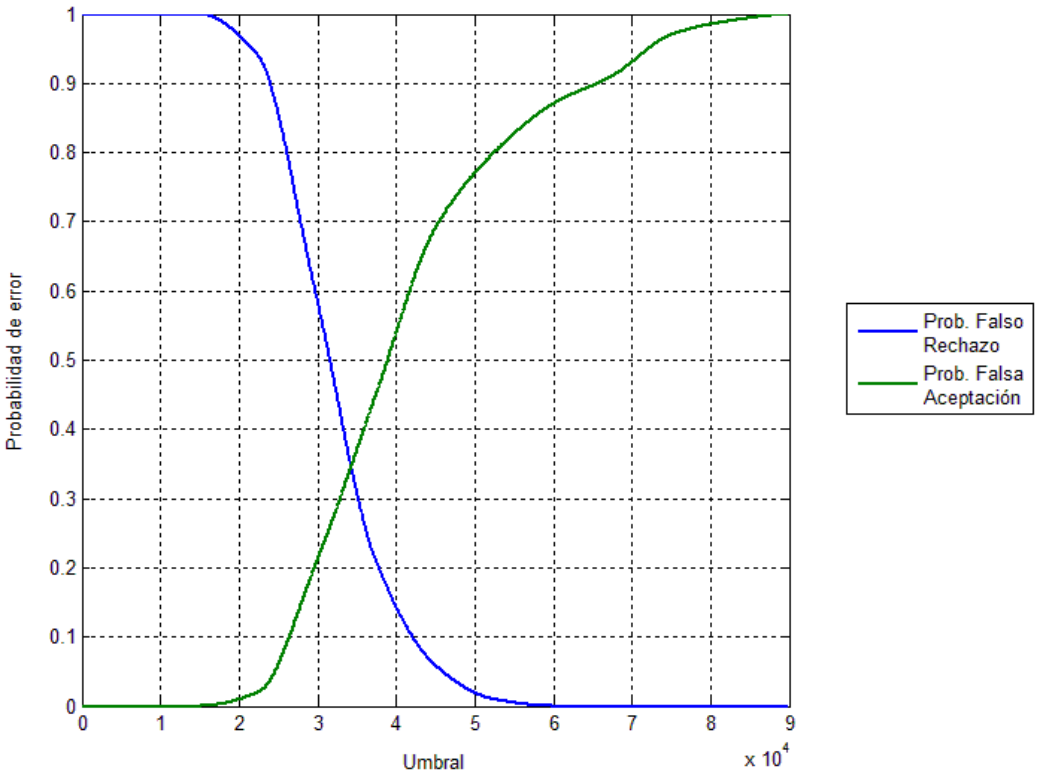


Figura 6.10

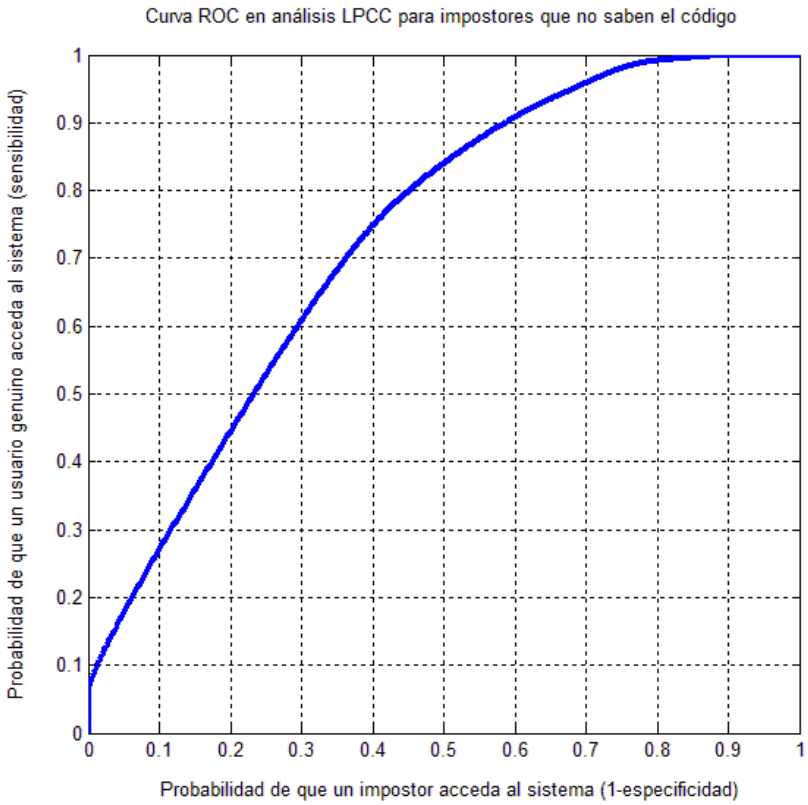


Figura 6.11

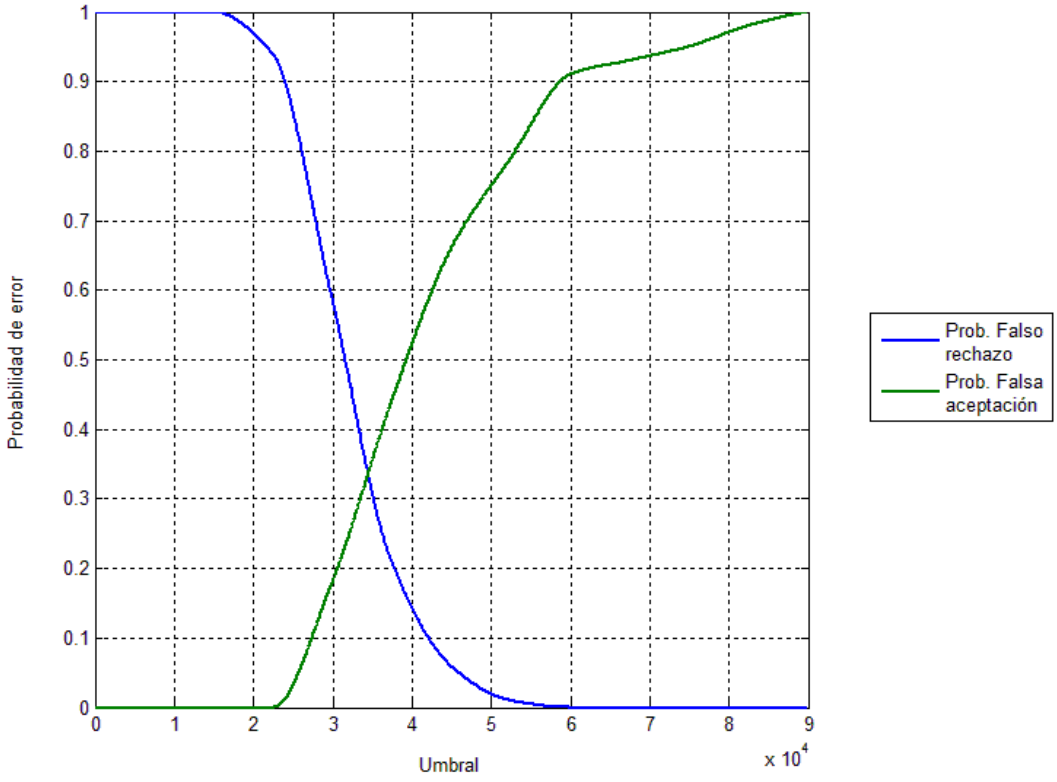


Figura 6.12

Capítulo 7

CONCLUSIONES

En vista de los resultados obtenidos se ha llegado a las siguientes conclusiones:

- Es necesario realizar un segmentado de tramas sonoras, siendo el mejor método para dicha tarea el de análisis de energía + ZCR.
- Los valores óptimos para el tamaño de ventana y el solape entre ventanas de análisis son del 40% y 25ms respectivamente para una velocidad de articulación estándar.
- El bloque de extracción de características se ha implementado correctamente ya que se obtienen tasas de error similares para la caracterización MFCC como para LPCC.
- El área bajo la curva ROC es algo mayor en el sistema LPCC que en MFCC y por tanto tiene un mejor rendimiento, además la fdp para los dos tipos de errores se asemeja más a una normal con LPCC, lo que siempre es deseable.
- La precisión del sistema es inferior a la esperada, para la mejor forma de caracterización que es LPCC y poniendo el mejor umbral la precisión es del 73% y por lo tanto la probabilidad total de error, suma de las probabilidades de falso rechazo y de falsa aceptación es del 27%. Esto se debe en primer lugar a la pequeña longitud del segmento de audio utilizado (aproximadamente 1s.) que tiene poca información fonética de la voz y por tanto contiene poca información espectral, la cual es insuficiente para caracterizar de forma robusta a un determinado locutor. En segundo lugar la calidad del audio no es la mejor, las grabaciones se tomaron con ruido de fondo en algunos casos. Por último queda en entredicho la capacidad del DTW como método clasificador.

- En comparación con LPCC para MFCC variaciones pequeñas del umbral tienen gran repercusión en el tipo de error que va predominar en el sistema, es decir LPCC tiene menos sensibilidad a variaciones del umbral.
- El comportamiento del sistema es más robusto para impostores que no saben el código que para los que sí lo saben.
- A día de hoy es muy complicado construir una aplicación comercial lo suficientemente fiable y rápida de este tipo, solo teniendo en cuenta el nivel espectral de la voz.

Capítulo 8

TRABAJOS FUTUROS

Se podría probar el sistema implementado con otras frases o códigos más largos y estudiar de forma analítica la importancia de la longitud del texto pronunciado en las tasas de reconocimiento y error.

Este trabajo también puede ser continuado con el diseño de nuevos métodos de entrenamiento del modelo del locutor. También se pueden utilizar otros métodos de comparación que no sean DTW, fundamentalmente los basados en análisis estadístico como los modelos ocultos de Markov y modelos de mezclas gaussianas.

Por otro lado sería interesante tener en cuenta niveles más altos del lenguaje, es decir estudiar de algún modo la forma en la que se producen los cambios entre fonemas.

También sería posible encadenar una versión mejorada de este sistema con otro que reconozca el mensaje pronunciado, es decir primero se comprobaría si el mensaje es correcto y luego si la persona que lo ha pronunciado es la que dice ser.

Capítulo 9

BIBLIOGRAFÍA

[1] Ken Phillips, Biometric identification comparison chart, PC week, Marzo 1997.

[2] Simo Huopio, Biometric Identification. Seminar on Network security: Authorization and Access Control in Open Network Environment, 1998.

[3] David Everett. Identity verification and biometrics. Computer Security Reference Book, pág 37.

[4] www.wikipedia.es

[5] Apuntes de PDVA. Tema 2. Generación de la voz. Universidad Pública de Navarra 2009 M.Zivanovic

[6] C. García, D. Tapias.-División de Tecnología del Habla.-Telefónica Investigación y Desarrollo, S.A. Unipersonal

[7] Daniel Jurafsky & James H. Martin. Speech and Languaje processing - Pearson International Edition Pág 329

[8] Markel 1972. Metodo basado en autocorrelacion para detectar tramas sonoras. Simplified Inverse Filtering Technique (SIFT).

[9] Thomas F. Quatieri. Speech Signal Processing-Principles and practice - Prentice Hall Pag.712

[10] Merit Fair 2011. Delta-Spectral Cepstral Coefficients for Robust Speaker Recognition; University of Maryland.

[11] Digital Speech Processing, Synthesis and recognition-second edition, Sadaoki Furui capítulo 5.

[12]Digital Speech Processing, Synthesis and Recognition-second edition, Sadaoki Furui, pág 66

[13] Ben Gold and Nelson Morgan. Speech and audio signal processing - Jhon Wiley & sons. Pág 300

[14] Koolwaaij, J. and Boves, L. (2000). Local normalization and delayed decision making in speaker detection and tracking. Digital Signal Processing, pag 113

[15] Digital Speech Processing, Synthesis and recognition, Sadaoki Furui, pág 266.

GLOSARIO

ANN: Artificial Neural Networks.

CMVN: Cepstral Median Variance Normalization.

DC: Direct Current. Nivel de continua de la señal.

DCT: Discrete Cosine Transform. Variante de la DFT.

DFT: Discrete Fourier Transform. Transformada para obtener el espectro de una señal.

DTW: Dynamic Time Warping.

DSP Toolbox: Digital Signal Processing Toolbox para Matlab.

EER: Equal Error Rate.

F.A.: Falsa Aceptación.

F.R.: Falso rechazo.

FFT: Fast Fourier Transform. Variante de la DFT con mayor eficiencia computacional.

FIR: Finite Impulse Response.

Fdp: Función de densidad de probabilidad.

Formantes: frecuencias amplificadas por el tracto vocal con mayor presencia espectral.

GMM-UBM: Gaussian Mixture Models – Universal Background Model.

HMM: Hidden Markov Models.

LPC: Linear Prediction Coefficients

LPCC: Linear Prediction Cepstral Coefficients

Mel: escala no lineal que pretende simular el comportamiento del oído humano.

MFCC: Mel Cepstrum Cepstral Coefficients.

Pitch: Frecuencia fundamental o tono de la señal de voz.

PDVA: Procesado Digital de Voz y Audio.

RAL: Reconocimiento Automático del Locutor.

RASTA: RelAtive SpecTrA.

ROC: Receiver Operating Characteristic

SIFT: Simplified Inverse Filtering Technique.

SIM: Subscriber Identity Module. Tarjeta que permite tener servicio de red telefónica.

Shazam: aplicación movil para la detección automática de canciones mediante análisis espectral.

VGA: Video Graphics Array, estándar antiguo de transmisión de video desarrollado por IBM con una resolución máxima de 600 x 800 píxeles.

ZCR: Zero Crossing Rate.

Anexo 1 – Código Matlab MFCC

Testeo_mfcc_bueno.m

```
%** PARÁMETROS EXTRACCIÓN MFCC*************************************
**********************************************************************
M=28;                       %numero de filtros del banco
M2=20;                      %numero de coef. cepstrales (primeros filtros
                            del banco creado)
R= [200 3500]               %mínimo y máximo del banco de filtros
deteccion_tsonoras=3;       %1=no hay deteccion de tramas sonoras,
                            2=detección por energía y ZCR, 3=detección por
                            busqueda de pitch (SIFT)
ventana=0.03;               %tamaño de la ventana de análisis en segundos
overlap=0.7;                % cantidad de solapamiento entre ventanas de
                            análisis, 0 solapamiento total, 1 no hay
                            solapamiento
%**********************************************************************
%**********************************************************************
tests1(1)={'1tv1'};
tests1(2)={'1tv2'};
tests1(3)={'1tv3'};
tests1(4)={'1tv4'};
tests1(5)={'1tv5'};
tests1(6)={'1tv6'};
tests1(7)={'1tv7'};
tests1(8)={'1tv8'};
tests1(9)={'1tv9'};
tests1(10)={'1tv10'};
tests1(11)={'2tv1'};
tests1(12)={'3tv1'};
tests1(13)={'4tv1'};
tests1(14)={'5tv1'};
tests1(15)={'6tv1'};
tests1(16)={'7tv1'};
tests1(17)={'8tv1'};
tests1(18)={'9tv1'};
tests1(19)={'10tv1'};
tests1(20)={'2tv2'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('1u1','1u2','1u3','1u4','1e1','1e2','1e3','1e4',
tests1(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(1,i)= {'OK'};
    else
        resultado1(1,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(1,i)= {'F.A.'};
    else
        resultado1(1,i)={'OK'};
    end
end
end
clear Disttotal umbral;
%----------------------------------------------------------------------
```

```
tests2(1)={'2tv1'};
tests2(2)={'2tv2'};
tests2(3)={'2tv3'};
tests2(4)={'2tv4'};
tests2(5)={'2tv5'};
tests2(6)={'2tv6'};
tests2(7)={'2tv7'};
tests2(8)={'2tv8'};
tests2(9)={'2tv9'};
tests2(10)={'2tv10'};
tests2(11)={'1tv2'};
tests2(12)={'3tv2'};
tests2(13)={'4tv2'};
tests2(14)={'5tv2'};
tests2(15)={'6tv2'};
tests2(16)={'7tv2'};
tests2(17)={'8tv2'};
tests2(18)={'9tv2'};
tests2(19)={'10tv2'};
tests2(20)={'1tv3'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('2u1','2u2','2u3','2u4','2e1','2e2','2e3','2e4',
tests2(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(2,i)= {'OK'};
    else
        resultado1(2,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(2,i)= {'F.A.'};
    else
        resultado1(2,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%--------------------------------------------------------------------------

tests3(1)={'3tv1'};
tests3(2)={'3tv2'};
tests3(3)={'3tv3'};
tests3(4)={'3tv4'};
tests3(5)={'3tv5'};
tests3(6)={'3tv6'};
tests3(7)={'3tv7'};
tests3(8)={'3tv8'};
tests3(9)={'3tv9'};
tests3(10)={'3tv10'};
tests3(11)={'1tv3'};
tests3(12)={'2tv3'};
tests3(13)={'4tv3'};
tests3(14)={'5tv3'};
tests3(15)={'6tv3'};
tests3(16)={'7tv3'};
tests3(17)={'8tv3'};
tests3(18)={'9tv3'};
tests3(19)={'10tv3'};
tests3(20)={'2tv2'};
```

```
for i=1:20
[Disttotal,
umbral]=entrenamiento('3u1','3u2','3u3','3u4','3e1','3e2','3e3','3e4',
tests3(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(3,i)= {'OK'};
    else
        resultado1(3,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(3,i)= {'F.A.'};
    else
        resultado1(3,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%--------------------------------------------------------------------
tests4(1)={'4tv1'};
tests4(2)={'4tv2'};
tests4(3)={'4tv3'};
tests4(4)={'4tv4'};
tests4(5)={'4tv5'};
tests4(6)={'4tv6'};
tests4(7)={'4tv7'};
tests4(8)={'4tv8'};
tests4(9)={'4tv9'};
tests4(10)={'4tv10'};
tests4(11)={'1tv4'};
tests4(12)={'2tv4'};
tests4(13)={'3tv4'};
tests4(14)={'5tv4'};
tests4(15)={'6tv4'};
tests4(16)={'7tv4'};
tests4(17)={'8tv4'};
tests4(18)={'9tv4'};
tests4(19)={'10tv4'};
tests4(20)={'3tv3'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('4u1','4u2','4u3','4u4','4e1','4e2','4e3','4e4',
tests4(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(4,i)= {'OK'};
    else
        resultado1(4,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(4,i)= {'F.A.'};
    else
        resultado1(4,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%--------------------------------------------------------------------
```

```
tests5(1)={'5tv1'};
tests5(2)={'5tv2'};
tests5(3)={'5tv3'};
tests5(4)={'5tv4'};
tests5(5)={'5tv5'};
tests5(6)={'5tv6'};
tests5(7)={'5tv7'};
tests5(8)={'5tv8'};
tests5(9)={'5tv9'};
tests5(10)={'5tv10'};
tests5(11)={'1tv5'};
tests5(12)={'2tv5'};
tests5(13)={'3tv5'};
tests5(14)={'4tv5'};
tests5(15)={'6tv5'};
tests5(16)={'7tv5'};
tests5(17)={'8tv5'};
tests5(18)={'9tv5'};
tests5(19)={'10tv5'};
tests5(20)={'4tv4'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('5u1','5u2','5u3','5u4','5e1','5e2','5e3','5e4',
tests5(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(5,i)= {'OK'};
    else
        resultado1(5,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(5,i)= {'F.A.'};
    else
        resultado1(5,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%--------------------------------------------------------------------

tests6(1)={'6tv1'};
tests6(2)={'6tv2'};
tests6(3)={'6tv3'};
tests6(4)={'6tv4'};
tests6(5)={'6tv5'};
tests6(6)={'6tv6'};
tests6(7)={'6tv7'};
tests6(8)={'6tv8'};
tests6(9)={'6tv9'};
tests6(10)={'6tv10'};
tests6(11)={'1tv6'};
tests6(12)={'2tv6'};
tests6(13)={'3tv6'};
tests6(14)={'4tv6'};
tests6(15)={'5tv6'};
tests6(16)={'7tv6'};
tests6(17)={'8tv6'};
tests6(18)={'9tv6'};
tests6(19)={'10tv6'};
tests6(20)={'4tv4'};
```

```
for i=1:20
[Disttotal,
umbral]=entrenamiento('6u1','6u2','6u3','6u4','6e1','6e2','6e3','6e4',
tests6(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(6,i)= {'OK'};
    else
        resultado1(6,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(6,i)= {'F.A.'};
    else
        resultado1(6,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-----------------------------------------------------------------------

tests7(1)={'7tv1'};
tests7(2)={'7tv2'};
tests7(3)={'7tv3'};
tests7(4)={'7tv4'};
tests7(5)={'7tv5'};
tests7(6)={'7tv6'};
tests7(7)={'7tv7'};
tests7(8)={'7tv8'};
tests7(9)={'7tv9'};
tests7(10)={'7tv10'};
tests7(11)={'1tv7'};
tests7(12)={'2tv7'};
tests7(13)={'3tv7'};
tests7(14)={'4tv7'};
tests7(15)={'5tv7'};
tests7(16)={'6tv7'};
tests7(17)={'8tv7'};
tests7(18)={'9tv7'};
tests7(19)={'10tv7'};
tests7(20)={'6tv6'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('7u1','7u2','7u3','7u4','7e1','7e2','7e3','7e4',
tests7(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(7,i)= {'OK'};
    else
        resultado1(7,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(7,i)= {'F.A.'};
    else
        resultado1(7,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-----------------------------------------------------------------------
```

```
tests8(1)={'8tv1'};
tests8(2)={'8tv2'};
tests8(3)={'8tv3'};
tests8(4)={'8tv4'};
tests8(5)={'8tv5'};
tests8(6)={'8tv6'};
tests8(7)={'8tv7'};
tests8(8)={'8tv8'};
tests8(9)={'8tv9'};
tests8(10)={'8tv10'};
tests8(11)={'1tv8'};
tests8(12)={'2tv8'};
tests8(13)={'3tv8'};
tests8(14)={'4tv8'};
tests8(15)={'5tv8'};
tests8(16)={'6tv8'};
tests8(17)={'7tv8'};
tests8(18)={'9tv8'};
tests8(19)={'10tv8'};
tests8(20)={'7tv7'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('8u1','8u2','8u3','8u4','8e1','8e2','8e3','8e4',
tests8(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(8,i)= {'OK'};
    else
        resultado1(8,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(8,i)= {'F.A.'};
    else
        resultado1(8,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------------
tests9(1)={'9tv1'};
tests9(2)={'9tv2'};
tests9(3)={'9tv3'};
tests9(4)={'9tv4'};
tests9(5)={'9tv5'};
tests9(6)={'9tv6'};
tests9(7)={'9tv7'};
tests9(8)={'9tv8'};
tests9(9)={'9tv9'};
tests9(10)={'9tv10'};
tests9(11)={'1tv9'};
tests9(12)={'2tv9'};
tests9(13)={'3tv9'};
tests9(14)={'4tv9'};
tests9(15)={'5tv9'};
tests9(16)={'6tv9'};
tests9(17)={'7tv9'};
tests9(18)={'8tv9'};
tests9(19)={'10tv9'};
tests9(20)={'8tv8'};
```

```
for i=1:20
[Disttotal,
umbral]=entrenamiento('9u1','9u2','9u3','9u4','9e1','9e2','9e3','9e4',
tests9(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(9,i)= {'OK'};
    else
        resultado1(9,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(9,i)= {'F.A.'};
    else
        resultado1(9,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%--------------------------------------------------------------------

tests10(1)={'10tv1'};
tests10(2)={'10tv2'};
tests10(3)={'10tv3'};
tests10(4)={'10tv4'};
tests10(5)={'10tv5'};
tests10(6)={'10tv6'};
tests10(7)={'10tv7'};
tests10(8)={'10tv8'};
tests10(9)={'10tv9'};
tests10(10)={'10tv10'};
tests10(11)={'1tv10'};
tests10(12)={'2tv10'};
tests10(13)={'3tv10'};
tests10(14)={'4tv10'};
tests10(15)={'5tv10'};
tests10(16)={'6tv10'};
tests10(17)={'7tv10'};
tests10(18)={'8tv10'};
tests10(19)={'9tv10'};
tests10(20)={'9tv9'};
for i=1:20
[Disttotal,
umbral]=entrenamiento('10u1','10u2','10u3','10u4','10e1','10e2','10e3'
,'10e4',tests10(i),M,M2,R,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(10,i)= {'OK'};
    else
        resultado1(10,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(10,i)= {'F.A.'};
    else
        resultado1(10,i)={'OK'};
    end
end
end
clear Disttotal umbral;
aciertos=0;
FA=0;
```

```
FR=0;
for i=1:20
    for j=1:10
       aciertos=aciertos+strcmp(resultado1{j,i},'OK');
       FR=FR+strcmp(resultado1{j,i},'F.R.');
       FA=FA+strcmp(resultado1{j,i},'F.A.');

    end
end
Tasa_FA= FA/2;
Tasa_FR= FR/2;
TASA_OK= aciertos/2;
```

Testeo_mfcc_codigoimpostor.m

Es igual que Testeo_mfcc_bueno.m solo que cambian los archivos a testear, y son los que siguen:

```
tests1(1)={'1tf1'};
tests1(2)={'1tf2'};
tests1(3)={'1tf3'};
tests1(4)={'1tf4'};
tests1(5)={'1tf5'};
tests1(6)={'1tf6'};
tests1(7)={'1tf7'};
tests1(8)={'1tf8'};
tests1(9)={'1tf9'};
tests1(10)={'1tf10'};
tests1(11)={'2tf1'};
tests1(12)={'3tf1'};
tests1(13)={'4tf1'};
tests1(14)={'5tf1'};
tests1(15)={'6tf1'};
tests1(16)={'7tf1'};
tests1(17)={'8tf1'};
tests1(18)={'9tf1'};
tests1(19)={'10tf1'};
tests1(20)={'2tf2'};
```

```
tests3(1)={'3tf1'};
tests3(2)={'3tf2'};
tests3(3)={'3tf3'};
tests3(4)={'3tf4'};
tests3(5)={'3tf5'};
tests3(6)={'3tf6'};
tests3(7)={'3tf7'};
tests3(8)={'3tf8'};
tests3(9)={'3tf9'};
tests3(10)={'3tf10'};
tests3(11)={'1tf3'};
tests3(12)={'2tf3'};
tests3(13)={'4tf3'};
tests3(14)={'5tf3'};
tests3(15)={'6tf3'};
tests3(16)={'7tf3'};
tests3(17)={'8tf3'};
tests3(18)={'9tf3'};
tests3(19)={'10tf3'};
tests3(20)={'2tf2'};
```

```
tests5(1)={'5tf1'};
tests5(2)={'5tf2'};
tests5(3)={'5tf3'};
tests5(4)={'5tf4'};
tests5(5)={'5tf5'};
tests5(6)={'5tf6'};
tests5(7)={'5tf7'};
tests5(8)={'5tf8'};
tests5(9)={'5tf9'};
tests5(10)={'5tf10'};
tests5(11)={'1tf5'};
tests5(12)={'2tf5'};
tests5(13)={'3tf5'};
tests5(14)={'4tf5'};
tests5(15)={'6tf5'};
tests5(16)={'7tf5'};
tests5(17)={'8tf5'};
tests5(18)={'9tf5'};
tests5(19)={'10tf5'};
tests5(20)={'4tf4'};
```

```
tests2(1)={'2tf1'};
tests2(2)={'2tf2'};
tests2(3)={'2tf3'};
tests2(4)={'2tf4'};
tests2(5)={'2tf5'};
tests2(6)={'2tf6'};
tests2(7)={'2tf7'};
tests2(8)={'2tf8'};
tests2(9)={'2tf9'};
tests2(10)={'2tf10'};
tests2(11)={'1tf2'};
tests2(12)={'3tf2'};
tests2(13)={'4tf2'};
tests2(14)={'5tf2'};
tests2(15)={'6tf2'};
tests2(16)={'7tf2'};
tests2(17)={'8tf2'};
tests2(18)={'9tf2'};
tests2(19)={'10tf2'};
tests2(20)={'1tf3'};
```

```
tests4(1)={'4tf1'};
tests4(2)={'4tf2'};
tests4(3)={'4tf3'};
tests4(4)={'4tf4'};
tests4(5)={'4tf5'};
tests4(6)={'4tf6'};
tests4(7)={'4tf7'};
tests4(8)={'4tf8'};
tests4(9)={'4tf9'};
tests4(10)={'4tf10'};
tests4(11)={'1tf4'};
tests4(12)={'2tf4'};
tests4(13)={'3tf4'};
tests4(14)={'5tf4'};
tests4(15)={'6tf4'};
tests4(16)={'7tf4'};
tests4(17)={'8tf4'};
tests4(18)={'9tf4'};
tests4(19)={'10tf4'};
tests4(20)={'3tf3'};
```

```
tests6(1)={'6tf1'};
tests6(2)={'6tf2'};
tests6(3)={'6tf3'};
tests6(4)={'6tf4'};
tests6(5)={'6tf5'};
tests6(6)={'6tf6'};
tests6(7)={'6tf7'};
tests6(8)={'6tf8'};
tests6(9)={'6tf9'};
tests6(10)={'6tf10'};
tests6(11)={'1tf6'};
tests6(12)={'2tf6'};
tests6(13)={'3tf6'};
tests6(14)={'4tf6'};
tests6(15)={'5tf6'};
tests6(16)={'7tf6'};
tests6(17)={'8tf6'};
tests6(18)={'9tf6'};
tests6(19)={'10tf6'};
tests6(20)={'4tf4'};
```

```
tests7(1)={'7tf1'};
tests7(2)={'7tf2'};
tests7(3)={'7tf3'};
tests7(4)={'7tf4'};
tests7(5)={'7tf5'};
tests7(6)={'7tf6'};
tests7(7)={'7tf7'};
tests7(8)={'7tf8'};
tests7(9)={'7tf9'};
tests7(10)={'7tf10'};
tests7(11)={'1tf7'};
tests7(12)={'2tf7'};
tests7(13)={'3tf7'};
tests7(14)={'4tf7'};
tests7(15)={'5tf7'};
tests7(16)={'6tf7'};
tests7(17)={'8tf7'};
tests7(18)={'9tf7'};
tests7(19)={'10tf7'};
tests7(20)={'6tf6'};
```

```
tests8(1)={'8tf1'};
tests8(2)={'8tf2'};
tests8(3)={'8tf3'};
tests8(4)={'8tf4'};
tests8(5)={'8tf5'};
tests8(6)={'8tf6'};
tests8(7)={'8tf7'};
tests8(8)={'8tf8'};
tests8(9)={'8tf9'};
tests8(10)={'8tf10'};
tests8(11)={'1tf8'};
tests8(12)={'2tf8'};
tests8(13)={'3tf8'};
tests8(14)={'4tf8'};
tests8(15)={'5tf8'};
tests8(16)={'6tf8'};
tests8(17)={'7tf8'};
tests8(18)={'9tf8'};
tests8(19)={'10tf8'};
tests8(20)={'7tf7'};
```

```
tests9(1)={'9tf1'};
tests9(2)={'9tf2'};
tests9(3)={'9tf3'};
tests9(4)={'9tf4'};
tests9(5)={'9tf5'};
tests9(6)={'9tf6'};
tests9(7)={'9tf7'};
tests9(8)={'9tf8'};
tests9(9)={'9tf9'};
tests9(10)={'9tf10'};
tests9(11)={'1tf9'};
tests9(12)={'2tf9'};
tests9(13)={'3tf9'};
tests9(14)={'4tf9'};
tests9(15)={'5tf9'};
tests9(16)={'6tf9'};
tests9(17)={'7tf9'};
tests9(18)={'8tf9'};
tests9(19)={'10tf9'};
tests9(20)={'8tf8'};
```

```
tests10(1)={'10tf1'};
tests10(2)={'10tf2'};
tests10(3)={'10tf3'};
tests10(4)={'10tf4'};
tests10(5)={'10tf5'};
tests10(6)={'10tf6'};
tests10(7)={'10tf7'};
tests10(8)={'10tf8'};
tests10(9)={'10tf9'};
tests10(10)={'10tf10'};
tests10(11)={'1tf10'};
tests10(12)={'2tf10'};
tests10(13)={'3tf10'};
tests10(14)={'4tf10'};
tests10(15)={'5tf10'};
tests10(16)={'6tf10'};
tests10(17)={'7tf10'};
tests10(18)={'8tf10'};
tests10(19)={'9tf10'};
tests10(20)={'9tf9'};
```

entrenamiento.m

```
function [Disttotal, umbral]=entrenamiento(umbral1, umbral2, umbral3,
umbral4, entrenamiento1, entrenamiento2, entrenamiento3,
entrenamiento4, testi,M,M2,R,deteccion_tsonoras,overlap,ventana)

[coefa]=extraccion_MFCC(entrenamiento1,M,M2,R,deteccion_tsonoras,overl
ap,ventana);
coefa1=coefa;
[coefa]=extraccion_MFCC(entrenamiento2,M,M2,R,deteccion_tsonoras,overl
ap,ventana);
coefa2=coefa;
[coefa]=extraccion_MFCC(entrenamiento3,M,M2,R,deteccion_tsonoras,overl
ap,ventana);
coefa3=coefa;
[coefa]=extraccion_MFCC(entrenamiento4,M,M2,R,deteccion_tsonoras,overl
ap,ventana);
coefa4=coefa;
[coefa]=extraccion_MFCC(umbral1,M,M2,R,deteccion_tsonoras,overlap,vent
ana);
coefa5=coefa;
[coefa]=extraccion_MFCC(umbral2,M,M2,R,deteccion_tsonoras,overlap,vent
ana);
coefa6=coefa;
[coefa]=extraccion_MFCC(umbral3,M,M2,R,deteccion_tsonoras,overlap,vent
ana);
coefa7=coefa;
[coefa]=extraccion_MFCC(umbral4,M,M2,R,deteccion_tsonoras,overlap,vent
ana);
coefa8=coefa;

%-------------------------------------------------------------------------
[Distancias,Distancias_primeras,umbral,modeloentrenado]=dtw2(coefa1,co
efa2,coefa3,coefa4,coefa5, coefa6, coefa7, coefa8, M2);
%[Distancias,umbral,modeloentrenado,vector]=dtw2_b(coefa1,coefa2,coefa
3,coefa4,M2);

[coefa]=extraccion_MFCC(testi{1},M,M2,R,deteccion_tsonoras,overlap,ven
tana);
coefatest=coefa;
[output,Disttotal,modeloentrenado,coefatest]=dtw4(coefatest,
modeloentrenado,M2,umbral);
```

extraccion_MFCC.m

```
function
[coefa]=extraccion_MFCC(archivo,M,M2,R,deteccion_tsonoras,overlap,vent
ana)
[x,fs,n]=wavread(archivo);
Ltrama= floor(fs*ventana);
nfft = 2^(nextpow2(Ltrama));
K = nfft/2+1;
h2m = @( hz )( 1127*log(1+hz/700));
m2h = @( mel )( 700 * exp(mel/1127)-700 );
      if deteccion_tsonoras==1
      limpia=x';
```

```
      elseif deteccion_tsonoras==2
      [limpia] = sonorapitch( x,fs,ventana );
      else
      [limpia] = deteccion_tsonora(x,fs,ventana);
      end
[ccjuntos] = MFCC(fs, limpia, ventana, overlap ,M,M2,K,R,h2m,m2h);
[ccjuntnorm] = CMVN2(ccjuntos,M2);
[delta,ddelta] = DIFMFCC( ccjuntnorm,M2);
[ data ] = vectorcarac( ccjuntnorm,ddelta,delta,M2);
coefa = data';
```

sonorapitch.m

```
function [ limpia ] = sonorapitch( x,fs,ventana )
Lsegmento=length(x);
Ltrama=(fs*ventana);
Ntramas=floor(Lsegmento/Ltrama);
sonoridad=zeros(1,Ntramas);
sordo=zeros(1,Ltrama);
ini=0;
limpia=[];
senergia=[];
energia=0;
    for i=1:Ntramas
    trama=x(ini+1:ini+Ltrama);
    media=mean(trama);
    mediatotal=trama-media;
    for j=1:Ltrama
    energia=energia + (mediatotal(j)*mediatotal(j));
    end
    energia=energia/Ltrama;
    senergia(i)= energia;
    B = fir1(25,(900/(fs/2)));
    A=(1:Ltrama);
    A(1,:)=1;
    mediatotal=mediatotal';
    filtro=filter(B,A,mediatotal);
    predictorlineal=lpc(filtro,4);
    residuo=filter(predictorlineal,A,filtro);
    autocorrelacion=xcorr(residuo);
    [amplitud1,posicion1] = max(autocorrelacion);
    ininicial=posicion1 + floor(0.006*fs);
    infinal=posicion1 + floor(0.025*fs);
    segundomaximo=autocorrelacion(ininicial:infinal);
    [amplitud2,posicion2] = max(segundomaximo);
    if amplitud2> 0.20*amplitud1
        pitch=posicion2 + floor(0.006*fs);
        pitchHz(i)=1/(pitch/fs);
        sonoridad(i)= 1;
        limpia(ini+1:ini+Ltrama)=mediatotal;
    else sonoridad(i)=0;
        pitchHz(i)=0;
        limpia(ini+1:ini+Ltrama)=sordo;
    end
    ini=ini+Ltrama;
    end
    maxene=max(senergia);
    ini=0;
    for k=1:Ntramas
```

```
        if senergia(k)> maxene/20
            limpia(ini+1:ini+Ltrama)=limpia(ini+1:ini+Ltrama);
        else
            limpia(ini+1:ini+Ltrama)=sordo;
        end
         ini=ini+Ltrama;
    end
    B = fir1(26, 75/(fs/2),'high');
    limpia=filter(B,1,limpia);
end
```

deteccionsonora.m

```
function [limpia] = deteccionsonora(x,fs,ventana)
%CÁLCULO DE LA ENERGÍA
Lsegmento=length(x);
Ltrama=(fs*ventana);
Ntramas=floor(Lsegmento/Ltrama);
senergia=[];
filtrada=[];
maximoener=0;
x=x/max(abs(x)); %Normalizo
for i=0:(Ntramas-1)
    energia=0;
    trama=x(1+(i*Ltrama):(i+1)*Ltrama
    media=mean(trama);
    tramamedia=trama-media;
    B = fir1(26, 75/(fs/2),'high');
    pasoalto=filter(B,1,tramamedia);
    ZCR(i+1)=mean(abs(diff(sign(pasoalto))));
    for j=1:Ltrama
    energia=energia + (pasoalto(j)*pasoalto(j));
    end
    senergia(i+1)= energia;
    if energia > maximoener
    maximoener=energia;
    end
    filtrada(1+(i*Ltrama):(i+1)*Ltrama)=pasoalto;
end
umbrale=maximoener/12;
umbralzcr=0.7;

for i=0:(Ntramas-1)
    if senergia(i+1) > umbrale
        if ZCR(i+1) < umbralzcr
        sonora(1+(i*Ltrama):(i+1)*Ltrama)=1;
        else
        sonora(1+(i*Ltrama):(i+1)*Ltrama)=0;
        end
    else
        sonora(1+(i*Ltrama):(i+1)*Ltrama)=0;
    end
end
[ml nl]=size(sonora);
[mx nx]=size(filtrada);
diferencia=abs(nx-nl);
sonora=horzcat(sonora,zeros(1,diferencia));
limpia=filtrada.* sonora;
```

```
end
```

MFCC.m

```
function [ccjuntos] = MFCC(fs,limpia,ventana,overlap,M,M2,K,R,h2m,m2h)
Lsegmento=length(limpia);
Ltrama= round(fs*ventana);
Ntramas=floor(Lsegmento/Ltrama);
coeficientesf = [1 -0.97];
enfatizada = filter(coeficientesf ,1 ,limpia);
h=hamming(Ltrama);
[H,f,C] = trifbank(M, K, R, fs, h2m, m2h );
contador=0;
ccjuntos=[];
for i=0:(Ntramas /overlap)-(1/overlap + 2)
    trama =
enfatizada(((Ltrama*i*overlap)+1):(Ltrama*i*overlap)+Ltrama);
    enventanada=h.*trama';
    magnitud=abs(fft(enventanada,Ltrama,1));
    FBE = H * magnitud(1:K,:);
    FBE2=FBE(1:M2,:);
    logf=log10(FBE2);
    coeficientes = dct( logf );
    coeficientes=coeficientes';
    cc(i+1,:)=coeficientes;
    a=isnan(coeficientes);
    b=isinf(coeficientes);
    c=a+b;
    if sum(c)==M2
        cc(i+1,:)=0;
    else
      for u=1:M2
            if coeficientes(u)==0
            cc(i+1,:)=0;
            end
      end
    end
end
for j=1:((Ntramas /overlap)- (1/overlap + 2))
    if sum(cc(j,:)) == 0
        ccjuntos=ccjuntos;
    else
        contador=contador+1;
        ccjuntos(contador,:)= cc(j,:);
    end
end
end
```

trifbank.m*

```
function [ H, f, c ] = trifbank( M, K, R, fs, h2w, w2h )

%   Inputs------------------------------------------------------------
%   M is the number of filters, i.e., number of rows of H
%   K is the length of frequency response of each filter
%   R is a two element vector that specifies frequency limits (Hz),
%   FS is the sampling frequency (Hz)
%   H2W is a Hertz scale to warped scale function handle
%   W2H is a wared scale to Hertz scale function handle
%   Outputs-----------------------------------------------------------
%   H is a M by K triangular filterbank matrix (one filter per row)
%   F is a frequency vector (Hz) of 1xK dimension
%   C is a vector of filter cutoff frequencies (Hz)

**************************************************************************

    if( nargin~= 6 ), help trifbank; return; end;
    f_min = 0;
    f_low = R(1);
    f_high = R(2);
    f_max = 4000
    f = linspace( f_min, f_max, K );
    fw = h2w( f );
    c = w2h( h2w(f_low)+[0:M+1]*((h2w(f_high)-h2w(f_low))/(M+1)) );
    cw = h2w( c );
    H = zeros( M, K );
    for m = 1:M

        % implements Eq. (6.140) on page 314 of [1]
        % k = f>=c(m)&f<=c(m+1); % up-slope
        % H(m,k) = 2*(f(k)-c(m)) / ((c(m+2)-c(m))*(c(m+1)-c(m)));
        % k = f>=c(m+1)&f<=c(m+2); % down-slope
        % H(m,k) = 2*(c(m+2)-f(k)) / ((c(m+2)-c(m))*(c(m+2)-c(m+1)));

        % implements Eq. (6.141) on page 315 of [1]
        k = f>=c(m)&f<=c(m+1); % up-slope
        H(m,k) = (f(k)-c(m))/(c(m+1)-c(m));
        k = f>=c(m+1)&f<=c(m+2); % down-slope
        H(m,k) = (c(m+2)-f(k))/(c(m+2)-c(m+1));
   end
   % H = H./repmat(max(H,[],2),1,K);  % normalize to unit height
(inherently done)
   H = H./repmat(trapz(f,H,2),1,K); % normalize to unit area
```

** Esta función ha sido desarrollada por Kamil Wojcicki en Junio de 2011*

CMVN2.m

```
function [ ccjuntnorm ] = CMVN2( ccjuntos,M2)
[a h] = size(ccjuntos);
Ncoef = a;
varianza=[];

for i=1:(M2)
    var=0;
    media(i) = sum(ccjuntos(:,i))/Ncoef;
    for t=1:a
        var= var+(ccjuntos(t,i)-media(i))^2;
    end
    varianza(i)= var/(Ncoef-1);
end
for k=1:(M2)
    for j=1:a
        ccjuntnorm(j,k)=((ccjuntos(j,k)- media(k))/varianza(k));
    end
end
end
```

DIFMFCC.m

```
function [delta,ddelta,l] = DIFMFCC( ccjuntnorm , M2)
[a h] = size(ccjuntnorm);
%[aj hj] = size(ccjuntnorm);
l=2;

for i=1:M2
    for j=1+l:a-l
    del=0;
        for k=-l:l
            if k~=0
                del = del +( k * ccjuntnorm(j-k,i)/abs(k));

            end
        end
        delta(j-l,i)=del;
    end
end
```

vectorcarac.m

```
function [ data ] = vectorcarac( ccjuntnorm,ddelta,delta,M2)
[a h]=size(ccjuntnorm);
[da dh]=size(delta);
[dda ddh]=size(ddelta);
      for dp=1:(a-da)
      dpadding(dp,:)=zeros(1,M2);
      end
      for ddp=1:(a-dda)
      ddpadding(ddp,:)=zeros(1,M2);
      end
```

```
delta=[delta;dpadding];
ddelta=[ddelta;ddpadding];
data=[ccjuntnorm delta ddelta];
```

dtw3.m*

```
function [Dist,DistNorm,D,k,w,rw,tw]=dtw3(r,t,pflag)
% [Dist,D,k,w,rw,tw]=dtw(r,t,pflag)
% Dynamic Time Warping Algorithm
% Dist is unnormalized distance between t and r
% D is the accumulated distance matrix
% k is the normalizing factor
% w is the optimal path
% t is the vector you are testing against
% r is the vector you are testing
% rw is the warped r vector
% tw is the warped t vector
% pflag  plot flag: 1 (yes), 0(no)
% Version comments:
% rw, tw and pflag added by Pau Mic

[row,M]=size(r); if (row > M) M=row; r=r'; end;
[row,N]=size(t); if (row > N) N=row; t=t'; end;
d=(repmat(r',1,N)-repmat(t,M,1)).^2;
D=zeros(size(d));
D(1,1)=d(1,1);
for m=2:M
    D(m,1)=d(m,1)+D(m-1,1);
end
for n=2:N
    D(1,n)=d(1,n)+D(1,n-1);
end
for m=2:M
    for n=2:N
        D(m,n)=d(m,n)+min(D(m-1,n),min(D(m-1,n-1),D(m,n-1)));
    end
end

Dist=D(M,N);
n=N;
m=M;
k=1;
w=[M N];
while ((n+m)~=2)
    if (n-1)==0
        m=m-1;
    elseif (m-1)==0
        n=n-1;
    else
      [values,number]=min([D(m-1,n),D(m,n-1),D(m-1,n-1)]);
      switch number
      case 1
        m=m-1;
      case 2
        n=n-1;
      case 3
        m=m-1;
        n=n-1;
      end
```

```
  end
    k=k+1;
    w=[m n; w];
end
DistNorm=Dist/k;

% warped waves
rw=r(w(:,1));
tw=t(w(:,2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
if pflag

    % --- Accumulated distance matrix and optimal path
    figure('Name','DTW - Accumulated distance matrix and optimal
path', 'NumberTitle','off');

    main1=subplot('position',[0.19 0.19 0.67 0.79]);
    image(D);
    cmap = contrast(D);
    colormap(cmap); % 'copper' 'bone', 'gray' imagesc(D);
    hold on;
    x=w(:,1); y=w(:,2);
    ind=find(x==1); x(ind)=1+0.2;
    ind=find(x==M); x(ind)=M-0.2;
    ind=find(y==1); y(ind)=1+0.2;
    ind=find(y==N); y(ind)=N-0.2;
    plot(y,x,'-w', 'LineWidth',1);
    hold off;
    axis([1 N 1 M]);
    set(main1, 'FontSize',7, 'XTickLabel','', 'YTickLabel','');

    colorb1=subplot('position',[0.88 0.19 0.05 0.79]);
    nticks=8;
    ticks=floor(1:(size(cmap,1)-1)/(nticks-1):size(cmap,1));
    mx=max(max(D));
    mn=min(min(D));
    ticklabels=floor(mn:(mx-mn)/(nticks-1):mx);
    colorbar(colorb1);
    set(colorb1, 'FontSize',7, 'YTick',ticks,
'YTickLabel',ticklabels);
    set(get(colorb1,'YLabel'), 'String','Distance', 'Rotation',-90,
'FontSize',7, 'VerticalAlignment','bottom');

    left1=subplot('position',[0.07 0.19 0.10 0.79]);
    plot(r,M:-1:1,'-b');
    set(left1, 'YTick',mod(M,10):10:M, 'YTickLabel',10*rem(M,10):-
10:0)
    axis([min(r) 1.1*max(r) 1 M]);
    set(left1, 'FontSize',7);
    set(get(left1,'YLabel'), 'String','Samples', 'FontSize',7,
'Rotation',-90, 'VerticalAlignment','cap');
    set(get(left1,'XLabel'), 'String','Amp', 'FontSize',6,
'VerticalAlignment','cap');

    bottom1=subplot('position',[0.19 0.07 0.67 0.10]);
    plot(t,'-r');
    axis([1 N min(t) 1.1*max(t)]);
    set(bottom1, 'FontSize',7, 'YAxisLocation','right');
```

```
    set(get(bottom1,'XLabel'), 'String','Samples', 'FontSize',7,
'VerticalAlignment','middle');
    set(get(bottom1,'YLabel'), 'String','Amp', 'Rotation',-90,
'FontSize',6, 'VerticalAlignment','bottom');

    % --- Warped signals
    figure('Name','DTW - warped signals', 'NumberTitle','off');

    subplot(1,2,1);
    set(gca, 'FontSize',7);
    hold on;
    plot(r,'-bx');
    plot(t,':r.');
    hold off;
    axis([1 max(M,N) min(min(r),min(t)) 1.1*max(max(r),max(t))]);
    grid;
    legend('signal 1','signal 2');
    title('Original signals');
    xlabel('Samples');
    ylabel('Amplitude');

    subplot(1,2,2);
    set(gca, 'FontSize',7);
    hold on;
    plot(rw,'-bx');
    plot(tw,':r.');
    hold off;
    axis([1 k min(min([rw; tw])) 1.1*max(max([rw; tw]))]);
    grid;
    legend('signal 1','signal 2');
    title('Warped signals');
    xlabel('Samples');
    ylabel('Amplitude');

end
```

**Esta función ha sido desarrollada por Pau Mic y otros.*

dtw2.m

```
function [Distancias,Distancias_primeras,
umbral,modeloentrenado]=dtw2(coefa1,coefa2,coefa3,coefa4,coefa5,
coefa6, coefa7, coefa8, M2)
Disttotal1=0;
Disttotal2=0;
Disttotal3=0;
Disttotal4=0;
Disttotal5=0;
Disttotal6=0;
Disttotal7=0;
Disttotal8=0;
Disttotal9=0;
Disttotal10=0;
    for i=1:(M2*3)
        t=coefa1(i,:);
        r=coefa2(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        DistNorm1(i)=DistNorm;
        medio1{i}=(rw+tw)/2;
        Disttotal1=Disttotal1+DistNorm;
    end
clear DistNorm D k w t r rw tw
    for i=1:(M2*3)
        t=coefa1(i,:);
        r=coefa3(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        DistNorm2(i)=DistNorm;
        medio2{i}=(rw+tw)/2;
        Disttotal2=Disttotal2+DistNorm;
    end
    clear DistNorm D k w t r rw tw
    for i=1:(M2*3)
        t=coefa1(i,:);
        r=coefa4(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        DistNorm3(i)=DistNorm;
        medio3{i}=(rw+tw)/2;
        Disttotal3=Disttotal3+DistNorm;
    end
    clear DistNorm D k w t r rw tw
    for i=1:(M2*3)
        t=coefa3(i,:);
        r=coefa4(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        DistNorm4(i)=DistNorm;
        medio4{i}=(rw+tw)/2;
        Disttotal4=Disttotal4+DistNorm;
    end
    for i=1:(M2*3)
        t=coefa2(i,:);
        r=coefa4(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        DistNorm5(i)=DistNorm;
        medio5{i}= (rw+tw)/2;
        Disttotal5=Disttotal5+DistNorm;
    end
    for i=1:(M2*3)
        t=coefa2(i,:);
        r=coefa3(i,:);
```

```
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        DistNorm6(i)=DistNorm;
        medio6{i}= (rw+tw)/2;
        Disttotal6=Disttotal6+DistNorm;
    end
    for i=1:(M2*3)
        vector=horzcat(DistNorm1(i), DistNorm2(i), DistNorm3(i),
DistNorm4(i), DistNorm5(i), DistNorm6(i));
        [minimo index]=min(vector);
        switch index
            case 1
                modeloentrenado{i}=medio1{i};
            case 2
                modeloentrenado{i}=medio2{i};
            case 3
                modeloentrenado{i}=medio3{i};
            case 4
                modeloentrenado{i}=medio4{i};
            case 5
                modeloentrenado{i}=medio5{i};
            case 6
                modeloentrenado{i}=medio6{i};
        end
    end

ENTRENO PARA UMBRAL-------------------------------------------------

   for i=1:(M2*3)
        t=modeloentrenado{i};
        r=coefa5(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        Disttotal7=Disttotal7+DistNorm;
    end
clear DistNorm D k w t r rw tw
    for i=1:(M2*3)
        t=modeloentrenado{i};
        r=coefa6(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        Disttotal8=Disttotal8+DistNorm;
    end
    clear DistNorm D k w t r rw tw
    for i=1:(M2*3)
        t=modeloentrenado{i};
        r=coefa7(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        Disttotal9=Disttotal9+DistNorm;
    end
     clear DistNorm D k w t r rw tw
    for i=1:(M2*3)
        t=modeloentrenado{i};
        r=coefa8(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        Disttotal10=Disttotal10+DistNorm;
    end
      Distancias_primeras=[
Disttotal1,Disttotal2,Disttotal3,Disttotal4, Disttotal5,Disttotal6];
Distancias=[ Disttotal7,Disttotal8,Disttotal9,Disttotal10];
umbral= mean(Distancias)+std(Distancias);
end
```

dtw4.m

```
function [output,Disttotal,coefatest]=dtw4(coefatest, modeloentrenado,
M2, umbral)

Disttotal=0;
 for i=1:(M2*3)
        t=modeloentrenado{i};
        r=coefatest(i,:);
        [Dist,DistNorm,D,k,w,rw,tw] = dtw3(t,r,0);
        Disttotal=Disttotal+DistNorm;
 end
if Disttotal > umbral
    output='Rechazado';
else
    output='Aceptado';
end
end
```

Anexo 2 – Código Matlab LPCC

Testeo_lpcc_bueno.m

```
%** PARÁMETROS EXTRACCIÓN LPCC*************************************
**********************************************************************
orden=10;               %Orden del filtro lpc
num=20;                 %Número de coeficientes
deteccion_tsonoras=3;   %modo de segmentado de tramas sonoras:
                        %1= sin segmentado
                        %2= segmentado por deteccion de energia y ZCR
                        %3= segmentado por detección de pitch mediante
                          SIFT
ventana=0.016;          %Tamaño de la ventana de análisis en s.
overlap=0.5;            %solape entre tramas,0 solape total; 1 sin
                         solaparse

%**********************************************************************
%**********************************************************************

tests1(1)={'1tv1'};
tests1(2)={'1tv2'};
tests1(3)={'1tv3'};
tests1(4)={'1tv4'};
tests1(5)={'1tv5'};
tests1(6)={'1tv6'};
tests1(7)={'1tv7'};
tests1(8)={'1tv8'};
tests1(9)={'1tv9'};
tests1(10)={'1tv10'};
tests1(11)={'2tv1'};
tests1(12)={'3tv1'};
tests1(13)={'4tv1'};
tests1(14)={'5tv1'};
tests1(15)={'6tv1'};
```

```
tests1(16)={'7tv1'};
tests1(17)={'8tv1'};
tests1(18)={'9tv1'};
tests1(19)={'10tv1'};
tests1(20)={'2tv2'};

for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('1u1','1u2','1u3','1u4','1e1','1e2','1e3','
1e4',tests1(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(1,i)= {'OK'};
    else
        resultado1(1,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(1,i)= {'F.A.'};
    else
        resultado1(1,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-----------------------------------------------------------------------

tests2(1)={'2tv1'};
tests2(2)={'2tv2'};
tests2(3)={'2tv3'};
tests2(4)={'2tv4'};
tests2(5)={'2tv5'};
tests2(6)={'2tv6'};
tests2(7)={'2tv7'};
tests2(8)={'2tv8'};
tests2(9)={'2tv9'};
tests2(10)={'2tv10'};
tests2(11)={'1tv2'};
tests2(12)={'3tv2'};
tests2(13)={'4tv2'};
tests2(14)={'5tv2'};
tests2(15)={'6tv2'};
tests2(16)={'7tv2'};
tests2(17)={'8tv2'};
tests2(18)={'9tv2'};
tests2(19)={'10tv2'};
tests2(20)={'1tv3'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('2u1','2u2','2u3','2u4','2e1','2e2','2e3','
2e4',tests2(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(2,i)= {'OK'};
    else
        resultado1(2,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(2,i)= {'F.A.'};
    else
        resultado1(2,i)={'OK'};
```

```
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------------
tests3(1)={'3tv1'};
tests3(2)={'3tv2'};
tests3(3)={'3tv3'};
tests3(4)={'3tv4'};
tests3(5)={'3tv5'};
tests3(6)={'3tv6'};
tests3(7)={'3tv7'};
tests3(8)={'3tv8'};
tests3(9)={'3tv9'};
tests3(10)={'3tv10'};
tests3(11)={'1tv3'};
tests3(12)={'2tv3'};
tests3(13)={'4tv3'};
tests3(14)={'5tv3'};
tests3(15)={'6tv3'};
tests3(16)={'7tv3'};
tests3(17)={'8tv3'};
tests3(18)={'9tv3'};
tests3(19)={'10tv3'};
tests3(20)={'2tv2'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('3u1','3u2','3u3','3u4','3e1','3e2','3e3','
3e4',tests3(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(3,i)= {'OK'};
    else
        resultado1(3,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(3,i)= {'F.A.'};
    else
        resultado1(3,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------------
tests4(1)={'4tv1'};
tests4(2)={'4tv2'};
tests4(3)={'4tv3'};
tests4(4)={'4tv4'};
tests4(5)={'4tv5'};
tests4(6)={'4tv6'};
tests4(7)={'4tv7'};
tests4(8)={'4tv8'};
tests4(9)={'4tv9'};
tests4(10)={'4tv10'};
tests4(11)={'1tv4'};
tests4(12)={'2tv4'};
tests4(13)={'3tv4'};
tests4(14)={'5tv4'};
tests4(15)={'6tv4'};
```

```
tests4(16)={'7tv4'};
tests4(17)={'8tv4'};
tests4(18)={'9tv4'};
tests4(19)={'10tv4'};
tests4(20)={'3tv3'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('4u1','4u2','4u3','4u4','4e1','4e2','4e3','
4e4',tests4(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(4,i)= {'OK'};
    else
        resultado1(4,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(4,i)= {'F.A.'};
    else
        resultado1(4,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------------
tests5(1)={'5tv1'};
tests5(2)={'5tv2'};
tests5(3)={'5tv3'};
tests5(4)={'5tv4'};
tests5(5)={'5tv5'};
tests5(6)={'5tv6'};
tests5(7)={'5tv7'};
tests5(8)={'5tv8'};
tests5(9)={'5tv9'};
tests5(10)={'5tv10'};
tests5(11)={'1tv5'};
tests5(12)={'2tv5'};
tests5(13)={'3tv5'};
tests5(14)={'4tv5'};
tests5(15)={'6tv5'};
tests5(16)={'7tv5'};
tests5(17)={'8tv5'};
tests5(18)={'9tv5'};
tests5(19)={'10tv5'};
tests5(20)={'4tv4'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('5u1','5u2','5u3','5u4','5e1','5e2','5e3','
5e4',tests5(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(5,i)= {'OK'};
    else
        resultado1(5,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(5,i)= {'F.A.'};
    else
        resultado1(5,i)={'OK'};
    end
end
```

```
end
clear Disttotal umbral;

%-------------------------------------------------------------------
tests6(1)={'6tv1'};
tests6(2)={'6tv2'};
tests6(3)={'6tv3'};
tests6(4)={'6tv4'};
tests6(5)={'6tv5'};
tests6(6)={'6tv6'};
tests6(7)={'6tv7'};
tests6(8)={'6tv8'};
tests6(9)={'6tv9'};
tests6(10)={'6tv10'};
tests6(11)={'1tv6'};
tests6(12)={'2tv6'};
tests6(13)={'3tv6'};
tests6(14)={'4tv6'};
tests6(15)={'5tv6'};
tests6(16)={'7tv6'};
tests6(17)={'8tv6'};
tests6(18)={'9tv6'};
tests6(19)={'10tv6'};
tests6(20)={'4tv4'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('6u1','6u2','6u3','6u4','6e1','6e2','6e3','
6e4',tests6(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(6,i)= {'OK'};
    else
        resultado1(6,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(6,i)= {'F.A.'};
    else
        resultado1(6,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------
tests7(1)={'7tv1'};
tests7(2)={'7tv2'};
tests7(3)={'7tv3'};
tests7(4)={'7tv4'};
tests7(5)={'7tv5'};
tests7(6)={'7tv6'};
tests7(7)={'7tv7'};
tests7(8)={'7tv8'};
tests7(9)={'7tv9'};
tests7(10)={'7tv10'};
tests7(11)={'1tv7'};
tests7(12)={'2tv7'};
tests7(13)={'3tv7'};
tests7(14)={'4tv7'};
tests7(15)={'5tv7'};
tests7(16)={'6tv7'};
tests7(17)={'8tv7'};
```

```
tests7(18)={'9tv7'};
tests7(19)={'10tv7'};
tests7(20)={'6tv6'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('7u1','7u2','7u3','7u4','7e1','7e2','7e3','
7e4',tests7(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(7,i)= {'OK'};
    else
        resultado1(7,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(7,i)= {'F.A.'};
    else
        resultado1(7,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------------
tests8(1)={'8tv1'};
tests8(2)={'8tv2'};
tests8(3)={'8tv3'};
tests8(4)={'8tv4'};
tests8(5)={'8tv5'};
tests8(6)={'8tv6'};
tests8(7)={'8tv7'};
tests8(8)={'8tv8'};
tests8(9)={'8tv9'};
tests8(10)={'8tv10'};
tests8(11)={'1tv8'};
tests8(12)={'2tv8'};
tests8(13)={'3tv8'};
tests8(14)={'4tv8'};
tests8(15)={'5tv8'};
tests8(16)={'6tv8'};
tests8(17)={'7tv8'};
tests8(18)={'9tv8'};
tests8(19)={'10tv8'};
tests8(20)={'7tv7'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('8u1','8u2','8u3','8u4','8e1','8e2','8e3','
8e4',tests8(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(8,i)= {'OK'};
    else
        resultado1(8,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(8,i)= {'F.A.'};
    else
        resultado1(8,i)={'OK'};
    end
end
end
clear Disttotal umbral;
```

```
%-------------------------------------------------------------------------
tests9(1)={'9tv1'};
tests9(2)={'9tv2'};
tests9(3)={'9tv3'};
tests9(4)={'9tv4'};
tests9(5)={'9tv5'};
tests9(6)={'9tv6'};
tests9(7)={'9tv7'};
tests9(8)={'9tv8'};
tests9(9)={'9tv9'};
tests9(10)={'9tv10'};
tests9(11)={'1tv9'};
tests9(12)={'2tv9'};
tests9(13)={'3tv9'};
tests9(14)={'4tv9'};
tests9(15)={'5tv9'};
tests9(16)={'6tv9'};
tests9(17)={'7tv9'};
tests9(18)={'8tv9'};
tests9(19)={'10tv9'};
tests9(20)={'8tv8'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('9u1','9u2','9u3','9u4','9e1','9e2','9e3','
9e4',tests9(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(9,i)= {'OK'};
    else
        resultado1(9,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(9,i)= {'F.A.'};
    else
        resultado1(9,i)={'OK'};
    end
end
end
clear Disttotal umbral;

%-------------------------------------------------------------------------
tests10(1)={'10tv1'};
tests10(2)={'10tv2'};
tests10(3)={'10tv3'};
tests10(4)={'10tv4'};
tests10(5)={'10tv5'};
tests10(6)={'10tv6'};
tests10(7)={'10tv7'};
tests10(8)={'10tv8'};
tests10(9)={'10tv9'};
tests10(10)={'10tv10'};
tests10(11)={'1tv10'};
tests10(12)={'2tv10'};
tests10(13)={'3tv10'};
tests10(14)={'4tv10'};
tests10(15)={'5tv10'};
tests10(16)={'6tv10'};
tests10(17)={'7tv10'};
tests10(18)={'8tv10'};
tests10(19)={'9tv10'};
```

```
tests10(20)={'9tv9'};
for i=1:20
[Disttotal,
umbral]=entrenamiento_lpcc('10u1','10u2','10u3','10u4','10e1','10e2','
10e3','10e4',tests10(i),orden,num,deteccion_tsonoras,overlap,ventana);
if i<=10
    if Disttotal <= umbral
        resultado1(10,i)= {'OK'};
    else
        resultado1(10,i)={'F.R.'};
    end
else if Disttotal <= umbral
        resultado1(10,i)= {'F.A.'};
    else
        resultado1(10,i)={'OK'};
    end
endend
clear Disttotal umbral;
aciertos=0;
FA=0;
FR=0;
for i=1:20
    for j=1:10
       aciertos=aciertos+strcmp(resultado1{j,i},'OK');
       FR=FR+strcmp(resultado1{j,i},'F.R.');
       FA=FA+strcmp(resultado1{j,i},'F.A.');
    end
end
Tasa_FA= FA/2;
Tasa_FR= FR/2;
TASA_OK= aciertos/2;
```

Testeo_lpcc_codigoimpostor.m

Es lo mismo que Testeo_lpcc_bueno solo que cambian los archivos del test por los del código erroneo. Son los siguientes ficheros:

```
tests1(1)={'1tf1'};
tests1(2)={'1tf2'};
tests1(3)={'1tf3'};
tests1(4)={'1tf4'};
tests1(5)={'1tf5'};
tests1(6)={'1tf6'};
tests1(7)={'1tf7'};
tests1(8)={'1tf8'};
tests1(9)={'1tf9'};
tests1(10)={'1tf10'};
tests1(11)={'2tf1'};
tests1(12)={'3tf1'};
tests1(13)={'4tf1'};
tests1(14)={'5tf1'};
tests1(15)={'6tf1'};
tests1(16)={'7tf1'};
tests1(17)={'8tf1'};
tests1(18)={'9tf1'};
tests1(19)={'10tf1'};
tests1(20)={'2tf2'};
```

```
tests2(1)={'2tf1'};
tests2(2)={'2tf2'};
tests2(3)={'2tf3'};
tests2(4)={'2tf4'};
tests2(5)={'2tf5'};
tests2(6)={'2tf6'};
tests2(7)={'2tf7'};
tests2(8)={'2tf8'};
tests2(9)={'2tf9'};
tests2(10)={'2tf10'};
tests2(11)={'1tf2'};
tests2(12)={'3tf2'};
tests2(13)={'4tf2'};
tests2(14)={'5tf2'};
tests2(15)={'6tf2'};
tests2(16)={'7tf2'};
tests2(17)={'8tf2'};
tests2(18)={'9tf2'};
tests2(19)={'10tf2'};
tests2(20)={'1tf3'};
```

```
tests3(1)={'3tf1'};
tests3(2)={'3tf2'};
tests3(3)={'3tf3'};
tests3(4)={'3tf4'};
tests3(5)={'3tf5'};
tests3(6)={'3tf6'};
tests3(7)={'3tf7'};
tests3(8)={'3tf8'};
tests3(9)={'3tf9'};
tests3(10)={'3tf10'};
tests3(11)={'1tf3'};
tests3(12)={'2tf3'};
tests3(13)={'4tf3'};
tests3(14)={'5tf3'};
tests3(15)={'6tf3'};
tests3(16)={'7tf3'};
tests3(17)={'8tf3'};
tests3(18)={'9tf3'};
tests3(19)={'10tf3'};
tests3(20)={'2tf2'};
```

```
tests4(1)={'4tf1'};
tests4(2)={'4tf2'};
tests4(3)={'4tf3'};
tests4(4)={'4tf4'};
tests4(5)={'4tf5'};
tests4(6)={'4tf6'};
tests4(7)={'4tf7'};
tests4(8)={'4tf8'};
tests4(9)={'4tf9'};
tests4(10)={'4tf10'};
tests4(11)={'1tf4'};
tests4(12)={'2tf4'};
tests4(13)={'3tf4'};
tests4(14)={'5tf4'};
tests4(15)={'6tf4'};
tests4(16)={'7tf4'};
tests4(17)={'8tf4'};
tests4(18)={'9tf4'};
tests4(19)={'10tf4'};
tests4(20)={'3tf3'};
```

```
tests5(1)={'5tf1'};
tests5(2)={'5tf2'};
tests5(3)={'5tf3'};
tests5(4)={'5tf4'};
tests5(5)={'5tf5'};
tests5(6)={'5tf6'};
tests5(7)={'5tf7'};
tests5(8)={'5tf8'};
tests5(9)={'5tf9'};
tests5(10)={'5tf10'};
tests5(11)={'1tf5'};
tests5(12)={'2tf5'};
tests5(13)={'3tf5'};
tests5(14)={'4tf5'};
tests5(15)={'6tf5'};
tests5(16)={'7tf5'};
tests5(17)={'8tf5'};
tests5(18)={'9tf5'};
tests5(19)={'10tf5'};
tests5(20)={'4tf4'};
```

```
tests6(1)={'6tf1'};
tests6(2)={'6tf2'};
tests6(3)={'6tf3'};
tests6(4)={'6tf4'};
tests6(5)={'6tf5'};
tests6(6)={'6tf6'};
tests6(7)={'6tf7'};
tests6(8)={'6tf8'};
tests6(9)={'6tf9'};
tests6(10)={'6tf10'};
tests6(11)={'1tf6'};
tests6(12)={'2tf6'};
tests6(13)={'3tf6'};
tests6(14)={'4tf6'};
tests6(15)={'5tf6'};
tests6(16)={'7tf6'};
tests6(17)={'8tf6'};
tests6(18)={'9tf6'};
tests6(19)={'10tf6'};
tests6(20)={'4tf4'};
```

```
tests7(1)={'7tf1'};
tests7(2)={'7tfv2'};
tests7(3)={'7tf3'};
tests7(4)={'7tf4'};
tests7(5)={'7tf5'};
tests7(6)={'7tf6'};
tests7(7)={'7tf7'};
tests7(8)={'7tf8'};
tests7(9)={'7tf9'};
tests7(10)={'7tf10'};
tests7(11)={'1tf7'};
tests7(12)={'2tf7'};
tests7(13)={'3tf7'};
tests7(14)={'4tf7'};
tests7(15)={'5tf7'};
tests7(16)={'6tf7'};
tests7(17)={'8tf7'};
tests7(18)={'9tf7'};
tests7(19)={'10tf7'};
tests7(20)={'6tf6'};
```

```
tests8(1)={'8tf1'};
tests8(2)={'8tf2'};
tests8(3)={'8tf3'};
tests8(4)={'8tf4'};
tests8(5)={'8tf5'};
tests8(6)={'8tf6'};
tests8(7)={'8tf7'};
tests8(8)={'8tf8'};
tests8(9)={'8tf9'};
tests8(10)={'8tf10'};
tests8(11)={'1tf8'};
tests8(12)={'2tf8'};
tests8(13)={'3tf8'};
tests8(14)={'4tf8'};
tests8(15)={'5tf8'};
tests8(16)={'6tf8'};
tests8(17)={'7tf8'};
tests8(18)={'9tf8'};
tests8(19)={'10tf8'};
tests8(20)={'7tf7'};
```

```
tests9(1)={'9tf1'};
tests9(2)={'9tf2'};
tests9(3)={'9tf3'};
tests9(4)={'9tf4'};
tests9(5)={'9tf5'};
tests9(6)={'9tf6'};
tests9(7)={'9tf7'};
tests9(8)={'9tf8'};
tests9(9)={'9tf9'};
tests9(10)={'9tf10'};
tests9(11)={'1tf9'};
tests9(12)={'2tf9'};
tests9(13)={'3tf9'};
tests9(14)={'4tf9'};
tests9(15)={'5tf9'};
tests9(16)={'6tf9'};
tests9(17)={'7tf9'};
tests9(18)={'8tf9'};
tests9(19)={'10tf9'};
tests9(20)={'8tf8'};
```

```
tests10(1)={'10tf1'};
tests10(2)={'10tf2'};
tests10(3)={'10tf3'};
tests10(4)={'10tf4'};
tests10(5)={'10tf5'};
tests10(6)={'10tf6'};
tests10(7)={'10tf7'};
tests10(8)={'10tf8'};
tests10(9)={'10tf9'};
tests10(10)={'10tf10'};
tests10(11)={'1tf10'};
tests10(12)={'2tf10'};
tests10(13)={'3tf10'};
tests10(14)={'4tf10'};
tests10(15)={'8tf10'};
tests10(16)={'9tf10'};
tests10(17)={'9tf9'};
tests10(18)={'8tf10'};
tests10(19)={'9tf10'};
tests10(20)={'9tf9'};
```

entrenamiento_lpcc.m

```
function [Disttotal, umbral]=entrenamiento_lpcc(umbral1, umbral2,
umbral3, umbral4, entrenamiento1, entrenamiento2, entrenamiento3,
entrenamiento4, test,orden,num,deteccion_tsonoras,overlap,ventana)
[coefa]=extraccion_LPCC(entrenamiento1,orden,num,deteccion_tsonoras,
overlap,ventana);
coefa1=coefa;
[coefa]=extraccion_LPCC(entrenamiento2,orden,num,deteccion_tsonoras,
overlap,ventana);
coefa2=coefa;
[coefa]=extraccion_LPCC(entrenamiento3,orden,num,deteccion_tsonoras,
overlap,ventana);
coefa3=coefa;
[coefa]=extraccion_LPCC(entrenamiento4,orden,num,deteccion_tsonoras,
overlap,ventana);
coefa4=coefa;
[coefa]=extraccion_LPCC(umbral1,orden,num,deteccion_tsonoras,overlap,
ventana);
coefa5=coefa;
[coefa]=extraccion_LPCC(umbral2,orden,num,deteccion_tsonoras,overlap,
ventana);
coefa6=coefa;
[coefa]=extraccion_LPCC(umbral3,orden,num,deteccion_tsonoras,overlap,
ventana);
coefa7=coefa;
[coefa]=extraccion_LPCC(umbral4,orden,num,deteccion_tsonoras,overlap,
ventana);
coefa8=coefa;

%-------------------------------------------------------------------
M2=num;
[Distancias,Distancias_primeras,umbral,modeloentrenado]=dtw2(coefa1,co
efa2,coefa3,coefa4,coefa5, coefa6, coefa7, coefa8,M2);

%[Distancias,umbral,modeloentrenado,vector]=dtw2_b(coefa1,coefa2,coefa
3,coefa4,M2);

%-------------------------------------------------------------------

[coefa]=extraccion_LPCC(test{1},orden,num,deteccion_tsonoras,overlap,v
entana);
coefatest=coefa;
[output,Disttotal]=dtw4(coefatest,modeloentrenado,M2,umbral);
```

extraccion_LPCC.m

```
function
[coefa]=extraccion_LPCC(archivo,orden,num,deteccion_tsonoras,overlap,
ventana)
M2=num;
[x,fs,n]=wavread(archivo);
if deteccion_tsonoras==1
    limpia=x';
elseif deteccion_tsonoras==2
    [limpia] = sonorapitch( x,fs,ventana );
else
    [limpia] = deteccionts onora(x,fs,ventana);
end
[lpccjuntos]=LPCC2(overlap,ventana,fs,limpia,orden,num);
[lpccjuntnorm] = CMVN2_lpcc(lpccjuntos,M2);
[delta,ddelta] = DIFMFCC( lpccjuntnorm,M2);
[ data ] = vectorcarac( lpccjuntnorm,ddelta,delta,M2);
coefa = data';
```

LPCC2.m

```
function [lpccjuntos] = LPCC2( overlap,ventana, fs,limpia,orden,num )
Lsegmento=length(limpia);
Ltrama= round(fs*ventana);
Ntramas=floor(Lsegmento/Ltrama);
coeficientesf = [1 -0.97];
enfatizada = filter(coeficientesf ,1 ,limpia);
h=hamming(Ltrama);
lpccjuntos=[];
contador=0;

for i=0: ((Ntramas /overlap)-(1/overlap + 2))
    trama =
enfatizada(((Ltrama*i*overlap)+1):(Ltrama*i*overlap)+Ltrama);
    enventanada=h.*trama';
    predictor=lpc(enventanada,orden)';
    hlpc2cc = dsp.LPCToCepstral('CepstrumLength',num+1);
    coef = step(hlpc2cc,predictor)';
    predictor2(i+1,:)= predictor';
    a=isnan(coef);
    if sum(a)==num
         lpcc(i+1,:)=zeros(1,num+1);
    else
         lpcc(i+1,:)=coef;
    end
end
for j=1:((Ntramas /overlap)- (1/overlap + 2))
    if sum(lpcc(j,:)) == 0
        lpccjuntos=lpccjuntos;
    else
        contador=contador+1;
        lpccjuntos(contador,:)= lpcc(j,:);
    end
end
end
```

CMVN_lpcc.m

```
function [ lpccjuntnorm ] = CMVN2_lpcc( lpccjuntos,M2)
num=M2;
[a h] = size(lpccjuntos);
Ncoef = a;
varianza=[];
for i=1:(num+1)
    var=0;
    media(i) = sum(lpccjuntos(:,i))/Ncoef;
    for t=1:a
        var= var+(lpccjuntos(t,i)-media(i))^2;
    end
    varianza(i)= var/(Ncoef-1);
end
for k=1:(num+1)
    for j=1:a
        if k==1
            lpccjuntnorm(j,k)=0;
        else
        lpccjuntnorm(j,k)=((lpccjuntos(j,k)- media(k))/varianza(k));
        end
    end
end
end
```

El resto de funciones se reutilizan del análisis MFCC.

Procesado digital de voz para el reconocimiento del hablante aplicado a dispositivos móviles

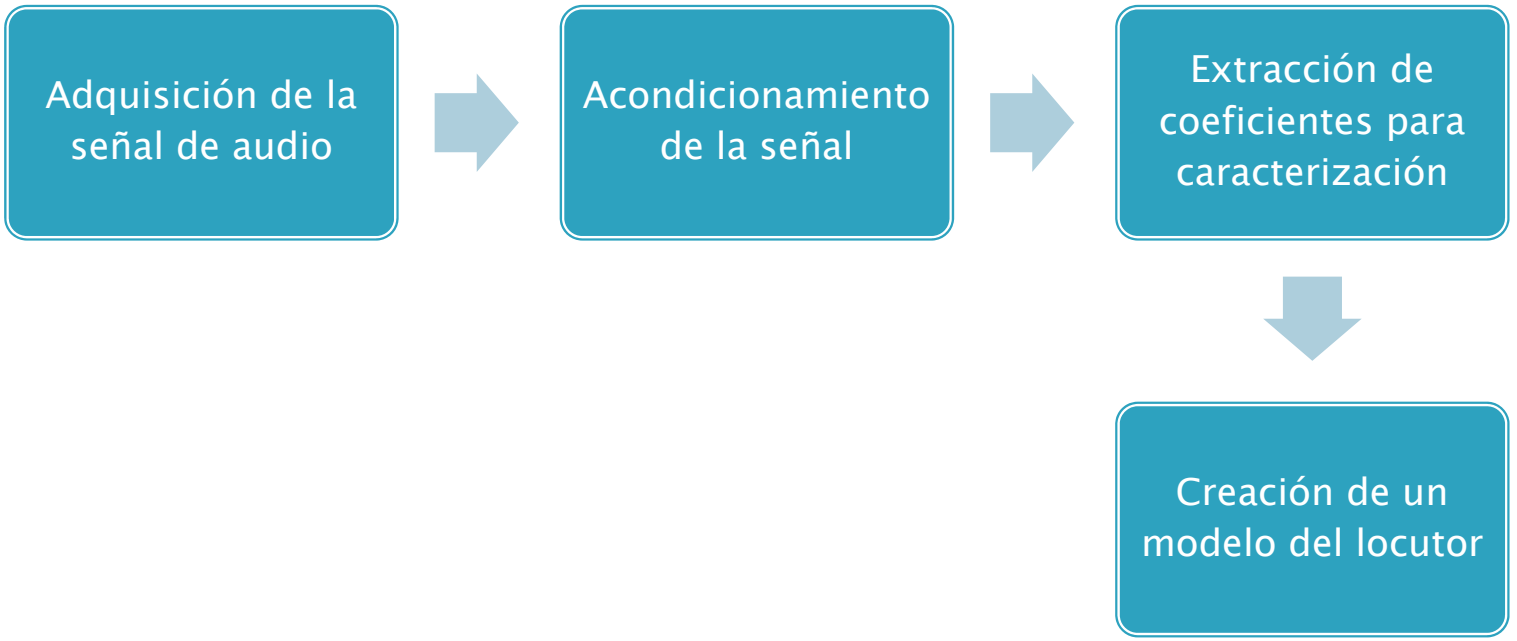
Daniel Moral Bárcena
Ingeniería Técnica de Telecomunicación

1 Introducción

- Resumen del PFC
  - Sistema de autenticación por medio de la señal de voz: **verificación automática del locutor.**
  - Desarrollado en lenguaje .m (Matlab).
  - Enfocado a dispositivos móviles.
  - Texto dependiente por su mejor rendimiento
  - Dos técnicas diferentes de caracterización de la voz

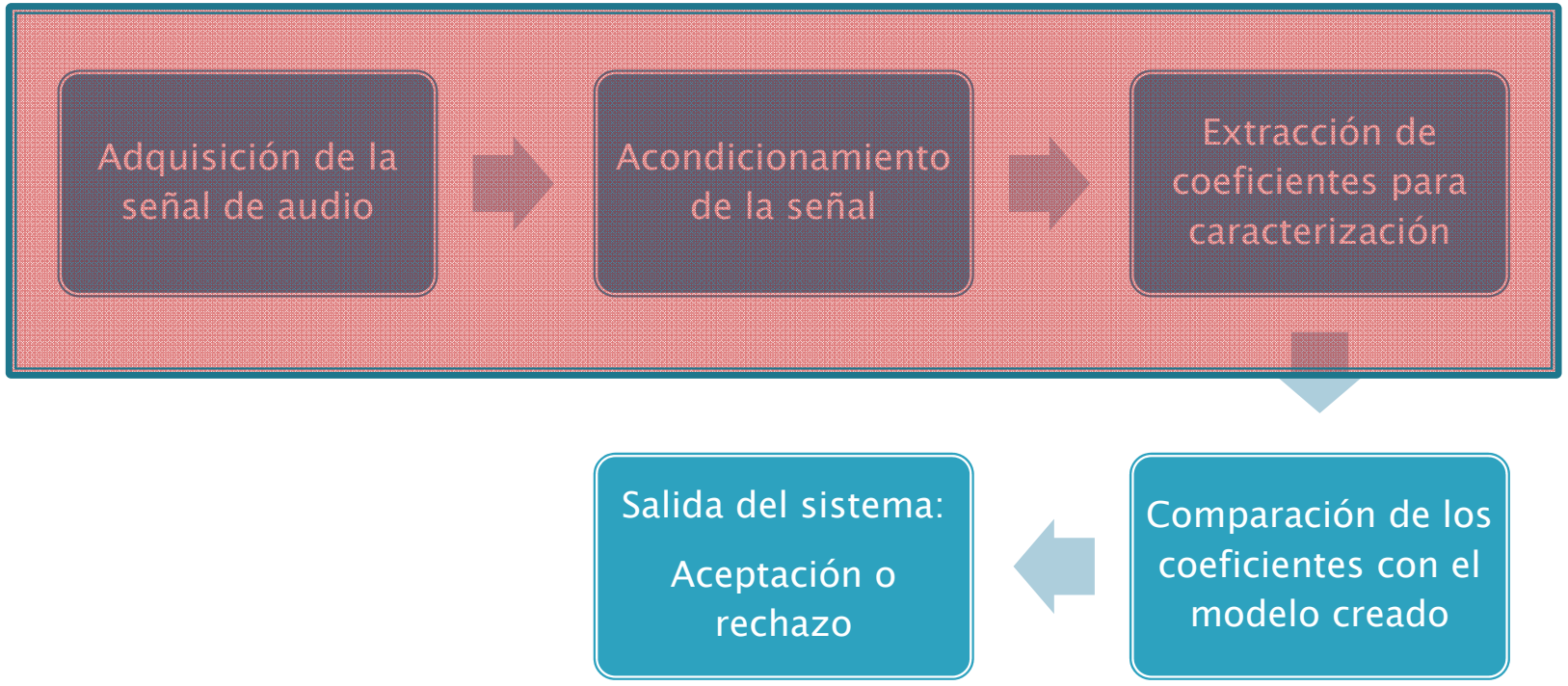
2 Etapas del sistema

Entrenamiento



2 Etapas del sistema

Test



2.1 Adquisición

- Grabación de un Código PIN de 4 dígitos
- Smartphone Alcatel OT 997
- Aplicación Android recForge
- 8 KHz, 16 bits, mono sin compresión (.wav)

2.2 Acondicionamiento

- Quitar componente DC

$$\bar{x} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]$$

- Normalizar [-1,1]

$$x_{norm}[n] = \frac{x[n]}{\max(|x[n]|)}$$

- Filtro paso alto
  - Corte en 75 Hz

2.2 Acondicionamiento

- Filtro pre-énfasis
  - +6 dB/Octava
- Enventanado
  - Ventana Hamming
  - Longitud de ventana variable
  - Solape variable
- Segmentado de tramas sonoras
  - Energia + ZCR
  - Detección de pitch (SIFT)

2.3 Extracción de coeficientes
MFCC

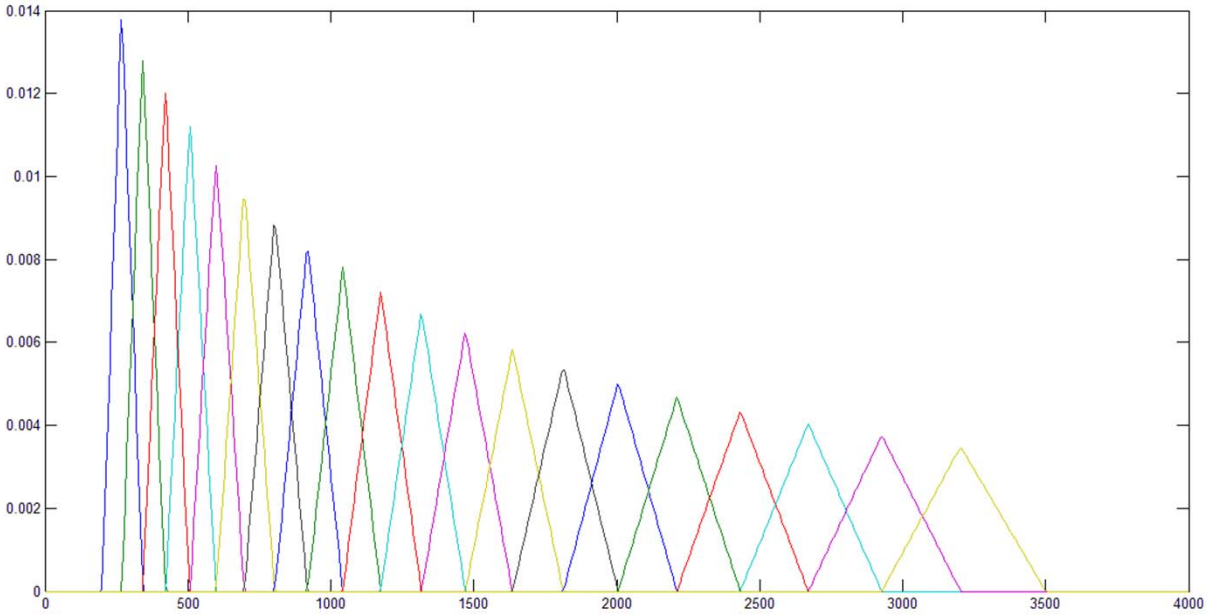
- DFT (FFT)

| [ ]| [ ]

- Banco de filtros en escala MEL

(———)

2.3 Extracción de coeficientes
MFCC



| | | [ ]

2.3 Extracción de coeficientes
MFCC

- Logaritmo

( )

- DCT

$$[\,] = \sum_{=0}^{-1} \left(\left(- \frac{1}{2} \right) \frac{}{} \right); \quad = 1, \ldots,$$

2.4 Extracción de coeficientes
LPCC

- Aplicar filtro LPC a la trama
  - Orden variable
- Transformación a dominio Cepstral

$$[\ ]$$

$$[\ ] \qquad \left(\quad - \right) \qquad [\quad]$$

$$[\ ] \qquad \left(\quad - \right) \qquad [\quad]$$

2.5 Extracción de coeficientes: Deltas

- Cálculo de coeficientes de variación en el tiempo de MFCC o LPCC
- Dan información sobre la coarticulación y cambios fonéticos

$$\Delta c_m[n] = \sum_{\tau=-\theta}^{\theta} \frac{\tau \, c_m[n+\tau]}{|\tau|} \qquad 1 \le n \le N$$

$$\Delta\Delta c_m[n] = \sum_{\tau=-\theta}^{\theta} \frac{\tau \, \Delta c_m[n+\tau]}{|\tau|} \qquad 1 \le n \le N$$

2.6 Extracción de coeficientes: CMVN

- Se normalizan los coeficientes respecto de la media y la varianza para eliminar los valores dispersos

2.7 Resumen de coeficientes

- 20 coeficientes cepstrales normalizados
- 20 coeficientes delta
- 20 coeficientes doble delta
- Un total de 60 coeficientes en cada trama de análisis.

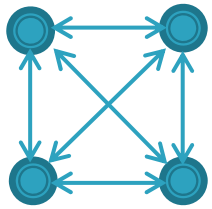
3 Creación del Modelo

- Necesarias 4 repeticiones del código.
- Se extrae la matriz de coeficientes para cada grabación.

Coeficiente / Trama	C_1	C_2	C_3	…	$\Delta\Delta C_N$
1					
2					
3					
4					
5					
6					
7					
8					
9					
…					
M					

3 Creación del Modelo

- Se comparan los vectores correspondientes al n-ésimo coeficiente de las 4 grabaciones.
- Un total de 6 comparaciones para cada coeficiente

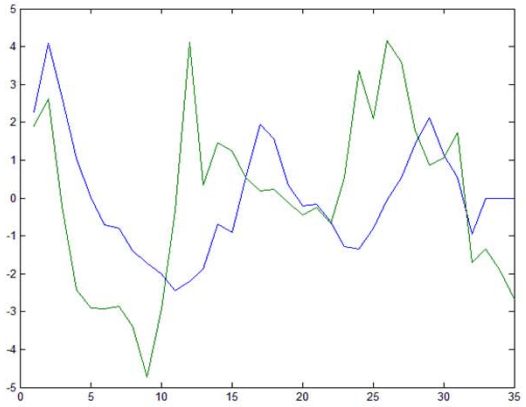


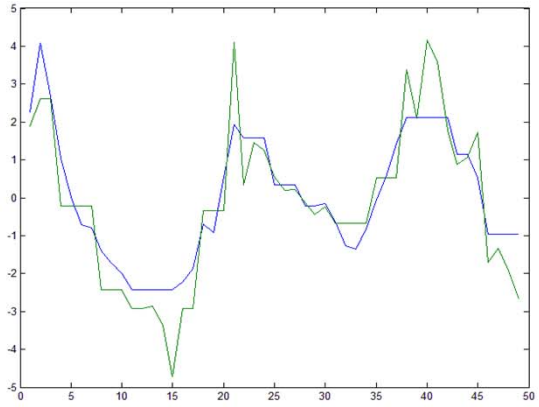
- Se coge el vector medio de las dos que más se parezcan.
- Se guarda dicho vector como coeficiente n del modelo del locutor
- Se realiza la misma operación para los 60 coef.

¿Cómo se compara?

4 Comparación DTW

- Alineamiento temporal dinámico
- Técnica para ajustar dos vectores similares que presentan desajustes temporales
- Se calcula la distancia o coste empleado para ajustar dos vectores
- Menor distancia en vectores con más parecido. (vectores extraídos de un mimo locutor)





4 Comparación DTW

- Se calculan las distancias entre valores del vector
- Como ambos vectores no tienen la misma longitud se hace zero padding en el más corto
- Se sigue el camino que requiera el menor coste
- Se suman los costes.

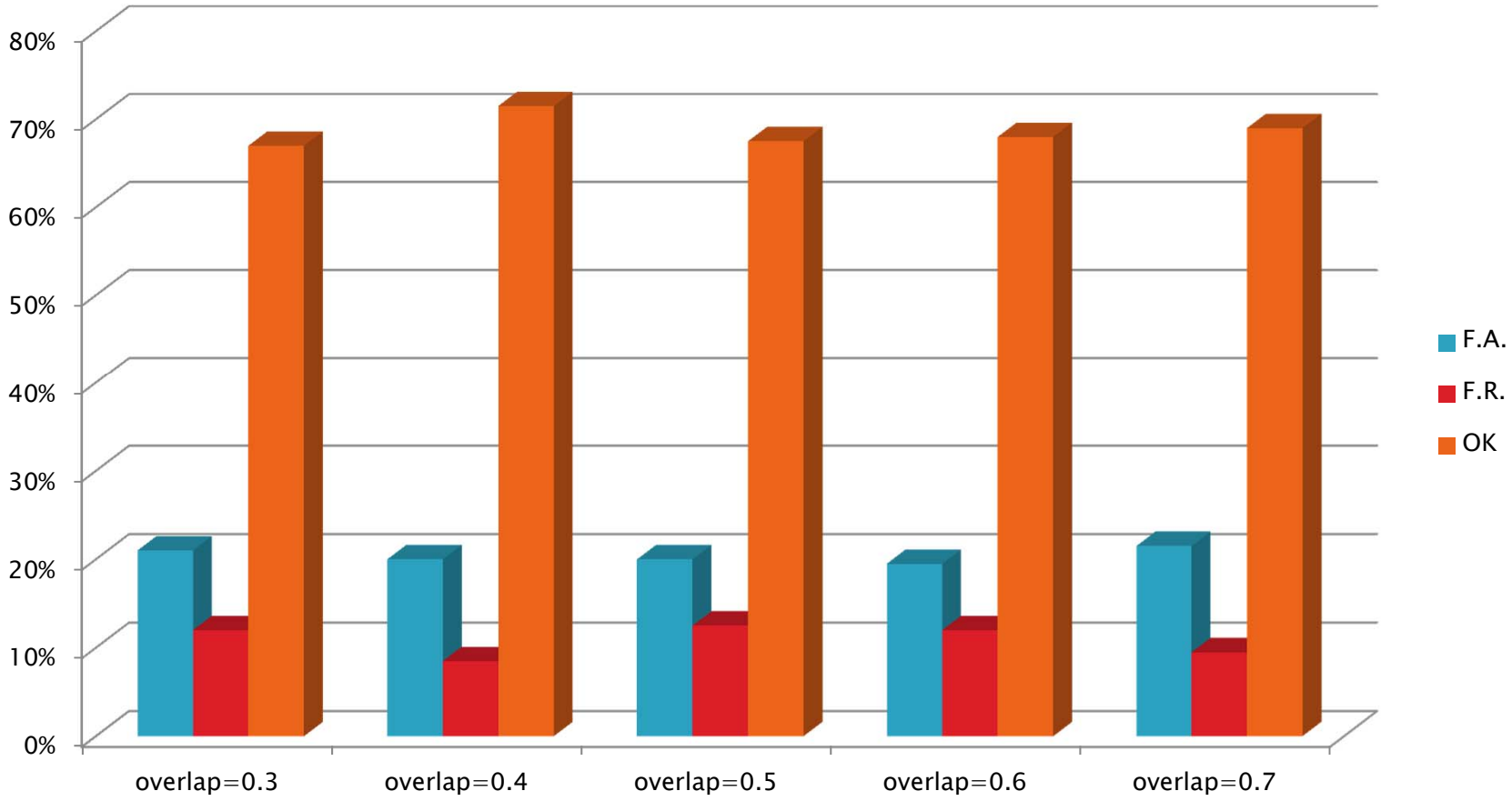
v_2 \ v_1	1	2	3	4	3	2	1	0
1	0	1	4	9	4	1	0	1
1	0	1	4	9	4	1	0	1
2	1	0	1	4	1	0	1	4
3	4	1	0	1	0	1	4	9
4	9	4	1	0	1	4	9	16
4	9	4	1	0	1	4	9	16
3	4	1	0	1	0	1	4	9
3	4	1	0	1	0	1	4	9
2	1	0	1	4	1	0	1	4
0	1	4	9	16	9	4	1	0

5 Umbral

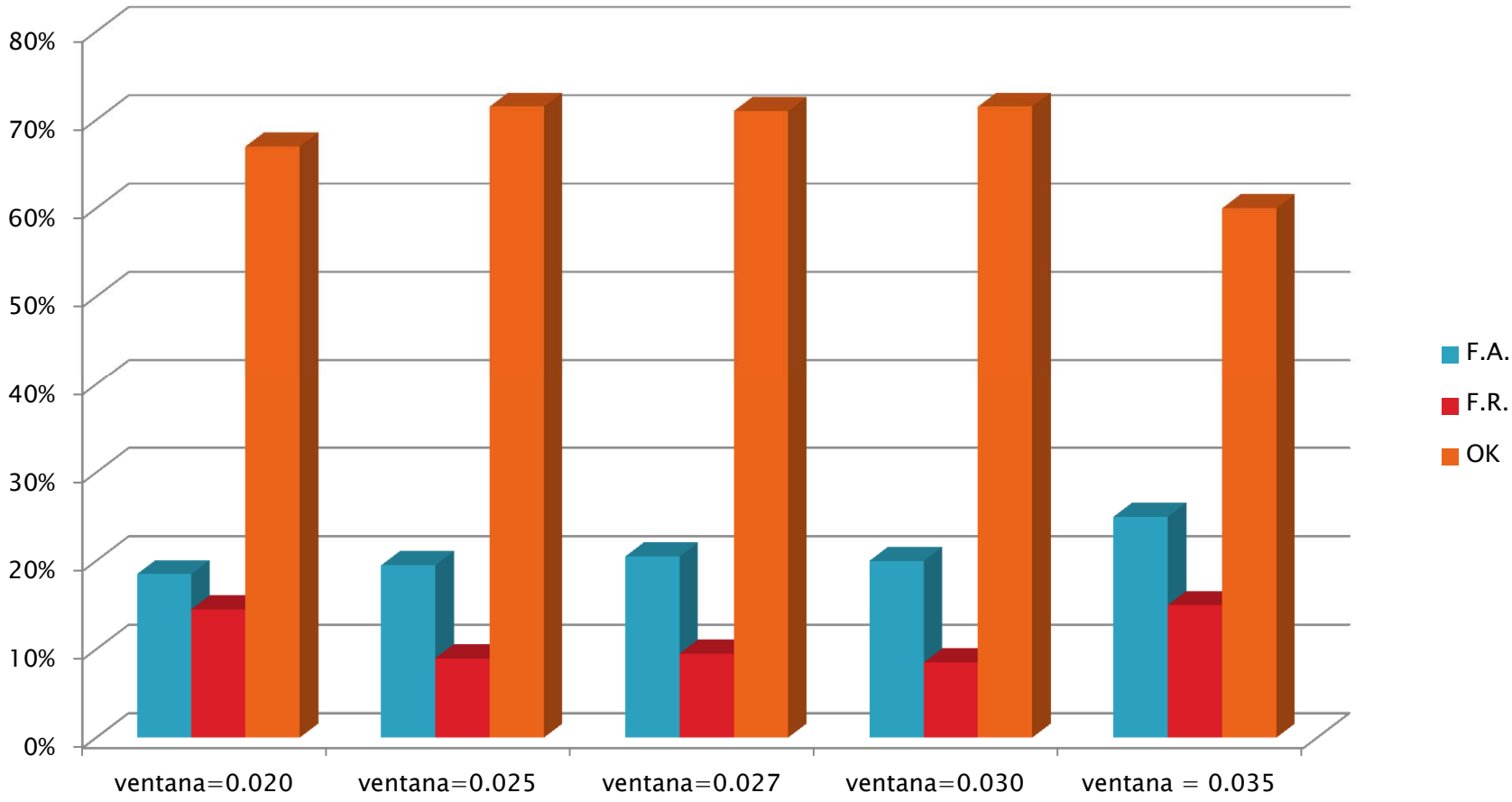
- Necesario realizar otras 4 grabaciones
- Se comparan con el modelo entrenado y se extraen 4 puntuaciones
- Se determina el umbral como la media mas la desviación típica de dichas puntuaciones.

$$\text{umbral} = \bar{x} + \sqrt{\frac{1}{4-1}\sum_{i=1}^{4}(x_i - \bar{x})^2}$$

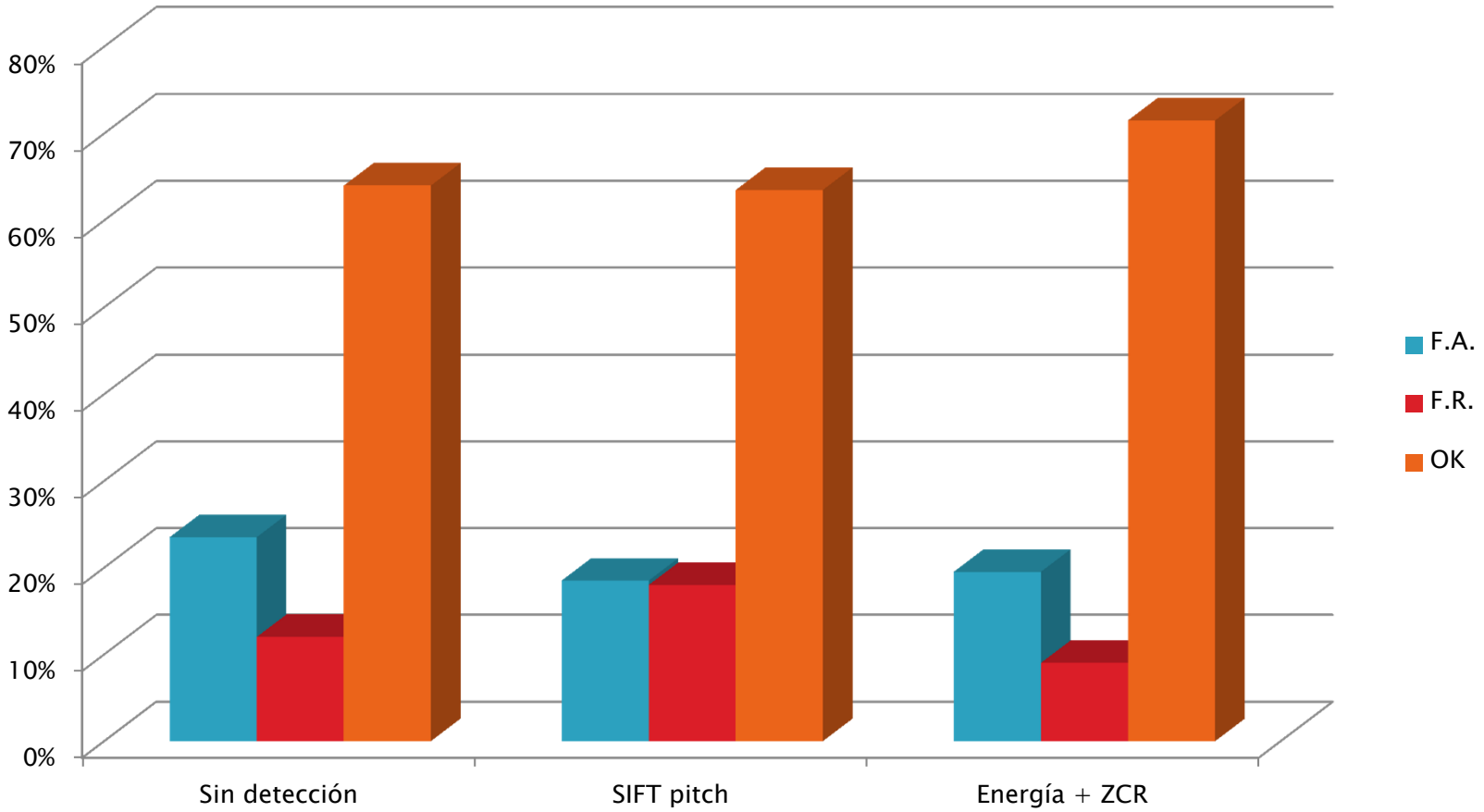
4 Resultados – Overlap



4 Resultados – Ventana

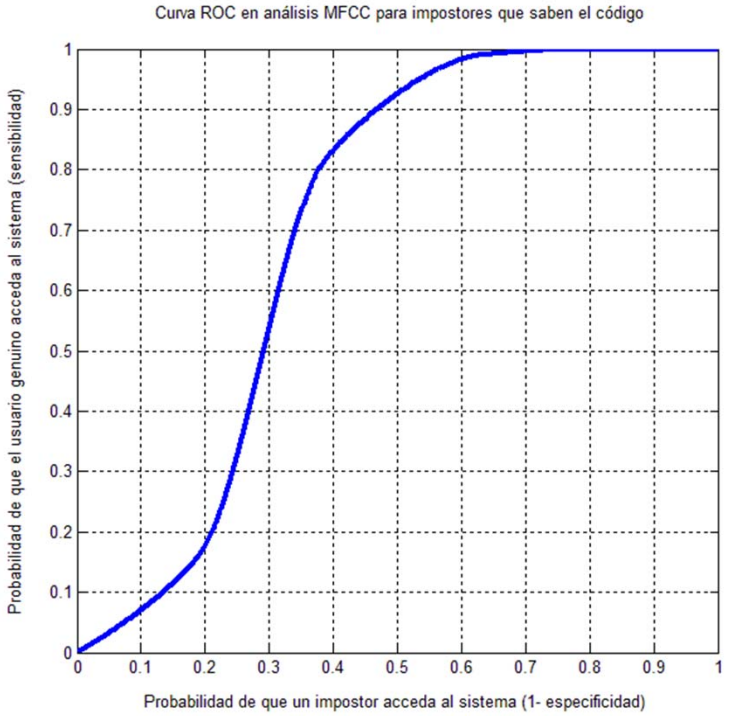


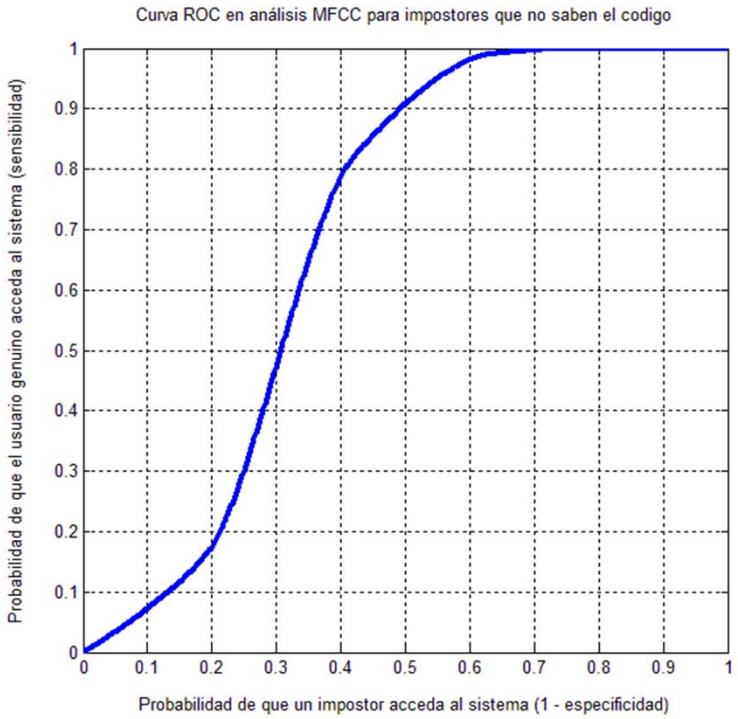
4 Resultados – Segmentado



4 Resultados – Curvas ROC

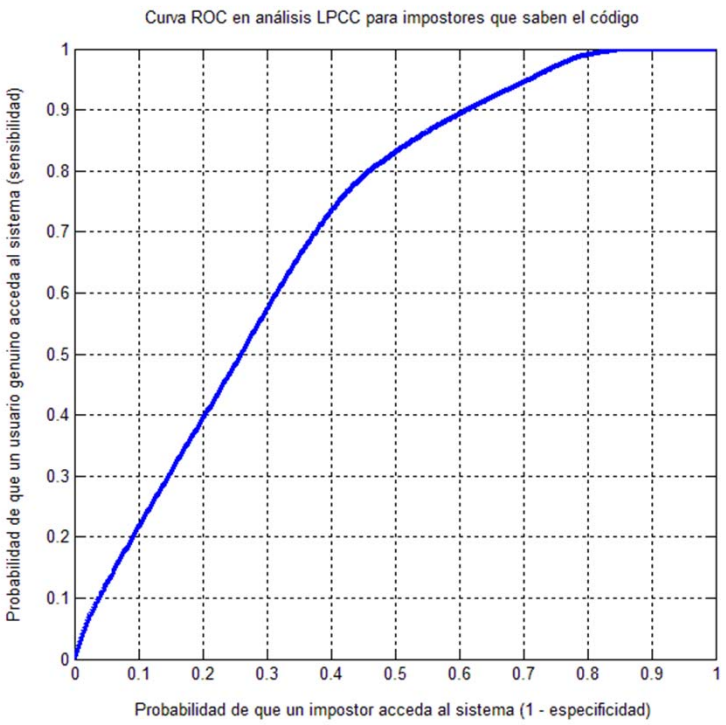
MFCC

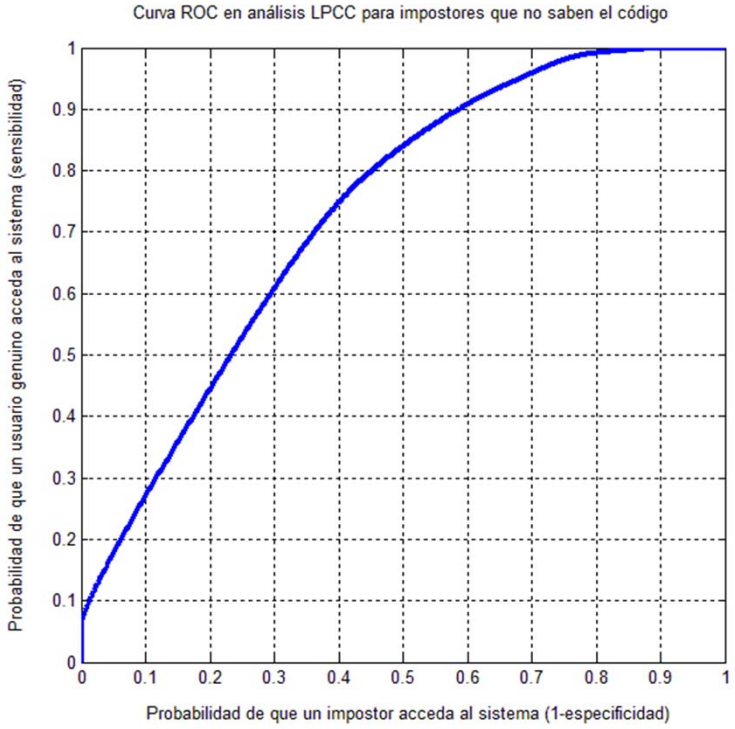




4 Resultados – Curvas ROC

LPCC





5 Conclusiones

- Valores óptimos de tamaño de ventana y solape: 0.025 s. y 40%
- Necesario segmentar las tramas sonoras.
- Necesario códigos más largos para que la robustez del sistema sea aceptable
- Mejor rendimiento con LPCC para el número de coeficientes utilizados.
- Es complicado desarrollar una aplicación de este tipo fiable para dispositivos móviles

6 Trabajos futuros

- Probar los algoritmos con segmentos de audio más largos.
- Probar los algoritmos cambiando el número de coeficientes
- Desarrollar nuevos algoritmos basados en modelos estadísticos para la creación del modelo del locutor y la comparación del test.
- Reforzar este sistema desarrollando otro que reconozca el mensaje.

FIN

¿Preguntas?