UNIVERSIDAD DE SEVILLA
ESCUELA SUPERIOR DE INGENIEROS
INGENIERÍA DE TELECOMUNICACIÓN

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
ÁREA DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

IMPLEMENTACIÓN DE UN SISTEMA DE RECONOCIMIENTO AUTOMÁTICO DE LOCUTORES BASADO EN LA TRANSFORMADA DE FOURIER - BESSEL

AUTORA: ARÁNZAZU CONESA PORRAS
TUTOR: RUBÉN MARTÍN CLEMENTE
JULIO 2001

Proyecto fin de carrera: **IMPLEMENTACIÓN DE UN SISTEMA DE RECONOCIMIENTO AUTOMÁTICO DE LOCUTORES BASADO EN LA TRANSFORMADA DE FOURIER - BESSEL**

Autora: Aránzazu Conesa Porras

Tutor: Rubén Martín Clemente

El tribunal nombrado para juzgar el proyecto fin de carrera arriba citado, compuesto por:

Presidente:

Vocal:

Vocal secretario:

acuerda otorgarle la calificación de:

Sevilla, a de de 2001

A mis padres.

Índice

Capítulo 1: Introducción..... 1

1.1: Objetivo del Proyecto 1
1.2: Introducción a los Sistemas de Reconocimiento 1
1.3: La Señal de Voz..... 5
1.4: Reconocimiento de la Voz..... 11

Capítulo 2: Reconocimiento de Locutores. Estado del Arte Actual..... 17

2.1: Introducción..... 17
2.2: Principios del Reconocimiento de Locutores 18
2.3: Parámetros Característicos..... 20
2.3.1: Definición de la Cepstrum Compleja 21
2.3.2: Cálculo de la Cepstrum usando la DFT..... 25
2.3.3: Aplicaciones de la Cepstrum al Procesado de Voz. Estimación de los Parámetros del Modelo de Voz 26
2.3.3.1: Parametrización de la Envolvente del Espectro..... 30
2.3.3.2: Extracción de la Característica 31
2.3.3.3: Clasificador Cuantizador de Vectores 36
2.3.3.4: Otros Parámetros Característicos para Identificación Robusta de Locutores sobre Canales Telefónicos..... 38
2.3.3.4.1: Descripción del Sistema de Referencia..... 40
2.3.3.4.2: Experimentos Preliminares 41
2.3.3.4.3: Extracción de la Característica para la Identificación del Locutor..... 43
2.3.3.4.3.1: Descripción del Front-End Básico..... 43
2.3.3.4.3.2: Diseño Perceptualmente Motivado del Banco de Filtros..... 44
2.3.3.4.3.3: Optimización de los Parámetros del Front-End para la Cepstrum Basada en Bancos de Filtros 46
2.3.3.4.3.4: Pendiente Espectral en Frecuencia basada en Bancos de Filtros 47
2.3.3.4.3.5: Optimización de los Parámetros del Front-End para la Pendiente Espectral..... 49

2.3.3.4.3.6: Combinación de Diferentes Características....50
2.3.3.4.3.7: Extensión al Reconocimiento de Voz....51
2.3.3.4.4: Problema del Modelo de Transformación: Métodos de Compensación de Canal basados en Modelos....52
2.3.3.4.4.1: Cantidad de Datos de Entrenamiento y Tamaño del Modelo....54
2.3.3.4.4.2: Distribución de la Varianza Sintética....55
2.3.3.4.4.3: Transformaciones Afines de las Varianzas....57
2.3.3.4.4.4: Traducción a un 'Target' Fijo....58
2.3.3.4.4.5: Transformación de la Varianza con Múltiples Características....60
2.3.3.4.4.6: Extensión a Reconocimiento de Locutores de Conjunto Abierto....60
2.3.3.4.5: Conclusiones de los Métodos Planteados....60
2.4: Métodos de Reconocimiento de Locutores con Dependencia del Texto.... 61
2.5: Métodos de Reconocimiento de Locutores con Independencia del Texto 61
2.6: Método de Reconocimiento de Locutores con Texto Propuesto 63

Capítulo 3: Descripción Global de la Solución Aportada: Identificación de Locutores usando Propiedades Basadas en la Expansión en Serie de Fourier – Bessel.... 65

3.1: Introducción.... 65
3.2: Funciones de Bessel.... 66
3.2.1: Método Extendido de las Series de Potencias.(Ecuación de Índice Inicial)....66
3.2.2: Ecuaciones de Bessel. Funciones de Bessel de Primera Clase....67
3.2.3: Conjuntos Ortogonales de Funciones....74
3.2.3.1: Series de Funciones Ortogonales....76
3.2.3.2: Ortogonalidad con Respecto a una Función Peso....77
3.2.4: Problema de Sturm-Liouville 79
3.3: Ortogonalidad de las Funciones de Bessel.... 86
3.4: Transformada de Fourier-Bessel.... 91
3.5: Representación de Señales usando Funciones de Bessel.... 93
3.6: Representación de Fourier – Bessel de Formas de Onda de Voz.... 94
3.6.1: Identificación de Resultados de Características Basadas en la Transformada de FB....97
3.7: Discusión.... 101
3.8: Cuantización Vectorial.... 103
3.8.1: Algoritmo para Diseño de Cuantizadores Vectoriales 109

3.8.1.1: Cuantizadores de Bloques 111
3.8.1.2: Medidas de Distorsión 111
3.8.1.3: Funcionamiento 116
3.8.1.4: Cuantización Óptima 117
Algoritmo (Distribución Conocida) 122
Algoritmo (Distribución Desconocida) 125
3.8.1.5: Elección de $\hat{A}_0$ 128
3.8.1.6: Condición Inicial por "Splitting" 129
3.8.1.7: Ejemplos 130
Fuentes Gaussianas 130
Ejemplo de Lloyd 132
Ejemplo de Voz 133

Capítulo 4: Descripción de Subsistemas 137

4.1: Obtención de Muestras: Creación de la Base de Datos 137
4.2: Cálculo de Coeficientes de Fourier-Bessel: Extracción de la Característica 138
4.2.1: Selección de Coeficientes 139
4.2.2: Ortogonalidad de las Funciones de Bessel 140
4.2.3: Regla de Simpson 141
4.2.4: Recontrucción de la Señal de Voz 143
4.2.5: Extracción de la Característica 144
4.2.6: Cálculo de Centroides: Cuantizador Vectorial 145
4.3: Fases del Sistema de Reconocimiento 146
4.3.1: Modo de Conjunto Cerrado 147
4.3.1.1: Fase de Entrenamiento 147
4.3.1.2: Fase de Prueba 148
4.3.2: Modo de Conjunto Abierto 150
4.3.2.1: Fase de Entrenamiento 150
4.3.2.2: Fase de Prueba 151
4.4: Descripción Funcional 152

Capítulo 5: Experimentos y Resultados 161

5.1: Introducción 161
5.2: Reconstrucción de la Señal de Voz 166
5.3: Pruebas de Reconocimiento de Conjunto Cerrado 170
5.4: Pruebas de Reconocimiento de Conjunto Abierto 182

Capítulo 6: Conclusiones y Futuras Líneas de Investigación 183

6.1: Conclusiones 183

6.2: Futuras Líneas de Investigación 185

6.3: Técnicas de Normalización 186

6.3.1: Normalización en el Dominio Paramétrico 186

6.3.2: Normalización en el Dominio Distancia/Similitud 187

6.4: Cuantización Binaria de Vectores Característicos para Identificación Robusta de Locutores con Texto Independiente 188

6.4.1: Introducción 189

6.4.2: Cuantización Binaria 190

6.4.3. Investigación de la Distorsión Media 193

6.4.4: Esquema de Identificación de Locutores 196

6.4.5: Estimación del Almacenamiento y Cálculo 200

6.4.6: Sumario 201

Capítulo 7: Planos:Listado de los Programas 203

7.1: Fase de Entrenamiento 203

7.1.1: Generación de la Base de Datos 203

7.1.2: Extracción de la Característica: Cálculo de los Coeficientes de Fourier-Bessel 260

7.1.3: Cuantización Vectorial 268

7.1.4: Modo de Reconocimiento de Conjunto Abierto: Cálculo del Umbral 276

7.1.5: Distorsión entre Locutores: Cálculo de la Distancia Euclídea 278

7.2: Fase de Identificación 283

7.2.1: Extracción de la Característica del Locutor de Prueba 305

7.2.3: Determinación de la Identidad del Locutor 307

Anexo I.: Glosario 311

Anexo II.: Referencias Bibliográficas 315

Capítulo 1: Introducción

1.1: Objetivo del Proyecto

El presente proyecto tiene por objeto realizar el estudio e implementación de un sistema de reconocimiento automático de locutores que emplea un método de transformación alternativo al convencionalmente usado, la transformada Cepstrum, que se basa en la expansión en serie de Fourier-Bessel para caracterizar la señal de voz.

1.2: Introducción a los Sistemas de Reconocimiento

Los interfaces del lenguaje hablado con los ordenadores son un tópico que ha seducido a ingenieros y a científicos de la voz a lo largo de cinco décadas. Para muchos, la habilidad de conversar libremente con una máquina representa el último desafío para nuestro entendimiento de los procesos de producción y percepción involucrados en la

comunicación de voz humana. Además de ser un tópico provocativo, los interfaces del lenguaje hablado se están rápidamente convirtiendo en una necesidad. En un futuro cercano, las redes interactivas proporcionarán fácil acceso a una riqueza de información y servicios que afectarán fundamentalmente al modo en que la gente trabaja, juega y dirige sus asuntos diarios. Hoy, tales redes se limitan a personas que pueden leer y tener acceso a los ordenadores –una parte relativamente pequeña de la población, incluso en los países más desarrollados. Los avances en la tecnología del lenguaje humano son necesarios para que el ciudadano medio se comunique con las redes usando técnicas de comunicación naturales que hacen uso de dispositivos corrientes, tales como teléfonos y televisores. Sin los avances fundamentales en los interfaces centrados en el usuario, una gran parte de la sociedad se abstendrá de participar en la era de la información, dando como resultado una mayor estratificación de la sociedad y una trágica pérdida del potencial humano.

Un interfaz hablado en el propio lenguaje del usuario, es ideal porque es la forma de comunicación humana más natural, flexible, eficiente y económica. El acceso hablado a ordenadores incorpora muchas tecnologías y aplicaciones diferentes, como se muestra en la Figura 1.1. En algunos casos, como mostramos en la parte inferior de la figura, uno no está interesado en el contenido lingüístico subyacente, sino en la identidad del locutor, o en el lenguaje hablado. El reconocimiento de voz puede implicar identificar a un locutor específico de entre una población conocida, lo cual tiene implicaciones en medicina legal, o verificar la identidad de un usuario esperado, activando por tanto el acceso controlado a locales (p. e., una sala de ordenadores o centro de control) y servicios (p. e., transacciones bancarias por voz).

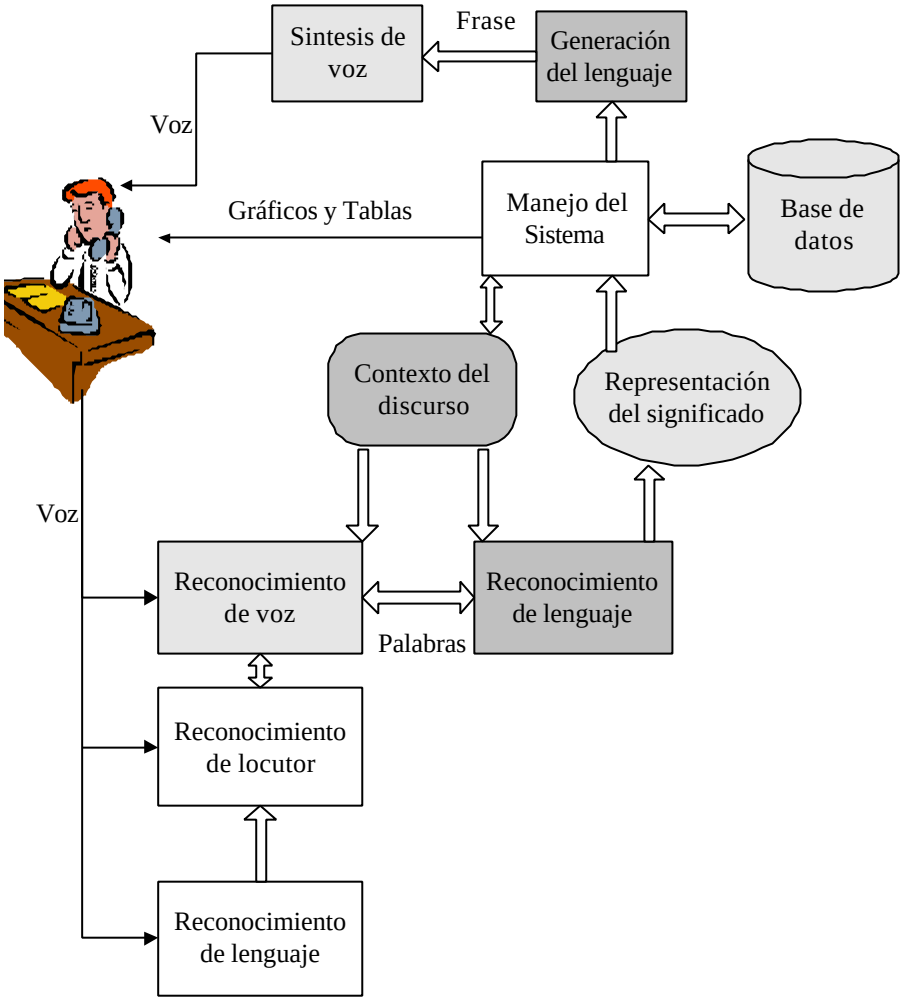


Figura 1.1: Tecnologías para interfaces de lenguaje hablado

Cuando uno piensa en hablar a los ordenadores, la primera imagen es normalmente el reconocimiento de voz, la conversión de una señal acústica a una cadena de palabras. Después de muchos años de investigación, la tecnología del reconocimiento de voz está comenzando a cruzar el umbral de la practicidad. La última década ha presenciado mejoras dramáticas en la tecnología del reconocimiento de voz, hasta el punto de que ya están disponibles algoritmos y sistemas de alto rendimiento. En algunos casos, la transición de las demostraciones en laboratorio al desarrollo comercial ya ha comenzado. Las capacidades de acceso por voz están emergiendo hasta tal punto que pueden proporcionar funciones tales como marcación por voz (p. e., Llama a casa), desvío de llamadas (p. e., Querría hacer una llamada a cobro revertido), entrada de datos

(p. e., dar el número de la tarjeta de crédito), y preparación de documentos estructurados (p. e., informe de una radiología). El reconocimiento de voz implica a varias tecnologías. En primer lugar, la señal digitalizada debe ser transformarda en un conjunto de medidas. Deben aportarse técnicas que permitan al sistema desarrollar robustez en presencia de variaciones del transductor y del entorno, y técnicas para adaptarse a estas variaciones. A continuación, los diferentes sonidos de voz se deben modelar apropiadamente. La técnica de modelado acústico más extendida es conocida por Modelo Oculto de Markov (*Hidden Markov Modeling*, HMM). La búsqueda de la respuesta final implica el empleo de especificaciones del lenguaje.

El reconocimiento de voz es un problema desafiante por sí mismo, con un conjunto de aplicaciones bien definido. Sin embargo, muchas tareas que se prestan al acceso por voz – concertar un viaje o seleccionar una película - son de hecho ejercicios para la resolución de problemas interactivos. La solución se construye incrementalmente, tanto con el usuario como con el ordenador jugando un papel activo en la “conversación”. Por tanto, se deben desarrollar varias tecnologías basadas en entrada y salida de datos, e integrarse para alcanzar su objetivo. La Figura 1.1 muestra los principales componentes de un sistema conversacional típico. La entrada de voz es procesada en primer lugar por la componente de reconocimiento de voz. La componente de lenguaje natural, que trabaja de acuerdo con el reconocedor, produce una representación de significado. La representación de significado puede ser usada para recobrar la información apropiada en el formato de texto, tablas y gráficos. Si la información de la declaración es insuficiente o ambigua, el sistema puede elegir preguntar al usuario para clarificar. La generación del lenguaje natural y la síntesis de voz, se puede usar para producir respuestas habladas que pueden servir para clarificar la información recibida. Durante todo el proceso, la información intercambiada se mantiene y realimenta a los componentes de reconocimiento de voz y de comprensión de lenguaje, para que las frases se puedan entender correctamente en el contexto.

1.3: La Señal de Voz

La voz es una onda de presión que se transmite al medio. La señal de voz se produce excitando el sistema del tracto vocal con una fuente de excitación de banda ancha. Podemos modelar el proceso de generación de la señal de voz como un sistema de transmisión acústico.

Los sonidos pueden clasificarse de forma genérica en sonoros y no-sonoros. En los primeros se abren y cierran las cuerdas vocales, modificando el área de la tráquea y produciendo un tren de impulsos cuasi-periódicos. El período o frecuencia fundamental de este tren de impulsos se conoce con el nombre de *pitch*, y su valor está comprendido entre 50 y 400 Hz para los hombres y es superior en mujeres y niños. En los sonidos no-sonoros el aire fluye libremente hasta alcanzar el tracto vocal al permanecer abiertas las cuerdas vocales. Finalmente, la variación voluntaria del tracto vocal, junto con el estado variante de las cuerdas, produce la voz.

El tracto vocal actúa como una cavidad resonante para los sonidos sonoros, estando centradas las frecuencias de resonancia para la mayoría de la gente en 500 Hz y sus armónicos pares. Esta resonancia produce grandes picos en el espectro resultante, a los cuales se les llama formantes. Además la señal tiene una naturaleza paso-baja y a partir de unos 4 KHz comienza a predominar el ruido.

En cambio, el segmento de voz no-sonoro presenta una estructura ruidosa tanto en el dominio del tiempo como en el de la frecuencia, no teniéndose formantes. Además la energía de la señal es mucho menor que la de los sonidos sonoros.

La Figura 1.2 muestras un esquema de las partes fundamentales del cuerpo humano involucradas en el proceso del habla.

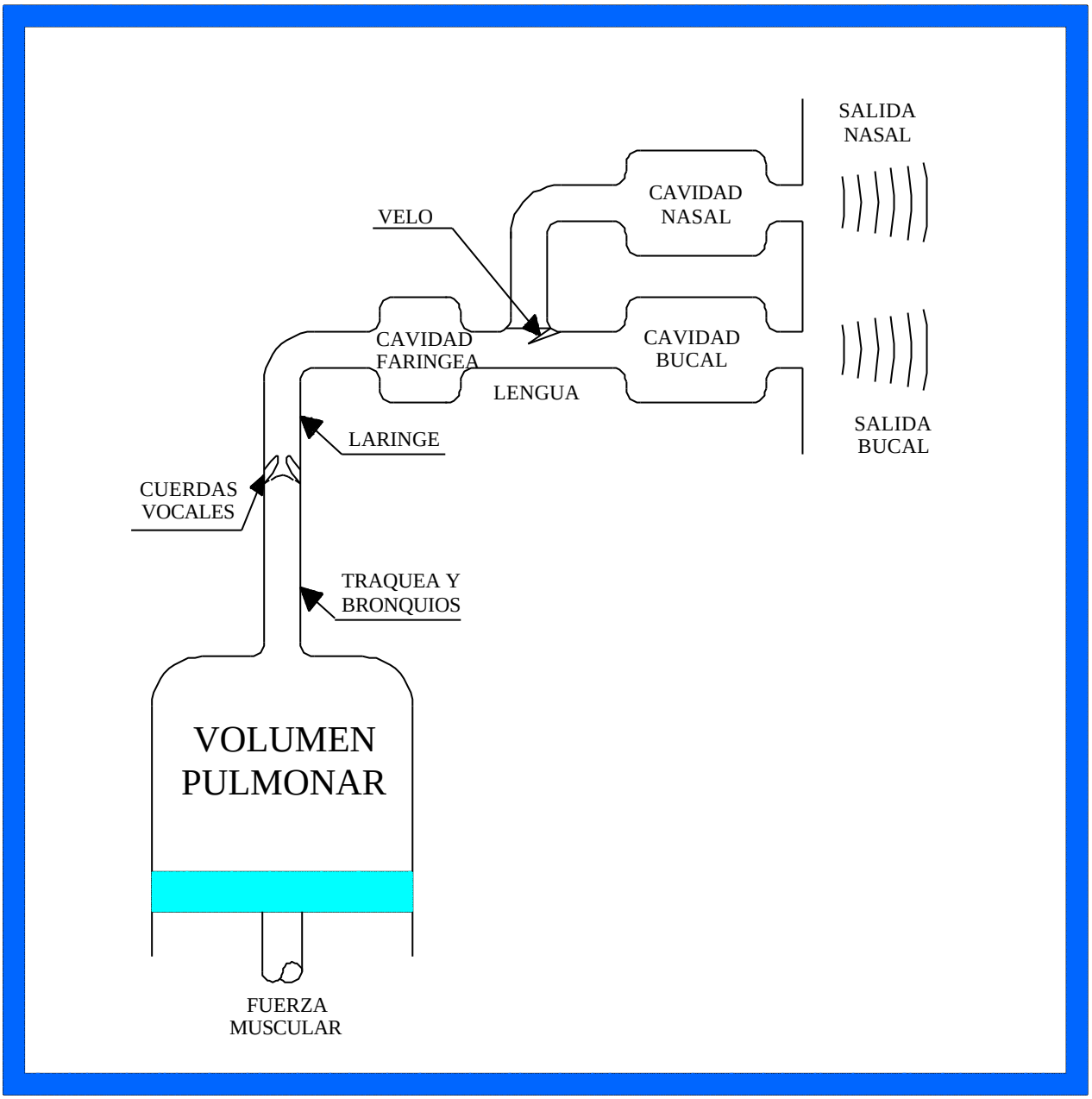


Figura 1.2: Esquema de las partes del cuerpo involucradas en el proceso del habla

En la Figura 1.2 se pueden ver dos partes importantes de la vocalización; la musculatura, produce rápidos movimientos de aire que crean sonidos y el tracto vocal que modela estos sonidos. Este 'modelaje' de los sonidos varía de persona a persona y de hombre a mujer. Estas variaciones producidas por el tracto vocal, nos ayudan a distinguir entre diferentes individuos, pero dificultan el proceso de reconocimiento de la voz.

Existen tres clases básicas de sonidos de voz que corresponden a diferentes formas de excitación del tracto vocal[1]. Concretamente:

- Sonidos sonoros, se producen excitando el tracto vocal con pulsos cuasi-periódicos de aire provocados por la apertura y cierre de la glotis.
- Sonidos fricativos, se producen al formar una constricción en algún lugar del tracto vocal y forzando el aire a través de la constricción, para que se cree la turbulencia, creando de ese modo una excitación ruidosa.
- Sonidos plosivos, se producen cerrando completamente el tracto vocal, subiendo la presión por detrás de la cavidad, y después aflojando la presión bruscamente.

En cada caso, la señal de voz se produce excitando el sistema del tracto vocal (un sistema de transmisión acústica) con una excitación de banda ancha. El tracto vocal varía su caída de forma relativamente lenta con el tiempo, y por ello se puede modelar como un filtro de lenta variación en el tiempo que impone las propiedades de su respuesta en frecuencia sobre el espectro de la excitación. El tracto vocal se caracteriza por sus frecuencias naturales (denominadas formantes), que corresponden a resonancias de su respuesta en frecuencia.

Si asumimos que las fuentes de excitación y la caída del tracto vocal son independientes, llegamos al modelo en tiempo discreto de la Figura 1.3 como representación de la forma de onda de la voz muestreada. En este modelo, se asume que

[1] *Capítulo 12 "Discrete Time Signal Processing". Alan V. Oppenheim, Ronal W. Schafer. Ed. Prentice Hall [6].*

las muestras de la señal de voz están a la salida de un sistema en tiempo discreto variante en el tiempo, que modela las resonancias del sistema del tracto vocal. El modo de excitación del sistema conmuta entre impulsos periódicos y ruido aleatorio, dependiendo del tipo de sonido que se produzca.

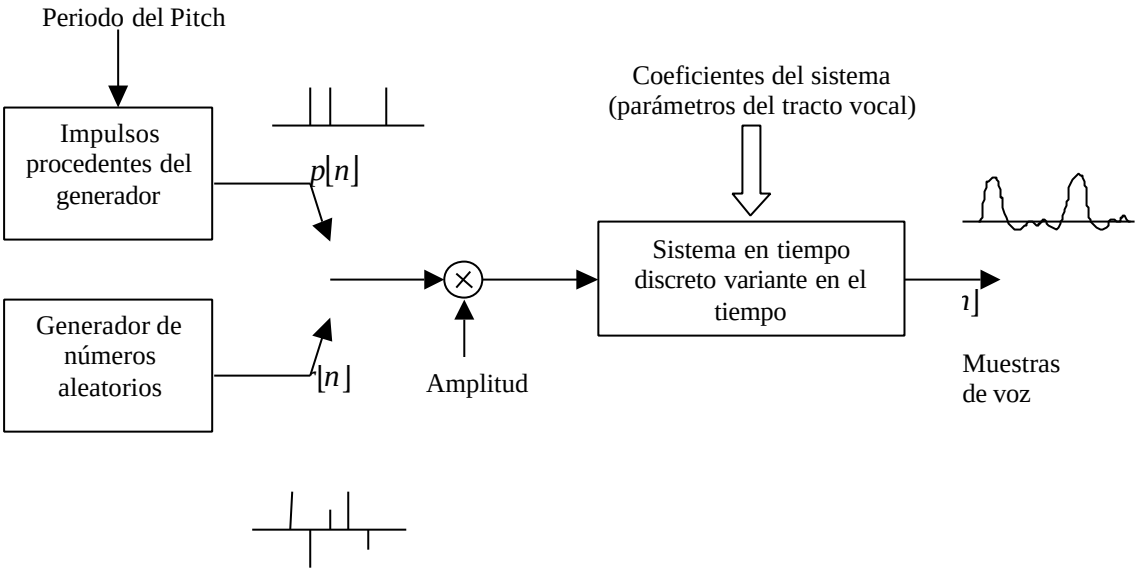


Figura 1.3: Modelo de producción de voz en tiempo discreto

Dado que la caída del tracto vocal cambia lentamente en la voz continua, es razonable asumir que el sistema en tiempo discreto del modelo tiene propiedades fijas a lo largo de un intervalo de tiempo del orden de 10 ms. Por tanto, el sistema en tiempo discreto se puede caracterizar en cada intervalo de tiempo por una respuesta al impulso o una respuesta en frecuencia o un conjunto de coeficientes para un sistema IIR. Concretamente, el modelo de la función del sistema del tracto vocal toma la forma siguiente

$$V(z) = \frac{\sum_{k=0}^{K} b_k z^{-k}}{\sum_{k=0}^{P} a_k z^{-k}} \tag{1.1}$$

o, equivalentemente,

$$V(z) = \frac{Az^{-K_0} \prod_{k=1}^{K_i} \left(1 - a_k z^{-1}\right) \prod_{k=1}^{K_o} \left(1 - b_k z^{-1}\right)}{\prod_{k=1}^{\lceil P/2 \rceil} \left(1 - r_k e^{jq_k} z^{-1}\right)\left(1 - r_k e^{-jq_k} z^{-1}\right)} \tag{1.2}$$

donde las cantidades $r_k e^{jq_k}$ son las frecuencias naturales complejas del tracto vocal que, por supuesto, son dependientes de la caída del tracto vocal y consecuentemente son variantes en el tiempo. Los ceros de $V(z)$ justifican la forma de onda del pulso glotal de duración finita y los ceros de trasmisión provocados por las constricciones del tracto vocal en la creación de los sonidos sonoros nasales y los fricativos. Tales ceros a menudo no se incluyen porque es muy difícil estimar sus posiciones a partir de la forma de onda de la voz exclusivamente, y porque se ha mostrado que la caída espectral de la señal de voz se puede modelar fielmente sin usar ceros si incluimos polos extra hasta justo el número necesitado para justificar las resonancias del tracto vocal. Los ceros se incluyen en nuestro análisis porque son necesarios para una representación fiable de la voz.

El sistema del tracto vocal es excitado por una secuencia de excitación p[n], que es un tren de impulsos cuando modela los sonidos sonoros de voz, y una secuencia de ruido pseudoaleatorio cuando modela los sonidos sordos de voz tales como fricativos y plosivos.

Muchos de los problemas fundamentales del procesado de voz se reducen a la estimación de los parámetros del modelo de la Figura 1.3. Estos parámetros son los siguientes:

- Los coeficientes de $V(z)$ de la ecuación 1.1 o las localizaciones de polos y ceros en la ecuación 1.2
- El modo de excitación del sistema del tracto vocal, es decir, *un tren de impulsos periódico o ruido aleatorio*
- La amplitud de la señal de excitación
- El periodo de la excitación de voz cuando es periódica

Existen dos tipos de información necesarios para proporcionar al procesado acústico datos para generar voz natural. Estos dos tipos de información se conocen como información segmental y suprasegmental.

1) La información segmental es la asociada a la cadena de sonidos que componen el mensaje. Los sonidos que se pueden producir con los órganos de articulación son variadísimos, aún considerando un único locutor. Pero cada idioma ha seleccionado una serie limitada de "sonidos ideales" aceptados por todos para su uso en el habla. Tenemos así consciencia de una serie limitada de representaciones abstractas fáciles de aprender y percibir. Estas representaciones abstractas se denominan fonemas y su número depende del idioma considerado; en el caso del Español existen 24 fonemas. Sin embargo, si nos situamos en el plano del habla, debido al carácter continuo del habla hay variaciones en el punto, modo y carácter sordo/sonoro de cada "sonido ideal". Estas condiciones dan lugar a los diferentes alófonos de un fonema, así por ejemplo los oclusivos sonoros se fricatizan en contextos vocálicos, lo que complica aún más el diseño de sistemas de reconocimiento o síntesis de voz de alta calidad.

2) La información suprasegmental es aquella que queda asociada a la prosodia. Y, en consecuencia, refleja tanto elementos lingüísticos (carácter de la frase, pausas, acentos, agrupación en elementos de significado, ...), como elementos no lingüísticos (características personales del locutor, estado de ánimo, ...). Esta información es clave para conseguir una alta tasa de éxito en los sistemas de reconocimiento de locutores y de voz. Y, generalmente, viene codificada a través de tres parámetros acústicos de la señal de voz:

   a) La evolución temporal de la frecuencia fundamental, que es el correlato físico más importante desde el punto de vista perceptivo.

   b) La duración de los segmentos o sonidos que componen la frase.

   c) La curva de energía de la señal acústica (la menos importante desde el punto de vista perceptivo).

1.4: Reconocimiento de la Voz

El reconocimiento de la voz es el proceso de convertir una señal acústica, capturada por un micrófono o un teléfono, en un conjunto de palabras. Las palabra reconocidas pueden ser el resultado final, para aplicaciones tales como comandos & control, entrada de datos, y preparación de documentación. Estas pueden también servir como entrada de un proceso más lingüístico para conseguir el entendimiento de la voz.

Los sistemas de reconocimiento de voz se pueden caracterizar por muchos parámetros, de los cuales, algunos de los más importantes se muestran en la Tabla 1.1[2].

[2] *Datos extraídos de "Speech Recognition", estudio elaborado por Victor Zue y Wayne Ward, referencia http://www.cslu.cse.ogi.edu/HLTsurvey/ch1node4.html [11]*

Parámetros	Rango
Modo del habla	De palabras aisladas a voz continua
Estilo del habla	De texto leído a voz espontánea
Alcance	De dependencia a independencia del locutor
Vocabulario	De pequeño (<20 pal.) a amplio (>20.000 pal.)
Modelo de lenguaje	De estado finito o sensible al contexto
Perplejidad	De pequeña (<10) a grande (>100)
SNR	De alta (>30 dB) a baja (<10 dB)
Transductor	De micrófono cancelador de ruido a teléfono

Tabla 1.1: Parámetros típicos empleados para caracterizar la capacidad de los sistemas de reconocimiento de voz

Un sistema de reconocimiento de palabras aisladas requiere que el locutor realice brevemente pausas entre las palabras, mientras que un sistema de reconocimiento de voz continua no lo necesita. Espontáneamente, o generados de forma improvisada, la voz contiene cambios de fluidez, y es mucho más difícil de reconocer que la voz leída de un escrito. Algunos sistemas requieren la inscripción o registro del locutor –un usuario debe proporcionar muestras de su voz antes de usarlas, mientras que otros sistemas son conocidos como 'independientes del locutor', y en ellos no es necesario dicho registro. Algunos de los restantes parámetros dependen de la tarea específica. El reconocimiento es generalmente más difícil cuando los vocabularios son amplios, o

tiene muchas palabras de sonidos similares. Cuando la voz se produce como una secuencia de palabras, se usan modelos de lenguaje o gramáticas artificiales para restringir la combinación de palabras.

El modelo de lenguaje más simple se puede especificar como una red de 'estado finito', donde las palabras permisibles que siguen a cada una de las palabras vienen dadas explícitamente. Modelos de lenguaje más generales que se aproximan al lenguaje natural se especifican en términos de una gramática de 'contexto sensible'.

Una medida popular de la dificultad de la tarea, que combina el tamaño del vocabulario y el modelo de lenguaje, es la *perplejidad*, definida aproximadamente como la media geométrica del número de palabras que pueden seguir a una palabra después de aplicar el modelo de lenguaje. Finalmente, existen algunos parámetros externos que pueden afectar al funcionamiento de los sistemas de reconocimiento de voz, incluyendo las características del ruido ambiental y el tipo y ubicación del micrófono.

El reconocimiento de voz es un problema difícil, debido ampliamente a la cantidad de fuentes de variabilidad asociadas a la señal. En primer lugar, las realizaciones acústicas de los fonemas, las unidades más pequeñas de sonido de las que están compuestas las palabras, son altamente dependientes del contexto en el que aparecen.

En segundo lugar, las *variabilidades acústicas* pueden resultar de cambios en el entorno así como en la posición y características del transductor. En tercer lugar, las variabilidades incluídas en el locutor pueden surgir de cambios en el estado físico y emocional del locutor, la cadencia del habla, o la calidad de la voz. Finalmente, las diferencias en el entorno sociolingüístico, el dialecto, el tamaño del tracto vocal y la forma pueden contribuir a las *variabilidades entre locutores*.

La Figura 1.4 muestra los componentes principales de un sistema típico de reconocimiento de voz. La señal de voz digitalizada se transforma en primer lugar en un conjunto de medidas útiles o características a una tasa fija, típicamente una vez cada 10-20 mseg. Estas medidas se usan entonces para buscar la palabra candidata más parecida,

haciendo uso de especificaciones impuestas por los modelos acústico, léxico y de lenguaje. A través de este proceso, los datos de entrenamiento se usan para determinar los valores de los parámetros del modelo.

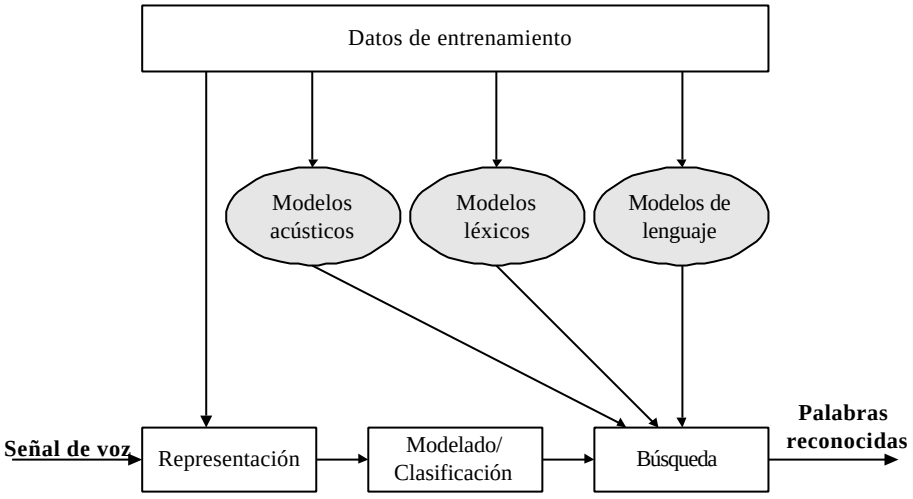


Figura 1.4: Componentes de un sistema típico de reconocimiento de voz

Los sistemas de reconocimiento de voz intentan modelar las fuentes de variabilidad descritas anteriormente de varias formas. En el nivel de representación de señales, las investigaciones han desarrollado representaciones que enfatizan las características de la señal perceptualmente importante con independencia del locutor, y desenfatizan las características dependientes del locutor. En el nivel de fonética acústica, la variabilidad de locutores se modela típicamente usando técnicas estadísticas aplicadas a grandes cantidades de datos. También se han desarrollado algoritmos de adaptación a locutores que adaptan modelos acústicos independientes del locutor a aquellos que el locutor actual use en el sistema.

La variabilidad a nivel de palabras se puede tratar permitiendo pronunciaciones alternativas de palabras de representaciones conocidas como *redes de pronunciación*. Las pronunciaciones alternativas comunes de palabras, así como los efectos del dialecto y el acento se tratan permitiendo algoritmos de búsqueda para encontrar caminos alternativos de fonemas a través de estas redes. Los modelos de lenguaje estadísticos,

basados en estimaciones de la frecuencia de ocurrencia de secuencias de palabras, se usan a menudo para guiar la búsqueda de la secuencia de palabras más probable.

El paradigma dominante del reconocimiento en los últimos quince años es conocido como Modelos Ocultos de Markov (*Hidden Markov Models*, HMM). Un HMM es un doble modelo estocástico, en el cual la generación de la cadena de fonemas base y la trama a trama, realizaciones acústicas firmes, se representan probabilísticamente como procesos de Markov. Las redes neuronales también se han empleado para estimar las tasas basadas en tramas; estas tasas se integran después en arquitecturas de sistemas basados en HMMs, que han llegado a denominarse *sistemas híbridos*.

Una característica interesante de los sistemas con tramas basadas en HMM es que los segmentos de voz se identifican durante el proceso de búsqueda, más que explícitamente. Una aproximación alternativa es identificar en primer lugar los segmentos de voz, después clasificar los segmentos y usar los segmentos obtenidos para reconocer palabras. Esta aproximación ha proporcionado un funcionamiento del reconocimiento competitivo en varias aplicaciones.

Capítulo 2: Reconocimiento de Locutores. Estado del Arte Actual

2.1: Introducción

En este capítulo nos proponemos realizar una revisión del estado del objeto de estudio en cuanto a metodología y modelos aplicables al Reconocimiento Automático de Locutores (Automatic Speaker Recognition, ASR). Pero, antes de comenzar esta revisión, hemos de hacer notar que, al haber manejado referencias bibliográficas de diversos grupos de investigación, nos hemos encontrado con diferentes terminologías y peculiaridades propios de cada lengua concreta, aunque se resaltan, en todos los casos, los fenómenos suprasegmentales, o prosódicos, que son universales. Por ello, en nuestro estudio se han tratado de verificar los datos aportados por los grupos de investigación extranjeros, pero aplicados a las características del lenguaje Español hablado.

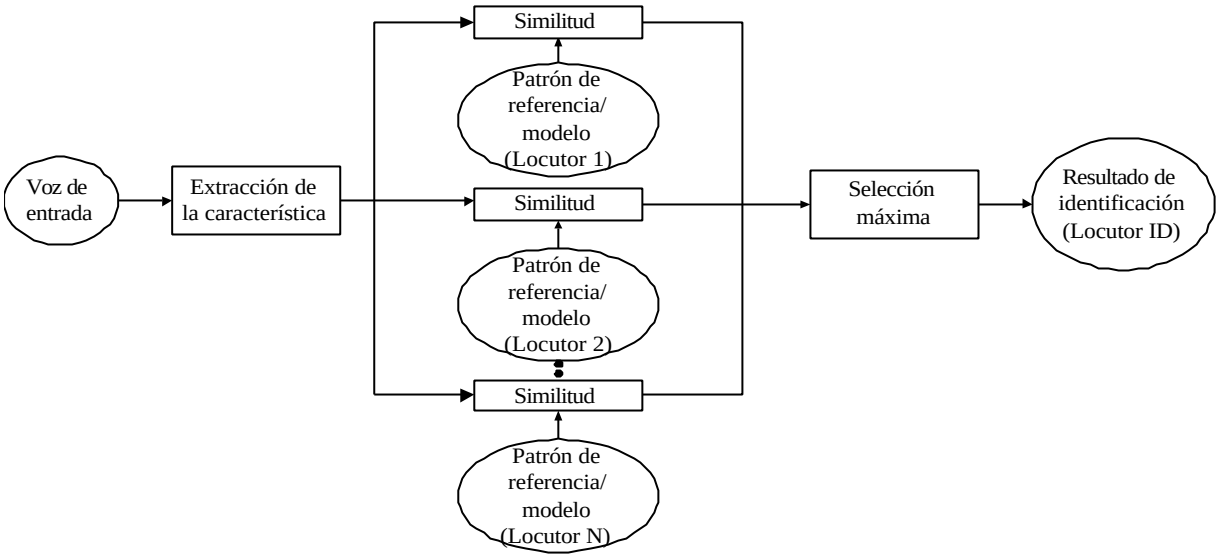
2.2: Principios del Reconocimiento de Locutores

Aunque los factores exactos de una señal de voz responsables de las características de los locutores no se conocen exactamente, es un hecho que los humanos sean capaces de distinguir entre locutores basándose en sus voces.

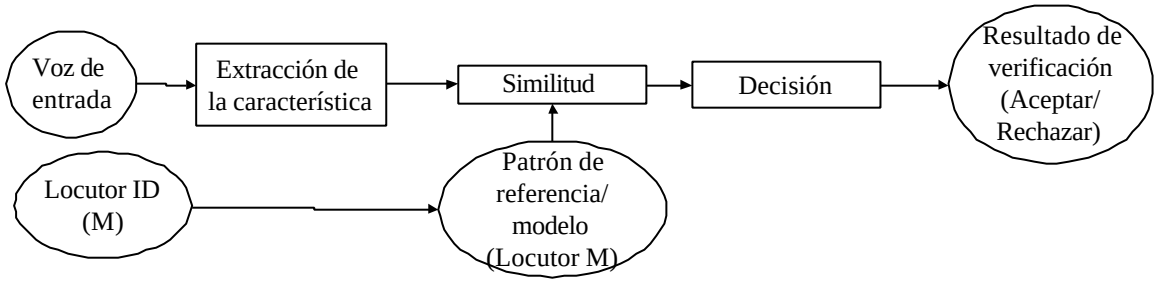
El reconocimiento de locutores, que puede ser clasificado en identificación y verificación, es el proceso de reconocer automáticamente quién está hablando basándose en información individual incluída en las ondas de la voz. Esta técnica hace posible usar la voz del locutor para verificar su identidad y controlar el acceso a servicios tales como tratamiento de voz (voice dailing), transacciones bancarias por teléfono, reconocimiento de voz para sistemas de buzón de voz inteligentes, compra telefónica, servicios de acceso a bases de datos, servicios de información, identificación de voz para acceso a edificios, correo vocal (voice mail), control de seguridad para áreas de información confidencial, y acceso remoto a ordenadores. En esta línea, se espera que la tecnología de reconocimiento de locutores cree nuevos servicios que harán nuestra vida diaria más cómoda. Otra aplicación importante de la tecnología de reconocimiento de locutores es para propósitos relativos a la medicina legal.

La Figura 2.1 muestra las estructuras básicas de los sistemas de identificación y verificación de locutores.

La identificación de locutores es el proceso de determinar qué locutor registrado proporciona un patrón de muestras dado. La verificación de locutores, por otro lado, es el proceso de aceptar o rechazar la petición de identidad de un locutor. La mayoría de las aplicaciones en las que se usa la voz como la clave para confirmar la identidad de un locutor se clasifican como verificación de locutores, o bien reconocimiento de locutores de conjunto cerrado, en el que se trata de identificar a un locutor particular como uno dentro de un conjunto finito de locutores de referencia.



(a) Identificación de locutores



(b) Verificación de locutores

Figura 2.1: Estructuras básicas de sistemas de reconocimiento de locutores

Existe también el caso llamado identificación de *conjunto abierto*, en el que un modelo de referencia de un locutor desconocido puede no existir. En este caso, un locutor puede ser identificado como perteneciente a un conjunto finito, o bien se puede desestimar como miembro del conjunto. Para la verificación de locutores, la afirmación de pertenencia de un locutor a un conjunto finito puede ser aceptada o rechazada. Este suele ser el caso en aplicaciones forenses. En esta situación, *el desconocido no se ajusta a ninguno de los modelos,* se requiere una decisión adicional alternativa. Tanto en el

proceso de identificación como en el de verificación, se puede emplear un umbral adicional de prueba para determinar si la coincidencia está lo suficientemente cerca para aceptar la decisión o bien si se necesitan más datos de voz.

Los métodos de reconocimiento de locutores también se pueden dividir en métodos de texto dependiente y de texto independiente. El primero requiere que el locutor pronuncie las palabras clave o frases que contengan el mismo texto para las pruebas de entrenamiento y reconocimiento (el locutor está restringido a lo que se debe decir), mientras que el último no cuenta con un texto hablado específico, ya que no se imponen restricciones.

Sin embargo, tanto el método de texto dependiente como el de texto independiente tienen un problema común. Estos sistemas pueden equivocarse fácilmente porque alguien que reproduzca la voz grabada de un locutor registrado diciendo las palabras o frases clave puede ser aceptado como el locutor registrado. Para vencer este problema, existen métodos en los que un pequeño conjunto de palabras, tales como dígitos, se usan como palabras clave y cada locutor es incitado a pronunciar una secuencia dada de palabras clave que se eligen aleatoriamente cada vez que se usa el sistema. Aun así, este método no es de confianza al completo, ya que puede equivocarse con equipamiento de grabación de electrónica avanzada que reproduzca las palabras clave en un orden solicitado. Por tanto, se ha propuesto recientemente un método de reconocimiento de locutores de texto seguro (machine-driven-text-dependent).

2.3: Parámetros Característicos

La identidad del locutor está correlada con las características fisiológicas y de comportamiento del locutor. Estas características existen tanto en la pendiente espectral (características del tracto vocal) como en las características suprasegmentales (características de la fuente de voz y características dinámicas que abarcan varios segmentos).

Las medidas del espectro a corto plazo más comunes que actualmente se usan son los coeficientes Cepstrum derivados de la Codificación Lineal Predictiva (*Linear Predictive Coding*, LPC) y sus coeficientes de regresión. La pendiente espectral reconstruida a partir de un conjunto truncado de coeficientes Cepstrum es mucho más suave que aquélla reconstruida a partir de los coeficientes LPC. Por tanto, proporciona una representación más estable de una repetición a otra de una declaración particular de un locutor. En cuanto a los coeficientes de regresión, típicamente los coeficientes de primer y segundo orden se extraen cada periodo de trama para representar la dinámica espectral. Estos coeficientes son derivados de las funciones temporales de los coeficientes Cepstrum y se denominan coeficientes delta y coeficientes delta-delta-Cepstrum, respectivamente.

El punto de partida para esta revisión es, por tanto, la característica convencional mayormente usada para el Reconocimiento Automático de Locutores: el Cepstrum[1].

2.3.1: Definición de la Cepstrum Compleja

Considérese una secuencia estable $x[n]$ cuya Transformada Z expresada en forma polar es:

$$X(z) = |X(z)|e^{jf(X(z))} \qquad (2.1)$$

donde $|X(z)|$ y $f(X(z))$ son la magnitud y el ángulo, respectivamente, del número complejo $X(z)$. Por ser $x[n]$ estable, la región de convergencia para $X(z)$ incluye el

[1] *Capítulo 12, "Discrete Time Signal Processing". Alan V. Oppenheim, Ronal W. Schafer. Ed. Prentice Hall [6].*

círculo unidad, y la Transformada de Fourier de $x[n]$ existe y es igual a $X(e^{jw})$. La Cepstrum Compleja que corresponde a $x[n]$ se define como una secuencia estable $\hat{x}[n]$ cuya transformada Z es

$$\hat{X}(z) = \log[X(z)] \qquad (2.2)$$

Aunque al definir la Cepstrum Compleja se puede usar cualquier base para el logaritmo, el logaritmo natural (de base e) es el típicamente empleado y asumirá a lo largo de toda la discusión. El logaritmo de una cantidad compleja $X(z)$ expresado en la forma de la ecuación (2.1) se define como

$$\log[X(z)] = \log\left[|X(z)|e^{jf(X(z))}\right] = \log|X(z)| + jf(X(z)) \qquad (2.3)$$

Como en la representación polar de un número complejo el ángulo es único sumando sólo múltiplos enteros de 2π, la parte imaginaria de la ecuación (2.3) aún no está bien definida. Resolveremos en breve este problema: por ahora asumimos que se ha elegido una definición apropiada.

La Cepstrum compleja existe si $\log[X(z)]$ tiene una representación en serie de potencias convergente de la forma

$$\hat{X}(z) = \log[X(z)] = \sum_{-\infty}^{\infty} \hat{x}[n]z^{-n}, \qquad |z| = 1 \qquad (2.4)$$

es decir, $\hat{X}(z) = \log[X(z)]$ debe cumplir todas las propiedades de la Transformada Z para una secuencia estable. Específicamente, la región de convergencia de la representación en serie de potencias de $\log[X(z)]$ debe ser de la forma

$$r_R \prec |z| \prec r_L \tag{2.5}$$

donde $0 \prec r_R \prec 1$ y $r_L \succ 1$. Si este es el caso, $\hat{x}[n]$, la secuencia de coeficientes de la serie de potencias, corresponde a la Cepstrum compleja de $x[n]$ y también viene dada por la integral de la Transformada inversa de Z

$$\hat{x}[n] = \frac{1}{2p}\oint_C \log[X(z)]z^{n-1}dz \tag{2.6}$$

donde el contorno de integración está dentro de la región de convergencia. Como se requiere que $\hat{x}[n]$ sea estable, la región de convergencia incluye el círculo unidad, y la Cepstrum compleja puede representarse usando la transformada de Fourier inversa como

$$\begin{aligned}\hat{x}[n] &= \frac{1}{2p}\int_{-p}^{p} \log\left[X\left(e^{jw}\right)\right]e^{jwn}dw \\ &= \frac{1}{2p}\int_{-p}^{p} \left[\log\left|X\left(e^{jw}\right)\right| + jf\left(X\left(e^{jw}\right)\right)\right]e^{jwn}dw\end{aligned} \tag{2.7}$$

El término Cepstrum compleja se distingue de la definición general de la Cepstrum. El empleo de la palabra compleja en este contexto implica que se usa el logaritmo complejo en su definición. Esto no implica que la Cepstrum compleja necesariamente sea una secuencia de números complejos. De hecho, como veremos más adelante, la definición que elegimos para el logaritmo complejo asegura que la Cepstrum compleja de una secuencia real será también una secuencia real.

En contraste con la Cepstrum compleja, la Cepstrum $c_x[n]$ de una señal (a veces referida como Cepstrum real) se define como la transformada de Fourier inversa del logaritmo de la magnitud de la transformada de Fourier, es decir

$$c_x[n] = \frac{1}{2p}\int_{-p}^{p} \log\left|X\left(e^{jw}\right)\right| e^{jwn}\, dw \qquad (2.8)$$

Dado que la magnitud de la Transformada de Fourier es real y no negativa, se implican consideraciones no especiales en la definición del logaritmo de la ecuación (2.8). Comparando las ecuaciones (2.7) y (2.8), observamos que $c_x[n]$ es la transformada inversa de la parte real de $\hat{X}\left(e^{jw}\right)$. Consecuentemente, $c_x[n]$ es igual a la parte conjugada simétrica de $\hat{x}[n]$, es decir

$$c_x[n] = \frac{\hat{x}[n] + \hat{x}^*[-n]}{2} \qquad (2.9)$$

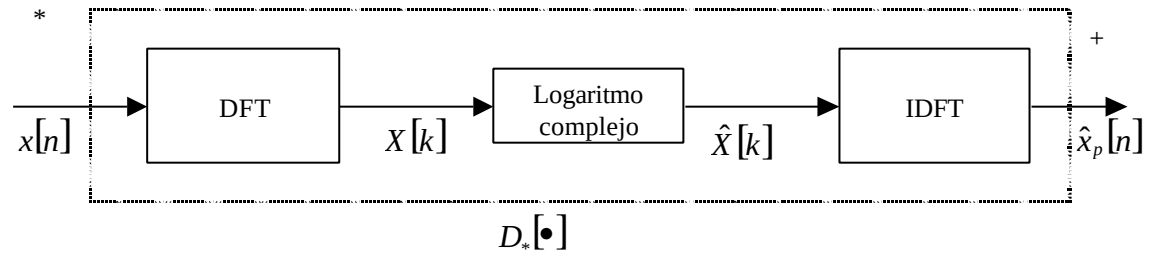
La Cepstrum es útil en muchas aplicaciones y, puesto que no depende de la fase de $X\left(e^{jw}\right)$, es mucho más fácil de calcular que la Cepstrum compleja. Sin embargo, dado que se basa únicamente en la magnitud de la transformada de Fourier, no es

invertible, es decir, $x[n]$ no puede ser recuperada a partir de $c_x[n]$. La Cepstrum compleja es, de algún modo, más difícil de calcular, pero es invertible.

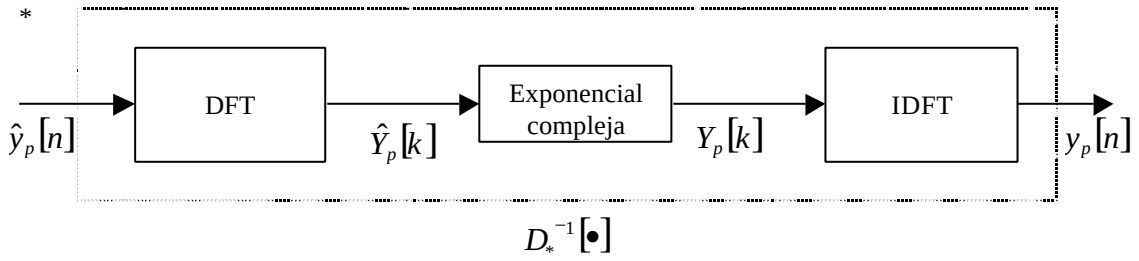
2.3.2: Cálculo de la Cepstrum usando la DFT

En la mayoría de las aplicaciones no dispondremos de expresiones analíticas de la señal y, por tanto, no podremos determinar analíticamente $\hat{x}[n]$ o $c_x[n]$. En este apartado, ilustramos el cálculo de la Cepstrum compleja y la Cepstrum usando la Transformada de Fourier Discreta (Discrete Fourier Transform, DFT).

Para calcular la Cepstrum compleja o la Cepstrum usando la DFT, como se muestra en la Figura 2.2, es necesario que la entrada sea de extensión finita. Por tanto, para el modelo de señal discutido en el punto anterior, $x[n]$ debe ser truncada.



(a)



(b)

Figura 12.9: Realización aproximada mediante el uso de la DFT de (a) el sistema característico para la convolución y (b) su inverso

Como sabemos, la convolución de una señal $v[n]$ con un tren de impulsos tal como $p[n]$ es el modelo para una señal que contiene eco. Dado que $x[n]$ es una convolución de $v[n]$ y $p[n]$, las ocasiones de eco a menudo no son fáciles de detectar al examinar $x[n]$. En el dominio Cepstrum, sin embargo, el efecto de $p[n]$ está presente como un tren de impulsos aditivos, y consecuentemente la presencia y localización de los ecos son a menudo más evidentes. Es por ello que la Cepstrum haya sido propuesta para ser usada como medio de detección de ecos. La misma idea ha sido empleada más tarde como base para detección del pitch vocal en señales de voz.

2.3.3: Aplicaciones de la Cepstrum al Procesado de Voz. Estimación de los Parámetros del Modelo de Voz

Motivados por la observación de que el sistema de audición humano percibe información basada en la energía en una banda de frecuencias más que en una sola frecuencia, los parámetros del dominio Cepstrum se usan comúnmente en sistemas de reconocimiento de voz e identificación de locutores. Los parámetros Cepstrum de una señal de voz x(n) se obtienen como la Transformada Discreta de Fourier Inversa (Inverse Discrete Fourier Transform, IDFT) de la salida de un conjunto de filtros de banda crítica cuya entrada es el logaritmo de la magnitud de la Transformada Discreta de Fourier (DFT) de $x(n)$, como se muestra en la Figura 2.3. A pesar de que los filtros de banda crítica son filtros pasa-todo para la extracción del pitch, por ejemplo, la superposición de bandas de filtros triangulares se usan para aplicaciones de reconocimiento de voz y locutores.

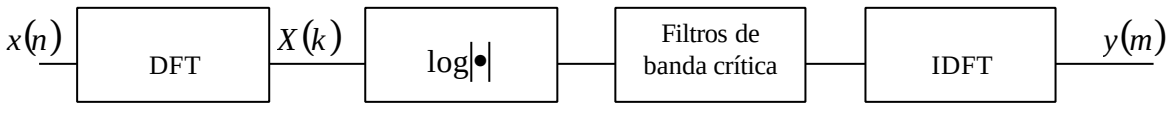


Figura 2.3: Generación de las características del dominio Cepstrum

Dirigiéndonos a este estudio, se debe elegir la primera frecuencia de interés del banco de filtros en 100 Hz aproximadamente, valor próximo a la frecuencia fundamental de la voz masculina. De esta forma, se siguen eligiendo espaciadamente bien frecuencias en una escala lineal, o bien frecuencias en la escala de frecuencias MEL hasta cubrir los 3500 Hz. Las salidas de los filtros centrados a las frecuencias elegidas se constituirán de la magnitud del espectro de cada trama de la señal de voz. Los filtros, si siguen la distribución lineal de las frecuencias centrales, tendrán un ancho de banda constante, mientras que aquellos para la escala de frecuencias MEL tendrán anchos de banda que se incrementan logarítmicamente con cada espectro de filtro que solapa el espectro del filtro adyacente. Los coeficientes Cepstrum se obtienen como la IDFT de la energía del logaritmo en los índices seleccionados. Los valores de la energía del logaritmo (previos a la operación de la IDFT) forman un segundo vector característico que se empleará para el proceso de discernir entre locutores[2].

Un problema común en los sistemas de identificación de locutores es que, en muchas aplicaciones, un canal de comunicaciones separa al usuario del sistema de reconocimiento. Los datos recogidos para entrenar los modelos de locutores son sólo representativos de una pequeña fracción de entre todas las condiciones acústicas que se pueden encontrar durante la prueba. Se intenta desgravar este problema proponiendo nuevas características que muestran una menor variación cuando la voz se corrompe por el ruido convolucional (canal) y/o ruido aditivo[3]. La característica convencional usada es el Cepstrum de predicción lineal (LP) descrita, que se deriva de una función de transferencia todo-polo que, a su vez, consigue una buena aproximación de la envolvente del espectro de la voz.

[2] *"A Comparison of Speaker Identification Results Using Features Based on Cepstrum and Fourier-Bessel Expansion". K. Gopalan, Timothy R. Anderson, Edward J. Cupples, Miembros del IEEE. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 3, Mayo, 1999 [1].*

[3] *"Speaker Identification Based on the Use of Robust Cepstral Features Obtained from Pole-Zero Transfer Functions". Mihailo S. Zilovic, Ravi P. Ramachandran, Richard J. Mammone, Miembros del IEEE. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 1, Enero, 1999 [3].*

Recientemente, se introdujo una nueva característica Cepstrum basada en una función de transferencia polo-cero (denominada componente adaptativa ponderada o Cepstrum ACW). Se han propuesto además cuatro nuevas características Cepstrum basadas en funciones de transferencia polo-cero:

- Una es un modo alternativo de hacer ponderación por componentes adaptativos y se denomina Cepstrum ACW2.
- Otras dos (conocidas como Cepstrum PFL1 y PFL2) se basan en un postfiltro polo-cero usado para mejora de la voz.
- Finalmente, un análisis autorregresivo de media variable (ARMA) de la voz da lugar a una función de transferencia polo-cero para describir la envolvente del espectro. La transformada Cepstrum de esta función de transferencia es la característica.

Con el objeto de comparar las diversas características, se han realizado experimentos incluyendo un sistema de identificación de locutores con un conjunto cerrado base de vectores cuantizados, e independencia de textos. Las características ACW y PFL1 son las preferidas, porque resultan iguales o mejores a la Cepstrum LP en todas las condiciones de prueba. El espectro correspondiente muestra un claro énfasis de los formantes y pendiente espectral nula. Ambas se basan en el principio de que para mejorar la robustez, es importante enfatizar los formantes. No se requiere una descripción detallada de la envolvente espectral. En el caso de la Cepstrum LP y el modelo ARMA, la envolvente del espectro cambia cuando la voz es corrompida por un canal y/o ruido. Sin embargo, los formantes, por sí mismos, permanecen intactos.

El estudio de estos métodos alternativos o complementarios de la Cepstrum LP se centró sobre el modo de ejecución de conjunto cerrado del sistema ASR (Automatic Speaker Recognition, Reconocimiento Automático de Locutores), y sobre el modo de reconocimiento independiente del texto.

El conjunto del sistema que consideraremos tendrá tres componentes:

1. Análisis de predicción lineal (LP) para la parametrización de la envolvente espectral.

2. Extracción de la característica para asegurar la discriminación del locutor.

3. Clasificación para tomar una decisión.

La entrada al sistema será una señal de voz posiblemente corrompida por ruido y posiblemente influenciada por otras condiciones del entorno (como efectos del canal) La salida será una decisión relativa a la identidad del locutor. Un sistema robusto realiza la tarea de reconocimiento con éxito incluso cuando la voz está corrompida por ruido y/o efectos del canal de comunicación. La situación ideal es conseguir un alto rendimiento en términos de precisión del reconocimiento dado cualquier tipo de material de voz. Se desarrollan, con este fin, características derivadas de robustos LP en modo de conjunto cerrado y texto independiente. Obsérvese que se usarán métodos existentes para la primera y tercera componentes del sistema.

Después de llevar a cabo el análisis LP de la voz, existen varias representaciones equivalentes de los parámetros LP. Una comparación de estos parámetros en términos de la precisión en el reconocimiento del locutor revela que la transformada Cepstrum LP es la mejor cuando entrenamiento y prueba se realizan sobre la voz limpia. El problema de la transformada Cepstrum LP es que un desajuste entre las condiciones de entrenamiento y prueba sacrifica un gran rendimiento en la ejecución, disminuyendo consecuentemente la robustez. La transformada Cepstrum LP se deriva de una función de transferencia polo-cero que describe la envolvente del espectro de la voz. Ésta en particular da información acerca de los formantes, lo que es crucial para que el reconocimiento del locutor tenga éxito. El intento de encontrar características más robustas es para transformar, en primer lugar, la función de transferencia todo-polo derivada del análisis LP en una función de transferencia polo-cero que da mayor énfasis a los formantes. La transformada Cepstrum de la función de transferencia polo-cero es la característica. Se formulan y comparan varias aproximaciones nuevas que convierten una función todo-polo en una polo-cero. Surge la cuestión del porqué un camino de dos pasos que va de la voz a una función de transferencia polo-cero. También se considera

un modelo polo-cero obtenido de un análisis directo autorregresivo de media variable (ARMA) de la voz como primer componente del sistema. Sin embargo, como se pone de manifiesto más tarde, el rendimiento obtenido de una aproximación ARMA es inferior al que se obtiene usando una función de transferencia polo-cero derivada del análisis LP.

2.3.3.1: Parametrización de la Envolvente del Espectro

El primer componente del sistema transforma la señal de voz en una representación compacta de la envolvente de su espectro. Para ello se usa un análisis predictivo lineal (LP). Un análisis LP de una señal de voz, basado en el modelo de que una muestra de voz es una combinación lineal ponderada de *p* muestras previas, proporciona un conjunto de coeficientes o pesos a_k. La ecuación fundamental que gobierna este modelo es

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + e(n) \qquad (2.10)$$

donde $s(n)$ es la señal de voz y $e(n)$ es el error de LP residual. Estos pesos corresponden a los coeficientes de la forma directa de un filtro no recursivo

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k} = \prod_{k=1}^{p} (1 - f_k z^{-1}) \qquad (2.11)$$

donde f_k para $1 \le k \le p$ representa los ceros de $A(z)$. Haciendo pasar la señal de voz a través del filtro $A(z)$, se obtiene el error LP residual $e(n)$ que está libre de

redundancia de muestras cercanas. La determinación de los coeficientes LP a_k se basa normalmente en la minimización del error cuadrático medio E_{mse} ponderado sobre un segmento de voz que conste de N muestras. En la minimización de E_{mse} usando la aproximación de la autocorrelación, los coeficientes a_k se calculan resolviendo un sistema de ecuaciones lineales. Por otra parte, se garantiza que $A(z)$ es de fase mínima. La magnitud del espectro de $1/A(z)$ describe la envolvente del espectro de la voz. Como $1/A(z)$ viene completamente determinada por sus polos f_k, el análisis LP se basa en un modelo todo-polo.

Un análisis ARMA conduce a una función de transferencia $U(z)/V(z)$ que aproxima la envolvente espectral. Usamos el método de Shank para determinar los coeficientes de $U(z)$ y $V(z)$. En esta aproximación, se determina en primer lugar una función de fase mínima $V(z)$ mediante un análisis LP, y es igual a $A(z)$. La respuesta al impulso de $1/V(z)$ es $h(n)$, que es truncado a N muestras puesto que el segmento de voz a analizar se compone de N muestras. El error es $s(n) - h(n) * u(n)$ donde $u(n)$ es la respuesta finita al impulso de $U(z)$. Sobre la minimización del error cuadrático medio, se hallan los coeficientes de $U(z)$ resolviendo un sistema de lineal de ecuaciones. Aunque no se garantiza que $U(z)$ sea de fase mínima, se puede forzar esta propiedad reflejando los ceros de $U(z)$ exteriores al círculo unidad para situarlos en el interior. El orden de $U(z)$ se determina empíricamente para obtener una aproximación aceptable de la envolvente espectral.

2.3.3.2: Extracción de la Característica

El primer componente proporciona tanto una función transferencia todo-polo como una polo-cero. El extractor de la característica generalmente realiza una transformación de la función y después computa la transformada Cepstrum como vector característico. Supóngase que una función de transferencia polo-cero $P(z)$ viene dada por

$$P(z) = \frac{U(z)}{V(z)} = \frac{\prod_{k=1}^{u}(1-u_k z^{-1})}{\prod_{k=1}^{u}(1-v_k z^{-1})} \qquad (2.12)$$

Si $P(z)$ es de fase mínima, la transformada Cepstrum $c_p(n)$ se puede obtener bien por recursión computacionalmente eficiente basada en los coeficientes polinómicos, o bien considerando las raíces polinómicas u_k y v_k como las dadas por

$$c_p(n) = \frac{1}{n}\sum_{k=1}^{v} v_k^n - \frac{1}{n}\sum_{k=1}^{u} u_k^n \qquad (2.13)$$

para $n \succ 0$.

La primera característica que consideramos es la transformada Cepstrum convencional LP del filtro todo-polo LP $1/A(z)$. Esto sirve como muestra con la que comparamos nuestras características propuestas. Para las siguientes cuatro características, la función de transferencia todo-polo LP $1/A(z)$ se transformará en una función polo-cero. Se sabe que la diferencia cuadrática media entre dos vectores Cepstrum está directamente relacionada con la diferencia cuadrática media en la magnitud del espectro de las funciones de transferencia de las que se derivaron los vectores Cepstrum. La magnitud del espectro de $1/A(z)$ obtenida de voz limpia y corrompida muestra un grado de desemejanza incluso alrededor de las regiones de los formantes. Esto se manifiesta como una clara diferencia en los vectores Cepstrum, que causa una degradación en el funcionamiento. Nuestro objetivo es transformar la función de transferencia todo-polo en una función de transferencia polo-cero tal que la diferencia en la magnitud del espectro decrezca cuando se suma ruido a la voz y/o la

voz pasa a través de un canal. Usamos una aproximación recientemente introducida con propósito comparativo y formulamos tres nuevas aproximaciones.

La aproximación existente es realizar en primer lugar una expansión de fracciones parciales de $1/A(z)$ para obtener

$$\frac{1}{A(z)} = \sum_{k=1}^{p} \frac{\lim\limits_{z \to f_k} [(1 - f_k z^{-1}) / A(z)]}{1 - f_k z^{-1}} = \sum_{k=1}^{p} \frac{r_k}{1 - f_k z^{-1}} \qquad (2.14)$$

Los experimentos realizados revelan que los residuos r_k muestran variaciones considerables especialmente para los polos no formantes cuando la voz está degradada. Por tanto, las variaciones en r_k se eliminaron forzando $r_k = 1$ para cualquier k. Con esto, la función de transferencia es del tipo polo-cero de la forma

$$\frac{N(z)}{A(z)} = \sum_{k=1}^{p} \frac{1}{1 - f_k z^{-1}} = \frac{1}{A(z)} \sum_{k=1}^{p} \prod_{i=1 \neq k}^{p} (1 - f_i z^{-1}) = p \frac{1 - \sum_{k=1}^{p-1} b_k z^{-k}}{1 - \sum_{k=1}^{p} a_k z^{-k}} \qquad (2.15)$$

Está demostrado que $N(z)$ es la derivada de $A(z)$ con respecto a z y por tanto, los coeficientes b_k se obtienen fácilmente a partir de los a_k como $b_k = (p-k)a_k / p$ para $k = 1$ hasta $p-1$. El desajuste en la magnitud del espectro de $N(z)/A(z)$ para voz limpia y corrompida se reduce aun más que el de $1/A(z)$. Se garantiza que el numerador polinómico $N(z)$ es de fase mínima. La transformada Cepstrum de $N(z)/A(z)$ se usa como el vector característico y se puede obtener mediante una

recursión eficiente basada en los coeficientes polinómicos. Este método se conoce como *Ponderación de componentes adaptativa (Adaptive Component Weighting*, ACW) y se usa primordialmente para mitigar los efectos del canal.

La primera nueva aproximación es una alternativa al método ACW. Desde la perspectiva del análisis del sistema, el filtro LP $1/A(z)$ se puede ver como la conexión en cascada de p filtros de primer orden con función de transferencia $1/(1-f_k z^{-1})$. Conectar estas p secciones de primer orden en paralelo da como resultado la función de transferencia polo-cero del conjunto para el método ACW [ver (2.15)]. Usando un razonamiento similar, $1/A(z)$ se puede interpretar como una conexión en cascada de secciones de segundo orden (parejas de secciones de primer orden). La combinación en paralelo de estas $p/2$ secciones de segundo orden da origen a otra función de transferencia polo-cero del conjunto. Nos referimos a ésta como a la aproximación ACW2. Para la conexión en cascada inicial, la surge la cuestión de qué secciones de primer orden emparejar. Se elige emparejar las secciones de primer orden especificadas por los polos complejos conjugados de $1/A(z)$. Los polos reales restantes también se emparejan. Supóngase que entre los p polos f_k, hay c polos complejos y $p-c$ polos reales. Los polos complejos conjugados se componen como $f_1,\ f_1^*,\ f_2, f_2^*, \ldots,\ f_{c/2},\ f_{c/2}^*$ donde f_k es el complejo conjugado de f_k^*. Los restantes polos reales se disponen como $f_{c+1},\ f_{c+2},\ \ldots,\ f_{p-1},\ f_p$. En este caso, la función de transferencia polo-cero viene dada por

$$\frac{N(z)}{A(z)} = \sum_{k=1}^{c/2} \frac{1}{(1-f_k z^{-1})(1-f_k^* z^{-1})} + \\ + \sum_{k=0}^{(p-c)/2-1} \frac{1}{(1-f_{c+2k+1} z^{-1})(1-f_{c+2(k+1)} z^{-1})} \qquad (2.16)$$

En la práctica, se observa que si hay polos reales, únicamente hay dos de ellos para el caso en que $p = 12$ asumiendo voz muestreada a 8 KHz. Por tanto, el emparejado óptimo de dos polos reales no es una distribución práctica. El motivo de emparejar pares complejos conjugados se basa en el hecho de que la respuesta al impulso de una sección de segundo orden especificada por un par de polos complejos conjugados es una senoide amortiguada. Esto proporciona un modelo polo-cero más natural de la señal de voz, representándola como una superposición de senoides moduladas en amplitud. Se conjetura que $N(z)$ es de fase mínima porque en la práctica no se halló instancia alguna de una $N(z)$ de fase no mínima. En un sistema real, cualesquiera raíces de $N(z)$ exteriores al círculo unidad se deberían reflejar en su interior. De nuevo, la transformada Cepstrum de $N(z)/A(z)$ se usa como el vector característico.

La otra familia de funciones de transferencia polo-cero que se formula se basa en el concepto de un postfiltro que se introdujo para mejorar la voz ruidosa. La filosofía de desarrollar un postfiltro revela el hecho de que se puede tolerar perceptualmente más ruido en las regiones de los formantes (picos espectrales) que en los valles del espectro. El postfiltro se obtiene de $A(z)$ y su función de transferencia viene dada por

$$H_{pf}(z) = \frac{A(z/b)}{A(z/a)} \qquad 0 \prec b \prec a \leq 1 \tag{2.17}$$

El espectro de $H_{pf}(z)$ enfatiza los picos formantes. La envolvente espectral de la voz tras el postfiltro se determina como la magnitud de la respuesta de

$$T(z) = \frac{A(z/b)}{A(z)A(z/a)} \tag{2.18}$$

Si $A(z)$ es de fase mínima, se garantiza que tanto $H_{pf}(z)$ como $T(z)$ son de fase mínima. La transformada Cepstrum de ambas funciones de transferencia polo-cero se usan como vectores característicos. Se puede mostrar la transformada Cepstrum de $H_{pf}(z)$ equivalente a la del Cepstrum LP ponderada por un factor $a^n - b^n$. Se puede mostrar la transformada Cepstrum de $T(z)$ equivalente a la Cepstrum LP ponderada por un factor $1 + a^n - b^n$. Se han considerado otros modos distintos de ponderar la Cepstrum LP (como ponderación de frecuencia, ponderación de variación inversa y ponderación paso de banda) en diversos estudios. Las ponderaciones que propuestas tienen una interpretación en términos de las funciones de transferencia. También, se desenfatizan los coeficientes Cepstrum de índices más bajos.

2.3.3.3: Clasificador Cuantizador de Vectores

Un clasificador cuantizador de vectores (VQ) se usa para traducir una decisión a la identidad del locutor. Obsérvese que no nos restringimos a este tipo de clasificador para las características que proponemos. Un clasificador VQ se usa porque se sabe que funciona muy bien y hará nuestros resultados extremadamente fiables. El sistema se muestra en la Figura 4. Para cada locutor, se usa un conjunto de vectores característicos de entrenamiento para diseñar un libro de código basado en el algoritmo de Linde-Buzo-Gray (LBG), algoritmo que describiremos en el siguiente capítulo. Habrá M libros de código, cada uno perteneciente a cada uno de los M locutores.

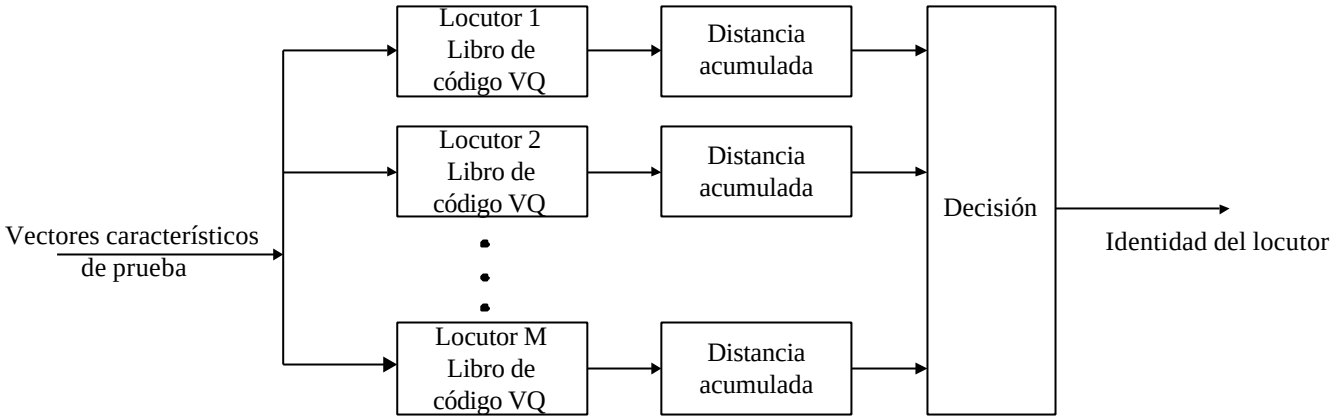


Figura 4: Diagrama de bloques de un sistema de identificación de locutores basado en la VQ

Para probar el sistema, se convierte una muestra (ejemplar de prueba) de uno de los M locutores en un conjunto de vectores característicos de prueba. Esto se cuantiza mediante cada uno de los M libros de código. El vector cuantizado es aquél que, de acuerdo con cierta distancia de medida, está más próximo al vector característico de prueba. Usamos la raíz de la distancia Euclídea como medida. Por tanto, se graban M distancias diferentes, una por cada libro de código. Este proceso se repite para cada vector característico de prueba. Las distancias se acumulan sobre el conjunto completo de vectores característicos. El libro de código que muestre la menor distancia acumulada identifica al locutor. Cuando se prueban muchas muestras, la tasa de éxito es el número de muestras para las que el locutor ha sido correctamente identificado dividido por el número total de muestras testeadas.

Los libros de código VQ se entrenarán para una condición particular, a saber, para la voz limpia. Se usarán diferentes condiciones de prueba correspondientes a voz limpia y corrompida para proporcionar una evaluación definitiva y cuantitativa de robustez. Si una característica es robusta, un desajuste entre las condiciones de prueba y entrenamiento debería provocar una ligera degradación en el funcionamiento o tasa de éxito.

De la forma mostrada, se examinan diversas características nuevas Cepstrum, basadas en funciones de transferencia polo-cero, en lo que se refiere a la robustez frente

al ruido y los efectos del canal. La referencia es la Cepstrum LP convencional basada en la función de transferencia LP todo-polo. Esta función todo-polo se convierte por distintos medios en funciones de transferencia polo-cero, de las cuales se obtiene la característica Cepstrum. Dos de las funciones polo-cero, en concreto, la ACW y la ACW2 se basan en una expansión parcial de fracciones de la función todo-polo LP. Una posterior normalización de los residuos es la clave para elevar la robustez. El espectro ACW enfatiza los formantes. Otras dos funciones polo-cero (PFL1 y PFL2) se basan en el concepto de postfiltro que inicialmente fue configurado para mejora de la voz. Las Cepstrum PFL1 y PFL2 son equivalentes a aplicar una ponderación a la Cepstrum LP convencional. Al igual que el espectro ACW, el espectro PFL1 enfatiza los formantes. Otro método de obtención de función de transferencia polo-cero es considerar un análisis ARMA de la voz.

A la vista de los experimentos realizados sobre voz limpia, voz ruidosa y voz sujeta a efectos del canal, la Cepstrum ACW y la Cepstrum PFL1 son las características preferidas. Tanto el espectro ACW como el PFL1 muestran características similares en que los formantes están enfatizados y no hay pendiente espectral. Esto implica que para identificación robusta de locutores, los formantes son extremadamente importantes. Por otra parte, una precisa representación de la envolvente espectral completa tanto por análisis LP como por análisis ARMA no es el mejor modo de proporcionar robustez. La envolvente del espectro del conjunto cambia cuando la voz es corrompida por un canal y/o ruido. Sin embargo, los formantes por sí mismos, permanecen más intactos.

2.3.3.4: Otros Parámetros Característicos para Identificación Robusta de Locutores sobre Canales Telefónicos

El funcionamiento de los sistemas actuales de identificación de locutores puede también empeorar significativamente bajo ciertas condiciones acústicas, tales como líneas telefónicas ruidosas. El método que se plantea se centra en los efectos de los desajustes acústicos entre los datos de entrenamiento y prueba, y se concentra en dos aproximaciones

1) extracción de características que son robustas frente a las variaciones del canal, y
2) transformación de los modelos de locutores para compensar los efectos del canal.

En primer lugar, un estudio experimental muestra que la optimización del procesado *front-end* de la señal de voz puede mejorar significativamente el funcionamiento del reconocimiento de locutores. Se introduce un nuevo diseño de banco de filtros para mejorar la robustez del cómputo del espectro de voz en la unidad de *front-end*. A continuación, se describe una nueva característica basada en las pendientes espectrales. Se demuestra que su habilidad para discriminar entre locutores es superior a la de la Cepstrum tradicional.

Asimismo, se presentan dos modelos de métodos de transformación que reducen más los efectos del canal. Estos métodos hacen uso de bases de datos estéreo recogidas localmente para estimar una transformación con varianza independiente del locutor para cada una de las características de la voz usadas por el clasificador. Las transformaciones construidas sobre esta base de datos estéreo pueden ser aplicadas entonces a modelos de locutores derivados de otras bases de datos[4].

Mostramos en primer lugar experimentalmente que el funcionamiento del reconocimiento de locutores depende fuertemente de la unidad *front-end* que preprocesa la señal de voz. Demostramos que se puede optimizar el *front-end* para mejorar consistente y significativamente el funcionamiento del sistema. También describimos un nuevo diseño de banco de filtros que mejora la robustez del cálculo del espectro de la voz. Después derivamos una nueva característica basada en las pendientes espectrales que se puede usar bien individualmente, o bien en combinación con la Cepstrum MEL.

[4] *"Robust Text-Independent Speaker Identification over Telephone Channels". Hema A. Murthy, Françoise Beaufays, Larry P. Heck, Miembros del IEEE; Mitchel Weintraub. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 5, Septiembre, 1999 [4].*

Los experimentos sobre los que se basó el estudio de este método tienen que ver con reconocimiento de locutores de conjunto cerrado, es decir, con el problema de identificar a un locutor entre un conjunto conocido de locutores. No obstante, mostramos que los métodos desarrollados para identificación de conjunto cerrado se extienden al reconocimiento de locutores de conjunto abierto, al problema de determinar si un locutor de prueba pertenece al conjunto de entrenamiento, y de identificarlo si es así.

Para determinar que los algoritmos desarrollados son generales y trabajan sobre bases de datos telefónicas distintas, se usaron varias bases de datos en este estudio. Estas bases de datos muestran diferentes condiciones de desajuste, contenían diferentes cantidades de datos de entrenamiento y prueba, que se recogieron de diferentes instituciones, e ilustran diferentes tipos de voz, desde dígitos leídos hasta voz conversacional sin restricciones.

2.3.3.4.1: Descripción del Sistema de Referencia

El sistema de reconocimiento de locutores usado como referencia para este caso consiste en una serie de modelos Gaussianos mixtos (Gaussian Mixture Models, GMM's) que modelan las voces de los locutores a identificar, junto con un clasificador que evalúa la similitud de segmentos de voz desconocida con respecto a estos modelos. En problemas de conjunto cerrado (identificación de locutores), el clasificador genera la hipótesis de la identidad de un locutor de prueba desconocido determinando qué modelo maximiza la similitud de la muestra de prueba. En problemas de conjunto abierto, el clasificador compara los grados/marcas de similitud con algún umbral para desestimar los segmentos de prueba que apenas coinciden con todos los modelos de entrenamiento, y en otro caso genera la hipótesis de las identidades del locutor basándose en la maximización de la similitud.

Tanto en entrenamiento como en prueba, las formas de onda de la voz son digitalizadas y preprocesadas por una unidad de *front-end* que extrae un conjunto de N_c coeficientes Cepstrum de frecuencias MEL de cada trama de datos. Los parámetros que

controlan el procesado *front-end* (por ejemplo, rango de frecuencia, corte del filtro y resolución del filtro) se fijan 'a priori', pero se pueden modificar si se desea. La eliminación de la media Cepstrum se puede aplicar (opcionalmente) a cada muestra para eliminar parte del solapamiento espectral que se produce en el canal de comunicación.

Para cada locutor, se construye un GMM usando los datos de entrenamiento del locutor para estimar las principales probabilidades, medias y varianzas de las Gaussianas. El número de Gaussianas, N_g, es el mismo para cada locutor y se elige dependiendo de la cantidad y naturaleza de los datos de entrenamiento y prueba. La generación de un GMM comienza con la elección de un libro de código de cuantización de vectores (Vector Quantization, VQ) aleatorio. Después el libro de código se continúa iterando, usando el algoritmo de Lloyd. En cada iteración VQ, las Gaussianas que contengan el mayor número de puntos dato se dividen, y las que contengan menos puntos dato, se eliminan. Tras la convergencia del libro de código VQ, las Gaussianas se ajustan/adecúan a las palabras de código, y sus parámetros se ajustan usando unas cuantas iteraciones de Expectación-Maximización (Expectation-Maximization, EM).

Cuando se le presenta un segmento de voz de un locutor desconocido, el clasificador chequea todas las tramas del segmento con energía suficientemente alta con todos los modelos de locutores, acumulando las similitudes logarítmicas de las tramas de voz para cada modelo. Entonces se toma una difícil decisión, tal como qué modelo acumuló la similitud logarítmica más alta, y este modelo se presenta hipotéticamente como perteneciente al locutor del que muestreamos el segmento de prueba.

2.3.3.4.2: Experimentos Preliminares

Los experimentos preliminares se realizaron con el sistema de referencia para medir el efecto de los desajustes del canal y la sustracción de la media Cepstrum sobre las tasas de error en el reconocimiento de locutores. Estos experimentos se realizan con un vector característicco Cepstrum MEL.

Todas las investigaciones han proporcionado resultados similares: la CMS elimina los efectos convolucionales pero no elimina el ruido aditivo, y no tiene en cuenta las no linealidades y no estacionariedad del canal.

Además, la CMS puede eliminar algunas de las características del locutor. Este resultado, que se puede atribuir a la cancelación de la información del tracto vocal por el filtrado paso de alta del CMS, es el esperado también en técnicas tales como el preprocesado RASTA.

Finalmente, se confirma que, más que la presencia de una unidad telefónica entre el locutor y el sistema ASR, es el posible desajuste entre las condiciones de entrenamiento y prueba lo que da lugar a un funcionamiento pobre en la identificación de locutores.

En otra serie de experimentos se midió la distorsión promedio entre los coeficientes Cepstrum grabados mediante un micrófono y sus grabaciones estéreo telefónicas. La medida de distorsión para los coeficientes Cepstrum k, d_k se define como

$$d_k = \frac{\left\langle \left(c_k^S - c_k^T\right)^2 \right\rangle}{\sigma_k^S \sigma_k^T} \qquad (2.19)$$

donde c_k^S y c_k^T denotan, respectivamente, los coeficientes Cepstrum k-ésimos de una trama de datos grabados con un micrófono y de su grabación estéreo telefónica; y σ_k^S y σ_k^T denotan la desviación estándar de los coeficientes Cepstrum con micrófono y telefónicos. El promedio $\langle \cdot \rangle$ se realiza sobre todas las unidades telefónicas y todos los locutores de la base de datos, y se estima de todas las tramas de varias frases de cada combinación telefónica del locutor. Si se analiza la distorsión promedio d_k frente al

índice *k* del coeficiente Cepstrum, con y sin sustracción de la media Cepstrum, se observa de nuevo que la sustracción de la media Cepstrum colabora en el decrecimiento de los efectos del canal, aunque la distorsión permanece significativamente después de la CMS. Además, los efectos del canal son más notables para los coeficientes Cepstrum de orden más alto. Esto puede deberse al hecho de que las energías totales de la voz para estos coeficientes son más bajas que aquellas para los coeficientes Cepstrum de orden más bajo, y que los efectos del ruido son, por tanto, relativamente más importantes.

Debido a que este trabajo se centró en el reconocimiento de locutores bajo condiciones de desajuste, la CMS se aplicó sistemáticamente en todos los casos (a menos que se especificara en otro caso) como un primer paso para eliminar los efectos del canal.

2.3.3.4.3: Extracción de la Característica para la Identificación del Locutor

En este apartado, mostramos que el funcionamiento de un sistema ASR depende fuertemente de los parámetros que describen la unidad *front-end* que procesa la voz de entrada. Para hacer más robusto el *front-end* sobre un amplio rango de parámetros, redefinimos el banco de filtros basándonos en la característica Cepstrum que se calcula. Demostramos a partir de aquí, que se puede obtener una gran mejora del funcionamiento al optimizar los parámetros del *front-end*.

Introducimos de nuevo la característica estudiada anteriormente: la pendiente espectral. Mostramos que la pendiente espectral (tras la reoptimización de los parámetros del *front-end*) discrimina mejor entre locutores que el Cepstrum. Además, discutimos que las dos características contienen información relativamente ortogonal, y mostramos que combinándolas mejoramos más el funcionamiento del sistema.

2.3.3.4.3.1: Descripción del Front-End Básico

El *front-end* básico usado en este trabajo transforma en primer lugar la señal de voz al dominio frecuencial a través de la Transformada Rápida de Fourier (Fast Fourier

Transform, FFT). La escala de frecuencias se transforma entonces de acuerdo con la escala MEL, para dar mayor resolución a bajas frecuencias y una resolución menor a altas frecuencias. Específicamente, implementamos la transformación bilineal propuesta por Acero

$$w_{new} = w + 2atan\frac{F_w sinw}{1 - F_w \cos w} \qquad (2.20)$$

donde la constante $F_w \in [0,1]$ controla la cantidad de desplazamiento o *warping*. La escala de frecuencias se multiplica entonces por un banco de N_f filtros, cuyas frecuencias centrales están uniformemente distribuídas en el intervalo $[Min_f, Max_f]$, a lo largo del eje de frecuencia transformado. El ancho de cada uno de estos filtros se extiende desde la frecuencia central del filtro previo a la frecuencia central del siguiente filtro. La caída del filtro es trapezoidal, y puede variar en todos los casos de triangular a rectangular. La caída de un conjunto particular de filtros se caracteriza por una constante T_r que mide la relación entre los lados menor y mayor del trapezoide ($T_r = 0$ significa filtros triangulares, $T_r = 1$ indica filtros rectangulares). Las energías del banco de filtros se calculan después integrando la energía de cada filtro, y se usa la Transformada Discreta del Coseno (Discrete Cosine Transform, DCT) para transformar las energías logarítmicas del banco de filtros en coeficientes Cepstrum. La sustracción de la media del Cepstrum se aplica a cada muestra de entrenamiento y prueba.

2.3.3.4.3.2: Diseño Perceptualmente Motivado del Banco de Filtros

Debido a que los anchos de banda de los filtros desplazados MEL se eligen en base al número de filtros, la estimación de la energía del banco de filtros puede ser pobre a ciertas frecuencias cuando la escala de frecuencias está desplazada,

especialmente si el número de filtros es grande. Idealmente, uno debería muestrear no uniformemente la transformada de Fourier durante el cómputo de la FFT para compensar la escala de frecuencia transformada. Alternativamente, uno puede hacer del ancho de banda de cada filtro una función de la frecuencia (en contraposición con una función de la distancia entre frecuencias centrales de filtros adyacentes como en el diseño de referencia). La cuestión es cómo se debería elegir esta función.

Frecuencia Central (f_c) (Hz)	Ancho de Banda (Bw)
$f_c \prec 1000$	137.5
$1000 \leq f_c \leq 2000$	$1.11(f_c)^{0.7}$
$f_c \succ 2000$	$10.84(f_c)^{0.4}$

Tabla 2.1: Anchos de banda de los filtros del Front-End como función de las frecuencias centrales de los filtros.

Klatt propuso un banco de 30 filtros basados en criterios de percepción auditiva. Debido a que los parámetros de Klatt se derivan de un tamaño fijo de banco de filtros, su diseño no pudo ser trasladado directamente a nuestro sistema. En su lugar, aproximamos los coeficientes de Klatt por la función dada en la Tabla 2.1. Para proporcionar mayor flexibilidad al sistema, se introdujo en el *front-end* un parámetro adicional, la escala del ancho de banda, B_s, para escalar uniformemente los anchos de banda de los filtros, es decir,

$$Bw_{new} = B_s Bw \qquad (2.21)$$

donde Bw es el ancho de banda especificado en la Tabla 2.1.

2.3.3.4.3.3: Optimización de los Parámetros del Front-End para la Cepstrum Basada en Bancos de Filtros

En el cómputo de la Cepstrum, afecta un gran número de parámetros, que son:

N_c, número de coeficientes Cepstrum

N_f, número de filtros del banco de filtros

$Max_f - Min_f$, ancho de banda efectivo de la voz

F_w, constante de frecuencia de desplazamiento: -1.0 a +1.0

T_r, caída de los filtros: 1.0 (rectangular), 0.0-1.0 (trapezoidal), 0.0 (triangular)

B_s, factor de escala del ancho de banda: 0.0-1.0

Desde el punto de vista del procesamiento de señales, N_c define la resolución que se requiere en el dominio Cepstrum, N_f define la resolución que se requiere en el dominio frecuencial, Min_f y Max_f definen el ancho de banda efectivo de la voz, F_ω define la resolución a diferentes frecuencias, y T_r define la caída o transición de los filtros. Pero, ¿cómo afectan estos parámetros al funcionamiento del clasificador?

Una serie de experimentos preliminares mostraron que los valores de la mayoría de los parámetros del *front-end* tuvieron un gran impacto sobre el funcionamiento del clasificador. Como no está claro cómo afecta cada parámetro individual del *front-end* a la tasa de error del clasificador, desarrollamos una optimización extensiva para todos los

parámetros del *front-end*, usando una base de datos concreta. Para asegurar que los parámetros optimizados no eran específicos para la base de datos usada, reunimos los conjuntos de parámetros que daban el mejor funcionamiento sobre aquélla, y los probamos sobre otra serie de bases de datos. Los parámetros del *front-end* que daban como resultado el mejor funcionamiento para todas las bases de datos fueron después elegidos como el nuevo *front-end* para el sistema de identificación de locutores que usa esa característica.

El método de optimización que se usó se basa en sucesivas búsquedas lineales. En cada iteración, todos los parámetros excepto uno se mantienen constantes, y se evalúa la tasa de error de clasificación de locutores para diferentes valores del parámetro restante (búsqueda en línea). El valor que conduce a la tasa de error más baja se conserva. Entonces se fija el parámetro optimizado, y se optimiza el siguiente parámetro. El procedimiento continúa hasta que la tasa de error alcanza un mínimo local. Los parámetros del *front-end* iniciaban normalmente el proceso de optimización, que resultaba de optimizaciones parciales hechas previamente en nuestro laboratorio. Estos parámetros iniciales fueron aquellos que se usaron en el sistema de referencia.

Es probable que, para órdenes bajos de los coeficientes Cepstrum, la información de los locutores predomine en la representación pero, conforme el número de coeficientes Cepstrum aumenta, la información del canal comienza a dominar.

Como el funcionamiento del sistema varía significativamente con la elección de los parámetros del *front-end*, la siguiente prueba que se debe realizar es determinar si esta ganancia del funcionamiento se mantiene para diferentes bases de datos. El sistema que dio como resultado el mejor funcionamiento promedio fue elegido como el nuevo *front-end* para el sistema de identificación de locutores usando la Cepstrum como característica.

2.3.3.4.3.4: Pendiente Espectral en Frecuencia basada en Bancos de Filtros

Discutíamos en un apartado anterior cómo computábamos la Cepstrum basada en bancos de filtros. La información contenida en la Cepstrum corresponde a la caída de

conjunto del espectro. Es probable que predomine el primer formante, porque el primer formante tiene la energía más alta, debido al efecto de la caída glotal. Es bien sabido que los formantes y sus transiciones son muy importantes para la percepción de la voz. En estudios psicológicos desarrollados por Klatt, se observó que, cuando varía la posición de los formantes, los sonidos percibidos por los oyentes son diferentes de los que se pretendían. El mismo estudio muestra que los humanos perciben el mismo sonido cuando las amplitudes relativas de los formantes se modifican en diferentes instantes del sonido.

Aunque se han desarrollado varios algoritmos para estimar las posiciones de las frecuencias formantes en la voz contínua, el problema de la extracción de los formantes no es trivial. Las máquinas tienden a cometer graves errores al estimar la localización de los formantes: se introducen los picos espúreos y los verdaderos a menudo se omiten. Por ello, buscamos una nueva medida que enfatizara la localización de los formantes sin estimarlos en realidad.

La pendiente espectral basada en bancos de filtros es una métrica que puede hacer esto. Al comparar las pendientes de dos espectros del mismo sonido, no se anotan las diferencias de amplitud, mientras que sí se capturan las situaciones y anchos de banda de las resonancias. La pendiente espectral se puede relacionar también con la caída del pulso glotal. Si asumimos un modelo del sistema fuente para la producción de la voz, el espectro correspondiente al sistema discurre por la parte superior del espectro correspondiente a la fuente. Incluso en el caso de que las situaciones de los máximos/picos sean las mismas para diferentes locutores, la información de la pendiente puede proporcionar a su vez información acerca de la inclinación (tilt) en el espectro, tilt espectral que se relaciona con la caída del pulso glotal.

Klatt sugirió una medida de pendiente espectral que usaron Hanson y Wakita para el reconocimiento de palabras aisladas. En el estudio posterior, la pendiente se calcula indirectamente, usando la relación entre la derivada del espectro y la Cepstrum ponderada. Este principio se puede aplicar a la Cepstrum basada en bancos de filtros únicamente cuando el número de filtros es infinito (y no solapados) y el número de

coeficientes Cepstrum es infinito. Proponemos, por tanto, una técnica basada en la métrica sugerida por Klatt, pero en la que las pendientes se computan de forma diferente.

Como con la característica Cepstrum, la señal de voz se transforma al dominio de la frecuencia mediante la FFT, y la escala de frecuencia se transforma. Se multiplica entonces el espectro por un banco de filtros similar al que se usaba para la Cepstrum de referencia. En una primera implementación, la pendiente espectral se calculó como la diferencia entre las energías logarítmicas de dos filtros consecutivos.

Este sistema se mejoró después de tres formas. En primer lugar, se reemplazó el banco de filtros original por el banco de filtros perceptual del apartado anterior. Después, se introdujo la CMS en el cálculo de la pendiente del banco de filtros para reducir los efectos del canal. Esto se hizo tomando la DCT de las energías logarítmicas del banco de filtros, eliminando el primer coeficiente Cepstrum, C_0, y calculando la DCT inversa de los restantes coeficientes Cepstrum. Las pendientes espectrales se determinaron entonces a partir del banco de filtros transformado. Por último, el cálculo de la pendiente se hizo más robusto frente a pequeñas variaciones en las energías logarítmicas del banco de filtros usando una técnica de regresión de tres puntos (la media cuadrática más baja se ajusta a una línea recta), en lugar de una simple diferencia entre filtros adyacentes. Este nuevo sistema se optimizó como se detalla debajo.

2.3.3.4.3.5: Optimización de los Parámetros del Front-End para la Pendiente Espectral

Los parámetros del *front-end* se reoptimizaron para la característica de pendiente espectral, y se probaron sobre todas las bases de datos. En el contexto de pendiente espectral, encontramos también que el funcionamiento del sistema variaba significativamente con la elección de los valores de los parámetros *front-end*. De nuevo, se desarrollaron sucesivas líneas de investigación para seleccionar el mejor *front-end*.

El sistema que dio como resultado el mejor funcionamiento promedio fue idéntico al mejor sistema del apartado anterior.

De todo lo anterior, podemos extraer que el funcionamiento del sistema de identificación de locutores fluctúa significativamente con la elección de los parámetros del *front-end*. Esta fluctuación podría deberse a una de estas dos razones:

1) las características son muy sensibles a los parámetros del *front-end*, y
2) los modelos generados son sensibles a pequeños cambios en los valores de los parámetros.

Es posible que las funciones de densidad de probabilidad usadas para representar las características no sean Gaussianas. Sin embargo, dado que cada elemento de la característica se representa por una función de densidad Gaussiana mixta, no es probable una adecuación o ajuste pobre entre el modelo y los datos. La variación sobre las bases de datos debería ser atribuída, por tanto, a la variación en las características del canal sobre las diferentes bases de datos, como a la sensibilidad de los parámetros *front-end* a las características del canal. Las nuevas características con el nuevo *front-end* son todavía sensibles a las variaciones del canal. En los apartados siguientes, nos dirigimos a esta cuestión desde un punto de vista de modelado, y mostramos que las nuevas características, junto con los modelos modificados pueden mejorar significativamente el funcionamiento del sistema.

2.3.3.4.3.6: Combinación de Diferentes Características

En la mayoría de los experimentos desarrollados sobre la identificación de locutores, usar el Cepstrum o la pendiente espectral dio como resultado un funcionamiento similar. Si las dos características contienen información complementaria, su combinación se puede esperar que funcione mejor que cualquiera de las características sola. Para verificar estas hipótesis, combinamos las dos características tomando, para cada muestra de prueba, la media aritmética de las probabilidades normalizadas de las observaciones obtenidas para cada característica individualmente. El factor de normalización para cada característica es simplemente la longitud del vector

característico. Esto previene de que el vector característico con mayor número de componentes domine la marca total.

Se observó que los sistemas combinados funcionaban uniformemente mejor que cualquiera de las características sola.

2.3.3.4.3.7: Extensión al Reconocimiento de Voz

Las técnicas descritas en las secciones previas se probaron sobre una tarea de reconocimiento de voz. Viterbi determinó dos nuevos conjuntos de parámetros de *front-end* para reconocimiento de voz alineando las transcripciones de unas cuantas horas de voz, modelando cada llamada con independencia del contexto con un GMM, y encontrando los parámetros *front-end* que minimizaban la tasa de error de clasificación de llamadas para las características Cepstrum y pendiente espectral. Estos *front-ends* se usaron después para desarrollar un experimento de reconocimiento de voz en el conjunto de desarrollo de 1995 para evaluación del reconocimiento continuo de la voz de un extenso vocabulario (Large Vocabulary Continuous Speech Recognition, LVCSR) sobre la base de datos Spanish Callhome. Esta base de datos consiste en voz de conversación telefónica sin restricciones, contiene muchos dialectos diferentes de español, y tiene tasas de error de reconocimiento notoriamente altas. La eliminación de la media Cepstrum se aplicó a nivel de frase en todos los experimentos.

La optimización de los parámetros del *front-end* proporcionó una reducción sobre la tasa de error de palabras (Word Error Rate, WER) sobre un estado del arte del reconocedor de voz, lo cual supuso una mejora significativa dada la dificultad de la tarea. Sin embargo, la mejora proporcionada por la pendiente espectral en los problemas de reconocimiento de locutores no se traslada al contexto de reconocimiento de voz. Esto confirma ‘a posteriori’ que la pendiente espectral transporta información que es más específica del locutor, por ejemplo la caída glotal, que de la declaración.

2.3.3.4.4: Problema del Modelo de Transformación: Métodos de Compensación de Canal basados en Modelos

La segunda parte de este trabajo trata de desarrollar algoritmos de transformación que obtienen modelos de locutores más robustos a los desajustes acústicos.

Muchos sistemas ASR se basan en la sustracción de la media del Cepstrum (*Cepstral Mean Subtraction, CMS*) para compensar los efectos del canal. Es bien sabido, sin embargo, que los desajustes del canal pueden ser aun una fuente significativa de errores después de la CMS. Por esta razón, se han propuesto en la literatura métodos de transformación Cepstrum más sofisticados. En diversos estudios, los vectores de compensación Cepstrum se derivan de una base de datos estéreo y se aplican a los datos de entrenamiento para ajustar los cambios del entorno. La compensación de vectores depende tanto de la SNR como de la identidad fonética de las tramas. Posteriormente, se estima una transformación afín de los vectores Cepstrum de una porción estéreo de la base de datos objeto de estudio, y después se aplica a los datos de entrenamiento.

El efecto de los canales de transmisión sobre la voz también se ha tratado en el contexto de reconocimiento de voz, donde los desajustes acústicos incrementan la confusión entre los teléfonos y conducen a errores en el reconocimiento de palabras. Sin embargo, pocos de los algoritmos desarrollados para el reconocimiento de voz pueden ser realmente aplicados al problema de reconocimiento de locutores. Por ejemplo, los algoritmos de adaptación que ajustan las características o los modelos para representar mejor los datos de prueba son difíciles de usar en el reconocimiento de locutores: si los modelos de locutores se ajustan a los datos de prueba, todos ellos convergen eventualmente hacia el mismo modelo, y se pierde la capacidad de discriminación del locutor. Otros algoritmos de reconocimiento de voz han atacado el problema del desajuste asumiendo que se dispone de un conocimiento “a priori” del desajuste: algunos algoritmos requieren datos estéreo que representen ambas condiciones, otros necesitan muestras de sonidos similares a través de diferentes canales. Estas aproximaciones son difíciles de implementar en el reconocimiento de locutores debido a la dificultad práctica de requerir cada uno de los locutores grabar grandes cantidades de

voz sobre múltiples canales. En el caso de la voz telefónica, este problema se podría aliviar dividiendo los canales en dos o tres categorías. En este trabajo, sin embargo, preferimos asumir que no se proporciona o extrae de la forma de onda de la voz un conocimiento "a priori" del desajuste, y mostramos que se puede obtener una mejora significativa sin tal conocimiento.

La técnica que proponemos compensa los desajustes del canal transformando los modelos de locutores. Hace uso de una base de datos auxiliar que contiene grabaciones estéreo para computar a las que nos referimos como una "distribución de varianza sintética". Esta distribución se puede usan entonces para derivar una transformación que se aplica a las varianzas de los modelos de locutores construidos sobre datos de entrenamiento respecto de otras bases de datos. Se proponen dos transformaciones de este tipo. Estas esencialmente aumentan las varianzas de los modelos de locutores en una cantidad apropiada para hacer los modelos de locutores más robustos a los efectos del canal. Estas transformaciones se pueden aplicar a diferentes características de voz, y se han probado tanto con modelos basados en Cepstrum como con modelos basados en la nueva característica descrita.

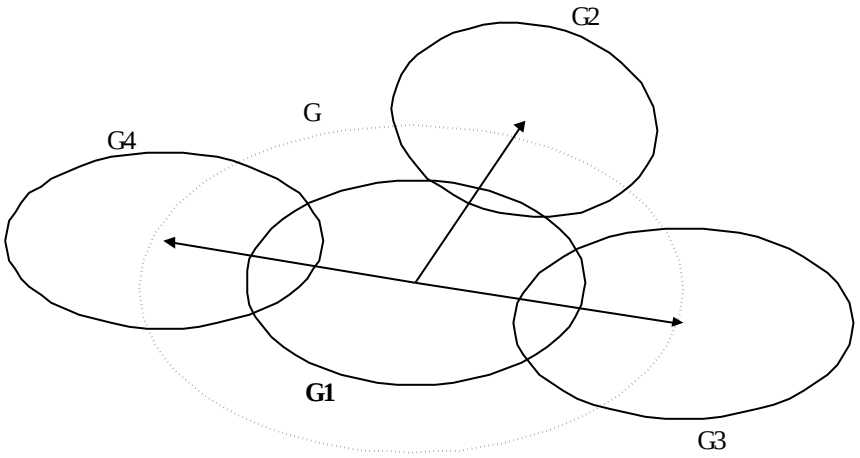


Figura 2.5: Agrupaciones de puntos dato en el espacio característico bidimensional.

El método que proponemos esencialmente desarrolla la compensación del canal, en contraposición a la adaptación del canal. Aspira a hacer los modelos de locutores más robustos a los desajustes del canal incrementando apropiadamente sus varianzas mientras se mantienen fijas sus medias. Los incrementos de la varianza son diferentes para cada coeficiente Cepstrum. Están pensados para tener en cuenta los desplazamientos desconocidos de las medias que se producen en las características cuando el canal cambia, así como para posibles variaciones de la varianza. La Figura 2.5 ilustra esto conceptualmente en un espacio característico bidimensional (2-D). Si G1 es un grupo de características observadas sobre los datos de entrenamiento recogidos de una unidad telefónica dada, las mismas tramas de voz transmitidas por otra unidad podrían parecerse a G2, ó G3 ó G4. Como nuestro sistema de referencia usa modelos Gaussianos mixtos, podemos pensar en G1 como una de las Gaussianas de las GMM de un locutor. Los cambios exactos en la media y varianza de G1 a G2, G3 ó G4 generalmente son desconocidos a la hora de la prueba. En lugar de intentar estimarlos a partir de los datos, reemplazamos G1 por G, una Gaussiana que cubre las posibles regiones en las que podemos esperar que se alojen los datos cuando son transmitidos a través de distintas líneas telefónicas. Las varianzas de los grupos G de todos los modelos de locutores constituyen lo que es referido como una Distribución de Varianza Sintética. Esta distribución de la varianza puede entonces usarse para obtener transformaciones de la varianza para otras bases de datos.

Como se discute en el siguiente apartado, esta aproximación puede también de algún modo compensar otros dos factores: la típicamente limitada cantidad de datos de entrenamiento y el tamaño limitado de los modelos de locutores.

2.3.3.4.4.1: Cantidad de Datos de Entrenamiento y Tamaño del Modelo

En condiciones de ajuste o concordancia, el funcionamiento de un sistema de identificación de locutores depende ampliamente de la cantidad de datos de entrenamiento disponibles: cuantos más datos haya, mejor se puede modelar la voz del locutor, y menor es la tasa de error. Esta observación también se sostiene para sistema con desajustes. Para comprobar esto, durante los experimentos, los conjuntos de prueba

se mantuvieron fijos conforme se incrementaba la cantidad de datos de entrenamiento. Se muestra que, incluso desajustados con los datos de prueba, una mayor cantidad de datos de entrenamiento disminuye significativamente la tasa de error de identificación de locutores. También cantidades mayores de datos de entrenamiento proporcionan modelos con más Gaussianas para obtener modelos más pequeños.

La cantidad de datos usados para construir un GMM y su número de Gaussianas se refleja directamente sobre la distribución de la varianza de las Gaussianas. Para una cantidad de datos dada, las Gaussianas de grandes GMM's tienen varianzas más bajas porque modelan menor cantidad de puntos dato. Además, para un tamaño del modelo dado, la varianza promedio aumenta con la cantidad de datos de entrenamiento. Esto se produce porque el algoritmo EM trata efectivamente de minimizar las varianzas de las Gaussianas del modelo, lo cual se puede conseguir cuando hay menos puntos dato por Gaussiana.

Esta observación sugiere que aumentar artificialmente las varianzas de las GMM's puede ser útil para compensar la ausencia de datos de entrenamiento, y para obtener modelos mayores que construir.

2.3.3.4.4.2: Distribución de la Varianza Sintética

Usando una base de datos que contenga voz obtenida de un micrófono y voz telefónica grabada en modo estéreo, se puede calcular una distribución de varianza sintética, como se ilustra en la Figura 2.5. Las muestras de micrófono de la base de datos se usan para construir los grupos G1, y sus grabaciones estéreo telefónicas se usan para estimar las varianzas de los grupos G. Debido a que los coeficientes Cepstrum de orden más bajo típicamente tienen un rango dinámico más amplio que los coeficientes de orden más alto, la distribución de la varianza se estima separadamente para cada dirección del espacio característico Cepstrum.

El algoritmo para el cómputo de la distribución de varianza sintética se puede resumir como sigue:

1) Seleccionar unas cuantas frases obtenidas mediante micrófono de cada locutor y construir con ellas un conjunto de GMM's de Gaussianas-N_g que se usarán como clasificadores de tramas.

2) Para cada locutor de la base de datos:

   a) Usar el GMM del locutor para etiquetar cada trama de los datos de micrófono restantes del locutor con el índices de la Gaussiana que maximiza su similitud logarítmica, esto es, clasificar las tramas en N_g grupos;

   b) Para cada Gaussiana del GMM (para cada grupo):

3) Calcular la media, m_s, y la varianza, s_s^2, de las tramas agrupadas por esta Gaussiana;

4) Calcular la varianza, s_T^2, de las grabaciones estéreo de estas tramas.

Estas grabaciones estéreo comprenden tramas grabadas sobre diversas unidades telefónicas. Para compensar el desplazamiento de las medias que se produce entre los datos 'microfónicos' y los telefónicos, se prefiere calcular la varianza s_T^2 con respecto a la media m_s de las tramas de micrófono más que con respecto a la media de las tramas telefónicas, m_T.

Las varianzas, s_T^2, constituyen las distribución de varianzas sintética deseada.

Usamos símbolos en cursiva para las medias y varianzas para enfatizar que estas son vectores de N_c (el número de coeficientes Cepstrum) elementos. La distribución de varianzas sintética es, por tanto, N_c-dimensional.

Se obtiene, como esperábamos, que la mayoría de las varianzas telefónicas son mayores que las varianzas con micrófono correspondientes. Esto no es inesperado porque hemos observado en el apartado anterior que los coeficientes Cepstrum de orden más alto son más sensibles a los efectos del canal.

2.3.3.4.4.3: Transformaciones Afines de las Varianzas

Los modelos de locutores entrenados mediante bases de datos que contienen voz recogida de un entorno acústico único (único terminal telefónico, línea telefónica única) pueden beneficiar mediante esta transformación para modificar sus varianzas e incrementar su cobertura acústica.

Expresando la transformación afín como $y = m_i x + t_i$ donde i hace referencia al coeficiente Cepstrum C_i, y x e y representan, respectivamente, la varianza de un grupo de tramas obtenidas mediante micrófono y la varianza de las correspondientes tramas telefónicas, los parámetros m_i y t_i se pueden estimar a partir de los datos, usando un ajuste por mínimos cuadrados:

$$m_i = \frac{\left\langle s^{\,2}_{T,p,j}(i)\right\rangle\left\langle s^{\,2}_{S,p,j}(i)\right\rangle - \left\langle s^{\,2}_{S,p,j}(i)\, s^{\,2}_{T,p,j}(i)\right\rangle}{\left(\left\langle s^{\,2}_{S,p,j}(i)\right\rangle\right)^2 - \left\langle\left(s^{\,2}_{S,p,j}(i)\right)^2\right\rangle}, \qquad (2.22)$$

$$t_i = \left\langle s^{\,2}_{T,p,j}(i)\right\rangle - m_i\left\langle s^{\,2}_{S,p,j}(i)\right\rangle \qquad (2.23)$$

donde $s^{\,2}_{S,p,j}(i)$ y $s^{\,2}_{T,p,j}(i)$ denotan, respectivamente, la varianza de los datos no telefónicos y telefónicos para la Gaussiana j-ésima del modelo del locutor p, para el coeficiente Cepstrum C_i, y donde $\langle . \rangle$ indica el promedio sobre p y j, es decir, sobre todas las Gaussianas y todos los locutores.

Las ecuaciones de transformación de la varianza vienen descritas por

$$s^{\,2}_{tfmed,q,l}(i) \equiv m_i\, s^{\,2}_{q,l}(i) + t_i \qquad (2.24)$$

donde $s^2_{q,l}(i)$ representa la varianza a transformar (más específicamente, la varianza de la l-ésima Gaussiana del modelo del locutor q, para C_i), y $s^2_{tfmed,q,l}(i)$ representa la misma varianza tras la transformación.

Un problema del método de transformación afín es que asume implícitamente que los datos de entrenamiento son proporcionados por un entorno acústico único. Si, en su lugar, los datos de entrenamiento se recogen de unas cuantas líneas telefónicas diferentes, uno podría esperar que estos datos cubren más del espacio característico y que el reducido desajuste requiere una transformación de la varianza diferente. Como los datos estéreo son difíciles de recoger de dos o más unidades telefónicas simultáneamente, se debe desarrollar otro método para tratar esta situación.

2.3.3.4.4.4: Traducción a un 'Target' Fijo

Para propósitos de análisis, la transformación afín se puede simplificar en una parte de escalado modelada por la pendiente m_i o en una parte de traducción modelada por el offset t_i. Se demuestra que la parte más significativa de la transformación se debe a su componente de offset aditiva. Esto se puede justificar intuitivamente: en primera aproximación, la voz proveniente de una línea telefónica se puede representar en el dominio Cepstrum como un proceso aleatorio resultante de la suma de una contribución de voz limpia y un efecto del canal. Como las señales son aditivas, también lo son sus varianzas. Por tanto, el término de traducción de la transformación de la varianza corresponde a una estimación del promedio de la varianza del canal. La transformación afín completa, con su término de escalado, afina este modelo teniendo en cuenta efectos no aditivos.

El método de traducción a un *target* fijo toma ventaja de esta observación para simplificar y generalizar la transformación de la varianza y permitirle tratar con entrenamiento multilínea y, como subproducto, compensar las limitadas cantidades de datos de entrenamiento.

En este método, la distribución de varianza sintética se ve como un 'target' u objetivo fijo a alcanzar por las varianzas de los modelos de locutores. Las varianzas de los modelos del locutor se traducen en una cantidad tal para hacerlos iguales a la media de la distribución de varianzas sintética. Matemáticamente, se puede describir la transformación como

$$s^{2}_{tfmed,q,l}(i) \equiv s^{2}_{q,l}(i) + t_i \qquad (2.25)$$

donde

$$t_i = \left\langle s^{2}_{T,p,j}(i) \right\rangle - \left\langle s^{2}_{q,l}(i) \right\rangle. \qquad (2.26)$$

A condición de que la distribución de varianzas sintética se calcule con una gran cantidad de datos de entrenamiento, el término de traducción t_i también corrige las varianzas del modelo del locutor, $s^{2}_{q,l}(i)$, para ser subestimada a causa de una falta de datos de entrenamiento.

Además de su capacidad de compensar pequeñas cantidades de datos de entrenamiento, este método se extiende fácilmente a condiciones de entrenamiento que incluyen más de una línea porque no realiza ninguna afirmación acerca de las condiciones de entrenamiento (en contraposición al método de transformación afín).

Se comprueba experimentalmente que la mejora proporcionada por la transformación de la varianza es esencialmente independiente del número de locutores. Transformar las varianzas modelo mejora el ajuste entre los modelos de locutores y los

datos de prueba, independientemente del número de locutores que exista en la base de datos.

2.3.3.4.4.5: Transformación de la Varianza con Múltiples Características

Hasta ahora, asumíamos que el Cepstrum era la única característica usada para la identificación de locutores. Cabe la posibilidad de construir un sistema combinando las características optimizadas del apartado 2.2.3.4.3 con la transformación de la varianza con blanco fijo. Los resulados de los experimentos realizados muestran que el sistema combinado reduce en gran medida la tasa de error en el reconocimiento.

2.3.3.4.4.6: Extensión a Reconocimiento de Locutores de Conjunto Abierto

Todos los estudios presentados son para identificación de locutores de conjunto cerrado. Otro problema importante es aquél del reconocimiento de locutores de conjunto abierto, donde los locutores 'blanco' (o auténticos locutores) deben ser identificados y los locutores 'impostores' deben ser rechazados. El reconocimiento de locutores de conjunto abierto implica muchos problemas que están fuera del alcance de este trabajo, sin embargo, se han realizado pruebas sobre reconocimiento de conjunto abierto, y muestran que la mejora del funcionamiento que observamos en la identificación de conjunto cerrado se mantiene en el caso de reconocimiento de locutores de conjunto abierto. La combinación de las características, junto con las transformaciones de la varianza, mejoraron significativamente todos los criterios que evaluamos.

2.3.3.4.5: Conclusiones de los Métodos Planteados

Hemos intentado compensar los efectos del canal en la identificación de locutores optimizando los parámetros del *front-end*, modificando el cálculo de nuestro banco de filtros, introduciendo una nueva característica, y transformando los modelos del locutor. Aunque se ha mostrado que se obtienen mejoras significativas, únicamente hemos arañado la superficie. En el contexto de extracción de la característica, la ganancia del funcionamiento se obtiene usando únicamente características del sistema (banco de

filtros basado en la Cepstrum y banco de filtros basado en la pendiente Cepstrum). Existe una necesidad de desarrollar algoritmos robustos de extracción de características fuente. En el contexto de modelo de transformación, el algoritmo de compensación de blanco fijo ha dado como resultado una significativa ganancia de funcionamiento, pero realmente no ha compensado por completo los efectos del canal. Esta transformación debería ser extendida a transformaciones dependientes del locutor y dependientes del micrófono, de las que se puede esperar que den más mejoras.

2.4: Métodos de Reconocimiento de Locutores con Dependencia del Texto

Los métodos con dependencia del texto se basan normalmente en técnicas de ajuste de patrones. En esta aproximación, la declaración de entrada es representada por una secuencia de vectores característicos, generalmente, vectores característicos espectrales de corto plazo. El eje de los tiempos de la declaración de entrada y cada patrón de referencia o modelo de referencia de los locutores registrados se alinean usando un algoritmo de torsión dinámica en el tiempo (*Dynamic Time Warping*, DTW) y se calcula el grado de similitud entre ellos, acumulado del principio al final de la declaración.

El modelo oculto de Markov (HMM) puede modelar eficientemente la variación estadística de las características espectrales. Por tanto, los métodos basados en HMM fueron introducidos como extensión de los métodos basados en DTW, y han conseguido tasas de reconocimiento significativamente mejores.

2.5: Métodos de Reconocimiento de Locutores con Independencia del Texto

Uno de los métodos de reconocimiento de locutores con independencia del texto con mayor éxito se basa en la cuantización vectorial (*Vector Quantization*, VQ). En este

método, los libros de código VQ, que consisten en un pequeño número de vectores característicos representativos, se usan como medio eficiente de caracterización de locutores específicos. El libro de código específico de un locutor se genera agrupando los vectores característicos de entrenamiento de cada locutor. En la etapa de reconocimiento, una declaración de entrada se cuantiza vectorialmente usando el libro de código de cada locutor de referencia, y la distorsión VQ acumulada sobre la declaración de entrada completa se usa para tomar la decisión de reconocimiento.

La variación temporal de los parámetros de la señal de voz a lo largo del tiempo (o a largo plazo) se puede representar por transiciones estocásticas de Markov entre estados. Por tanto, se han propuesto los métodos que usan un HMM ergódico, en el que todas las transiciones posibles entre estados están permitidas. Los segmentos de voz se clasifican en una de las categorías de la amplia fonética correspondiente a los estados del HMM. Tras la clasificación, se seleccionan las características apropiadas.

En la fase de entrenamiento, se generan los patrones de referencia y se calculan los umbrales de verificación para cada categoría fonética. En la fase de verificación, después de la categorización fonética, una comparación con el patrón de referencia para cada categoría particular proporciona una tasa de verificación para esa categoría. La tasa de verificación final es una combinación lineal ponderada de las tasas de cada categoría.

Este método fue extendido a la clase más rica de HMMs de mezcla autorregresiva. En estos modelos, los estados se describen como una combinación lineal (mezcla) de fuentes AR. Se puede demostrar que los modelos mixtos son equivalentes a un HMM más grande con estados simples, con restricciones adicionales sobre las posibles transiciones entre estados.

Se ha mostrado que un método HMM ergódico continuo es bastante superior a un método HMM ergódico discreto, y que un método HMM ergódico continuo es tan robusto como un método basado en VQ cuando se dispone de los suficientes datos de entrenamiento. Sin embargo, cuando se dispone de pocos datos, el método basado en VQ es más robusto que un método HMM continuo.

Recientemente ha sido propuesto un método que emplea características estadísticas dinámicas. En este método, se aplica un modelo de autorregresión multivariable (MAR) a las series temporales de vectores Cepstrum y se usa para caracterizar locutores. Resultaron tasas de identificación y verificación casi iguales a las obtenidas por un método basado en HMM.

2.6: Método de Reconocimiento de Locutores con Texto Propuesto

En el método de reconocimiento de locutores con texto propuesto, el sistema de reconocimiento propone a cada usuario una nueva sentencia clave cada vez que el sistema se usa, y acepta la declaración de entrada sólo cuando decide que fue el locutor registrado quien repitió la frase propuesta. La frase puede presentarse en forma de caracteres o bien ser pronunciada por una voz sintetizada. Debido a que el vocabulario es ilimitado, los impostores esperados no pueden conocer por anticipado qué frase se les solicitará. Este método no sólo reconoce locutores con seguridad, sino que también puede rechazar declaraciones cuyo texto difiere del texto propuesto, incluso si éste lo pronuncia el locutor registrado. Por tanto, una voz grabada puede ser correctamente rechazada.

Este método se ve facilitado usando modelos de fonemas de locutores específicos como unidades acústicas básicas. Uno de los mayores problemas de aplicación de este método es cómo crear correctamente estos modelos de fonemas de locutores específicos a partir de declaraciones de entrenamiento de tamaño limitado. Los modelos de fonemas se representan por modelos HMM Gaussianos mixtos continuos, y se construyen adaptando modelos de fonemas de locutores independientes a la voz de cada locutor. Con el fin de adaptar correctamente los modelos de los fonemas que no están incluidos en las declaraciones de entrenamiento, se propuso recientemente un nuevo método de adaptación también basado en HMMs.

En la etapa de reconocimiento, el sistema concatena los modelos de fonemas de cada locutor registrado para crear una sentencia HMM, de acuerdo con el texto propuesto. A continuación, se calcula el parecido de la entrada de voz contrastada con la frase modelo y se usa para la decisión de reconocimiento del locutor. Si el parecido es lo suficientemente alto, el locutor se acepta como el locutor esperado.

Capítulo 3: Descripción Global de la Solución Aportada: Identificación de Locutores usando Propiedades Basadas en la Expansión en Serie de Fourier – Bessel

3.1: Introducción

Las funciones base no senoidales con comportamiento cuasiperiódico poseen una mayor similitud estructural con las señales de voz que las senoides de la transformada de Fourier. Tanto los segmentos de voz como las funciones de Bessel presentan cuasiperiodicidad y amplitud decreciente con el tiempo. Por tanto, la representación de señales de voz usando estas funciones base puede terminar en menos componentes que usando las senoidales. Basándonos en esta observación, se ha ensayado con las funciones aperiódicas no senoidales tales como las senoides moduladas por una exponencial y las funciones de Bessel de primer tipo para el análisis de voz. En este

proyecto, consideramos un conjunto de características de la voz obtenidas usando las funciones de Bessel de primer orden en una aplicación de identificación de locutores. Mostramos que las funciones de Bessel se pueden usar para representar formas de onda de voz y que las características derivadas de la representación son comparables a un conjunto de características en el dominio Cepstrum en términos de la percepción de la voz y la identificación de locutores.

3.2: Funciones de Bessel

Antes de entrar en el estudio de las funciones de Bessel, debemos introducir algunos teoremas matemáticos que nos serán útiles para su posterior definición[1].

3.2.1: Método Extendido de las Series de Potencias.(Ecuación de Índice Inicial)

Es necesaria una extensión del método de las series de potencias porque varias ecuaciones diferenciales de segundo orden, de gran importancia en muchas aplicaciones, tienen coeficientes que no son analíticos en $x = 0$, pero son tales que se puede aplicar el teorema siguiente:

Teorema 1

Cualquier ecuación diferencial en la forma

$$y'' + \frac{a(x)}{x} y' + \frac{b(x)}{x^2} y = 0, \qquad (3.1)$$

[1] *Teoremas y demostraciones matemáticas para el estudio de las Funciones de Bessel extraídas de "Matemáticas Avanzadas para la Ingeniería". Volumen I. Erwin Kreyszig. Nueva edición en español de la 5ª edición en inglés. Ed. Noriega-Limusa [5].*

en donde las funciones $a(x)$ *y* $b(x)$ *son analíticas en* $x = 0$ *tiene al menos una solución que se puede representar en la forma*

$$y(x) = x^r \sum_{m=0}^{\infty} c_m x^m = x^r (c_0 + c_1 x + c_2 x^2 + \ldots) \qquad (c_0 \neq 0) \qquad (3.2)$$

en la que el exponente r puede ser cualquier número (real o complejo, y que se elige de modo que $c_0 \neq 0$*).*

El punto clave es que, en (3.2), se tenga una serie de potencias multiplicada por una sola potencia de x cuyo exponente no esté restringido a ser un entero no negativo. (Esta última restricción haría que, por definición, la expresión completa fuera una serie de potencias)

El método subsiguiente para resolver (3.1) se basa en este teorema y se llama método de Frobenius.

3.2.2: Ecuaciones de Bessel. Funciones de Bessel de Primera Clase

Una de las ecuaciones diferenciales más importantes en Matemáticas aplicadas es la Ecuación diferencial de Bessel

$$x^2 y'' + xy' + (x^2 - u^2) y = 0 \qquad (3.3)$$

en donde el parámetro ν es un número dado. Se supone que ν es real y no negativo. Esta ecuación es del tipo caracterizado en el Teorema 1, apartado 3.2.1; de donde se tiene una solución de la forma

$$y(x) = \sum_{m=0}^{\infty} c_m x^{m+r} \qquad (c_0 \neq 0) \tag{3.4}$$

Al sustituir esta expresión y sus derivadas en la ecuación de Bessel, se tiene

$$\sum_{m=0}^{\infty} (m+r)(m+r-1)c_m x^{m+r} + \sum_{m=0}^{\infty} (m+r)c_m x^{m+r} + \sum_{m=0}^{\infty} c_m x^{m+r+2} - \mu^2 \sum_{m=0}^{\infty} c_m x^{m+r} = 0. \tag{3.5}$$

Al igualar a cero la suma de los coeficientes de x^{s+r}, se encuentra

$$a) \qquad r(r-1)c_0 + rc_0 - r^2 c_0 = 0 \qquad (s=0)$$

$$b) \qquad (r+1)rc_1 + (r+1)c_1 - \mu^2 c_1 = 0 \qquad (s=1) \tag{3.6}$$

$$c) \qquad (s+r)(s+r-1)c_s + (s+r)c_s + c_{s-2} - \mu^2 c_s = 0 \qquad (s = 2, 3, \ldots).$$

A partir de 3.6(a) se obtiene la ecuación inicial

$$(r+u)(r-u)=0 \tag{3.7}$$

Las raíces son $r_1 = u \quad (\geq 0)$ y $r_2 = -u$.

Primero se determina una solución correspondiente a la raíz r_1. Para este valor de r, la ecuación 3.6(b) da $c_1 = 0$. La 3.6(c) se puede escribir

$$(s+r+u)(s+r-u)c_s + c_{s-2} = 0, \tag{3.8}$$

y, para $r = u$, esto toma la forma

$$(s+2u)sc_s + c_{s-2} = 0. \tag{3.9}$$

Dado que $c_1 = 0$ y $u \geq 0,$ se deduce que $c_3 = 0, c_5 = 0, \ldots,$ sucesivamente. Si, en (3.9), se hace $s = 2m$, para los otros coeficientes se obtiene

$$c_{2m} = -\frac{1}{2^2 m(u+m)} c_{2m-2}, \qquad m = 1, 2, \ldots \tag{3.10}$$

y, de manera sucesiva, se pueden determinar estos coeficientes.

c_0 permanece arbitraria; se acostumbra poner

$$c_0 = \frac{1}{2^u \Gamma(u+1)} \tag{3.11}$$

en donde $\Gamma(u+1)$ es la función gamma. Para los fines actuales, basta saber que $\Gamma(a)$ se define mediante la integral

$$\Gamma(a) = \int_0^\infty e^{-t} t^{a-1} dt \qquad (a > 0) \tag{3.12}$$

Al integrar por partes, se obtiene

$$\Gamma(a+1) = \int_0^\infty e^{-t} t^a dt = -e^{-t} t^a \Big|_0^\infty + a \int_0^\infty e^{-t} t^{a-1} dt. \tag{3.13}$$

La primera expresión de la derecha es cero y la integral que aparece en ese mismo lado es $\Gamma(a)$; esto conduce a la relación básica

$$\Gamma(a+1) = a\Gamma(a). \tag{3.14}$$

Como

$$\Gamma(1) = \int_0^{\infty} e^{-t}\, dt = 1, \qquad (3.15)$$

con base en (3.14) se concluye que

$$\Gamma(2) = \Gamma(1) = 1!, \qquad \Gamma(3) = 2\Gamma(2) = 2!, \ldots \qquad (3.16)$$

y, en general,

$$\Gamma(k+1) = k! \qquad (k = 0, 1, \ldots). \qquad (3.17)$$

Esto indica que la función gamma se puede considerar como una generalización de la función factorial, conocida desde el Cálculo elemental.

Regresando al problema de (3.10), (3.11) y (3.14) se obtiene

$$c_2 = -\frac{c_0}{2^2(u+1)} = -\frac{1}{2^{2+u}1!\Gamma(u+2)}$$

$$c_4 = -\frac{c_2}{2^2 2(u+2)} = \frac{1}{2^{4+u}2!\Gamma(u+3)}$$

y así, sucesivamente; en general,

$$c_{2m} = \frac{(-1)^m}{2^{2m+u}\, m!\Gamma(u+m+1)}. \qquad (3.18)$$

Al introducir estos coeficientes en (3.4) y recordando que $c_1 = 0, c_3 = 0, \ldots,$ se obtiene una solución particular de la ecuación de Bessel, la cual se denota por $J_u(x)$:

$$J_u(x) = x^u \sum_{m=0}^{\infty} \frac{(-1)^m x^{2m}}{2^{2m+u}\, m!\Gamma(u+m+1)}. \qquad (3.19)$$

Esta solución de (3.3) se conoce como función de Bessel de primera clase *y de orden n*. Se ve que tiene la forma de un producto, a saber, x^u por una serie de potencias. Esta serie converge para toda x lo cual puede verificarse con la prueba de la razón.

Con frecuencia, los valores enteros de ν se denotan por n. Así, entonces, para $n \geq 0$,

$$J_n(x) = x^n \sum_{m=0}^{\infty} \frac{(-1)^m x^{2m}}{2^{2m+n}\, m!(n+m)!}. \qquad (3.20)$$

Si, en (3.19), se reemplaza ν por -ν, se tiene

$$J_{-u}(x) = x^{-u} \sum_{m=0}^{\infty} \frac{(-1)^m x^{2m}}{2^{2m-u}\, m!\Gamma(m-u+1)}. \qquad (3.21)$$

Puesto que la ecuación de Bessel contiene a ν^2, J_u y J_{-u} son soluciones de ella para la misma ν. Si ν no es un entero, estas soluciones son linealmente independientes, debido a que el primer término de (3.19) y el primero de (3.21) son múltiples filtros diferentes de cero de x^u y x^{-u} respectivamente. Lo anterior conduce al resultado siguiente:

Teorema 2 (Solución general de la ecuación de Bessel)

Si n no es un entero, una solución general de la ecuación de Bessel, para toda $x \neq 0$ es

$$y(x) = a_1 J_u(x) + a_2 J_{-u}(x). \qquad (3.22)$$

Pero si ν es un entero, entonces (3.22) no es una solución general. Se llega a esta conclusión a partir del:

Teorema 3 (Dependencia lineal de las funciones de Bessel J_n y J_{-n})

Para $u = n$ entero, las funciones de Bessel $J_n(x)$ y $J_{-n}(x)$ son linealmente dependientes, porque

$$J_{-n}(x) = (-1)^n J_n(x) \qquad (n = 1, 2, \ldots). \qquad (3.23)$$

Demostración. Use (3.21) y haga que ν tienda hacia un entero positivo *n*. Entonces las funciones gamma de los coeficientes de los *n* primeros términos tienden al infinito , los propios coeficientes tienden a cero y la suma se inicia con $m = n$. Ya que, en este caso, $\Gamma(m-n+1) = (m-n)!$ [vea (3.17)], se obtiene

$$J_{-n}(x) = \sum_{m=n}^{\infty} \frac{(-1)^m x^{2m-n}}{2^{2m-n} m!(m-n)!} = \sum_{s=0}^{\infty} \frac{(-1)^{n+s} x^{2s+n}}{2^{2s+n}(n+s)!\, s!} \qquad (3.24)$$

en donde $m = n+s$ y $s = m-n$. En (3.20) se ve que la última serie representa a $(-1)^n J_n(x)$. Esto completa la demostración.

3.2.3: Conjuntos Ortogonales de Funciones

Los polinomios de Legendre y las funciones de Bessel gozan de una propiedad llamada ortogonalidad, la cual tiene una importancia general en las Matemáticas aplicadas a la Ingeniería. En esta sección se introducen los conceptos y notaciones correspondientes.

Por tanto, definamos primero la ortogonalidad de las funciones. Sean $g_m(x)$ y $g_n(x)$ dos funciones de valores reales, las cuales están definidas sobre un intervalo $a \le x \le b$ y son tales que la integral del producto $g_m(x)g_n(x)$ sobre este intervalo existe. Esta integral se denotará por (g_m, g_n). Ésta es una notación estándar sencilla que se usa con amplitud. Así entonces,

$$(g_m, g_n) = \int_a^b g_m(x) g_n(x) dx. \qquad (3.25)$$

Se dice que las funciones $g_m(x)$ y $g_n(x)$ son ortogonales sobre el intervalo $a \le x \le b$ si la integral (3.25) es cero, es decir,

$$(g_m, g_n) = \int_a^b g_m(x) g_n(x) dx = 0 \qquad (m \neq n) \tag{3.26}$$

Se dice que un conjunto de funciones de valores reales, $g_1(x), g_2(x), g_3(x), \ldots$ constituyen un conjunto ortogonal de funciones sobre un intervalo $a \le x \le b$ si estas funciones están definidas sobre el intervalo y si todas las integrales (g_m, g_n) existen y son cero para todos los pares distintos de funciones en el conjunto.

La raíz cuadrada no negativa de (g_m, g_m) se llama norma de $g_m(x)$ y generalmente se denota por $\|g_m\|$; por tanto,

$$\|g_m\| = \sqrt{(g_m, g_m)} = \sqrt{\int_a^b g_m^2(x) dx}. \tag{3.27}$$

En todo el análisis que sigue se hará la siguiente:

Suposición general

Todas las funciones con las cuales se trate son acotadas y tales que las integrales que se planteen existen; asimismo, todas las funciones tienen norma diferente de cero.

Es evidente que un conjunto ortogonal $g_1, g_2, \ldots$ sobre un intervalo $a \le x \le b$, cuyas funciones tengan norma 1, satisface las relaciones

$$(g_m, g_n) = \int_a^b g_m(x) g_n(x) dx = \begin{cases} 0 & \text{cuando } m \neq n \\ 1 & \text{cuando } m = n \end{cases} \quad \begin{pmatrix} m = 1, 2, \ldots \\ m = 1, 2, \ldots \end{pmatrix} \quad (3.28)$$

Un conjunto así se conoce como conjunto ortonormal de funciones sobre el intervalo $a \leq x \leq b$.

Obviamente, es posible obtener un conjunto ortonormal a partir de uno ortogonal, si se divide cada función entre su norma.

3.2.3.1: Series de Funciones Ortogonales

Los conjuntos ortogonales dan lugar a importantes tipos de desarrollos en serie, en una forma relativamente simple. En efecto, sea $g_1(x), g_2(x), \ldots$ cualquier conjunto ortogonal de funciones sobre un intervalo $a \leq x \leq b$ y suponga que $f(x)$ es una función dada que se puede representar en términos de las g_j por medio de una serie convergente

$$f(x) = \sum_{n=1}^{\infty} c_n g_n(x) = c_1 g_1(x) + c_2 g_2(x) + \ldots. \quad (3.29)$$

Esta serie se conoce como serie generalizada de Fourier de $f(x)$ y sus coeficientes $c_1, c_2, \ldots$ son las constantes de Fourier de $f(x)$ con respecto a ese conjunto ortogonal de funciones.

En virtud de la ortogonalidad, estas constantes se pueden determinar en una forma muy sencilla. En efecto, si se multiplican ambos miembros de (3.29) por $g_m(x)$ (m fijo), se integra sobre $a \leq x \leq b$ y se supone que es posible integrar término a término, se tiene

$$(f, g_m) = \int_a^b f g_m dx = \int_a^b (\sum_{n=1}^{\infty} c_n g_n) g_m dx = \sum_{n=1}^{\infty} c_n (g_n, g_m). \qquad (3.30)$$

La integral para la cual $n = w$ es igual a $(g_m, g_m) = \|g_m\|^2$, mientras que todas las demás integrales del segundo miembro son cero, debido a la ortogonalidad. De donde,

$$(f, g_m) = c_m \|g_m\|^2, \qquad (3.31')$$

y la fórmula deseada para las constantes de Fourier es

$$c_m = \frac{(f, g_m)}{\|g_m\|^2} = \frac{1}{\|g_m\|^2} \int_a^b f(x) g_m(x) dx \qquad (m = 1, 2, \ldots). \qquad (3.31)$$

3.2.3.2: Ortogonalidad con Respecto a una Función Peso

Algunos conjuntos importantes de funciones reales, $g_1, g_2, \ldots,$ que se presentan en las aplicaciones no son ortogonales, pero poseen la propiedad de que, para alguna función no negativa $p(x)$,

$$\int_a^b p(x) g_m(x) g_n(x) dx = 0 \qquad \text{cuando } m \neq n. \qquad (3.32)$$

Se dice entonces que un conjunto así es ortogonal, con respecto a la función peso $p(x)$, sobre el intervalo $a \le x \le b$. La norma de g_m ahora se define como

$$\|g_m\| = \sqrt{\int_a^b p(x) g_m{}^2(x)dx} \qquad (3.33)$$

y si la norma de cada función g_m es 1, se dice que el conjunto es ortonormal sobre ese intervalo, con respecto a $p(x)$.

Si se hace $h_m = \sqrt{p}g_m$, entonces (3.32) queda

$$\int_a^b h_m(x)h_n(x)dx = 0 \qquad (m \ne n) \qquad (3.34)$$

es decir, las funciones h_m forman un conjunto ortogonal en el sentido usual. En las secciones que siguen se considerarán algunos ejemplos importantes.

Si $g_1(x), g_2(x), \ldots$ es un conjunto ortogonal sobre un intervalo $a \le x \le b$ con respecto a una función peso p, y si una función dada $f(x)$ se puede representar por medio de una serie generalizada de Fourier [véase (3.29)]

$$f(x) = c_1 g_1(x) + c_2 g_2(x) + \ldots, \qquad (3.35)$$

entonces es posible determinar como antes a las constantes de Fourier $c_1, c_2, \ldots$ de $f(x)$ con respecto a ese conjunto; la única diferencia es que ahora esa serie tiene que multiplicarse por pg_m (en lugar de g_m) antes de integrar. Los otros pasos son los mismos antes descritos y dan

$$c_m = \frac{1}{\|g_m\|^2} \int_a^b p(x) f(x) g_m(x) dx \qquad (m = 1, 2, \ldots). \tag{3.36}$$

en donde la norma ahora queda definida por (3.33).

3.2.4: Problema de Sturm-Liouville

En las Matemáticas aplicadas a la Ingeniería surgen, como soluciones de las ecuaciones diferenciales lineales de segundo orden, varios conjuntos ortogonales importantes de funciones de la forma

$$[r(x)y']' + [q(x) + lp(x)]y = 0 \tag{3.37}$$

sobre algún intervalo dado $a \le x \le b$ satisfaciendo condiciones de la forma

$$a) \quad k_1 y(a) + k_2 y'(a) = 0 \qquad (k_1 \text{ y } k_2 \text{ no son ambas cero}) \tag{3.38}$$

$b)\quad l_1 y(b) + l_2 y'(b) = 0 \qquad (l_1 \text{ y } l_2 \text{ no son ambas cero})$

Aquí, l es un parámetro y k_1, k_2, l_1, l_2 son constantes reales dadas.

En relación con (3.37) y (3.38) se utilizan los términos siguientes: la ecuación (3.37) se llama ecuación de Sturm-Liouville. Se verá que las ecuaciones de Legendre, Bessel y otras se pueden escribir en la forma (3.37). Las condiciones (3.38) reciben el nombre de condiciones en la frontera, en virtud de que se refieren a los puntos extremos (puntos frontera) $x = a$ y $x = b$ de ese intervalo. Una ecuación diferencial junto con condiciones en la frontera constituye lo que se conoce como problema con valores en la frontera. El problema con valores en la frontera (3.37), (3.38) propuesto es un problema de Sturm Liouville.

En (3.37) y (3.38) se ve directamente que, para todo número l el problema tiene la solución trivial $y \equiv 0$, es decir, $y(x) = 0$ para toda x en ese intervalo. Las soluciones $y \neq 0$ si existen, se llaman *funciones características* (también *funciones propias*) o eigenfunciones del problema, y los valores de l para los que esas soluciones existen son los *valores característicos* (*valores propios*) o eigenvalores del problema.

Se puede demostrar que, bajo condiciones un tanto generales sobre las funciones p, q y r de (3.37), el problema de Sturm-Liouville (3.37), (3.38) tiene una infinidad de eigenvalores. Las condiciones que se enuncian en el Teorema 4, dado a continuación, son suficientes, junto con $p(x) > 0$ y $r(x) > 0$ sobre $a < x < b$.)

Además, las eigenfunciones de un problema de Sturm-Liouville tienen la importante propiedad de ortogonalidad siguiente, con respecto a la función peso $p(x)$.

Teorema 4 (Ortogonalidad de las eigenfunciones)

Supóngase que las funciones p, q, r y r' de la ecuación de Sturm-Liouville (3.37) son de valores reales y continuas sobre el intervalo $a \le x \le b$. Sean $y_m(x)$ y $y_n(x)$

eigenfunciones del problema de Sturm-Liouville (3.37), (3.38), las cuales corresponden a los eigenvalores diferentes l_m *y* l_n *respectivamente; entonces* y_m, y_n *son ortogonales sobre ese intervalo, con respecto a la función peso* p.

Si $r(a) = 0$ *se puede eliminar (3.38a) del problema; si* $r(b) = 0$ *entonces se puede eliminar (3.38b). Si* $r(a) = r(b)$ *se puede reemplazar (3.38) por*

$$y(a) = y(b), \qquad y'(a) = y'(b). \tag{3.39}$$

Demostración.

Por hipótesis, y_m satisface

$$(ry_m')' + (q + l_m p)y_m = 0, \tag{3.40}$$

y y_n satisface

$$(ry_n')' + (q + l_n p)y_n = 0. \tag{3.41}$$

Al multiplicar la primera ecuación por y_n y la segunda por y_m y sumar, se obtiene

$$\begin{aligned}(l_m - l_n)py_m y_n &= y_m(ry_n')' - y_n(ry_m')' \\ &= [(ry_n')y_m - (ry_m')y_n]'\end{aligned} \tag{3.42}$$

en donde la última igualdad se puede verificar fácilmente efectuando la derivación indicada de la última expresión entre corchetes. Esta expresión es continua sobre $a \le x \le b$ ya que, por hipótesis, r y r' son continuas, y y_m, y_n son soluciones de (3.37). Así entonces, integrando sobre x desde a hasta b se obtiene

$$(l_m - l_n)\int_a^b py_m y_n dx = [r(y_n' y_m - y_m' y_n)]_a^b \tag{3.43}$$

La expresión de la derecha es igual a

$$\begin{aligned}&r(b)[y_n'(b)y_m(b) - y_m'(b)y_n(b)] \\ &- r(a)[y_n'(a)y_m(a) - y_m'(a)y_n(a)].\end{aligned} \tag{3.44}$$

Caso 1. Si $r(a) = 0$ y $r(b) = 0$ entonces la expresión dada en (3.44) es cero; de donde, la expresión del primer miembro de (3.43) debe ser cero y, puesto que l_m y l_n son distintos, se obtiene la ortogonalidad deseada

$$\int_a^b p(x)y_m(x)y_n(x)dx = 0 \qquad (m \ne n) \tag{3.45}$$

sin utilizar las condiciones en la frontera (3.38).

Caso 2. Sea $r(b)=0$, pero $r(a)\neq 0$. Entonces el primer renglón de (3.44) es cero. Se considerará la expresión restante dada en (3.43). A partir de (3.38a) se tiene

$$\begin{aligned} k_1 y_n(a)+k_2 y_n'(a)&=0,\\ k_1 y_m(a)+k_2 y_m'(a)&=0. \end{aligned} \tag{3.46}$$

Sea $k_2\neq 0$; entonces, al multiplicar la primera ecuación por $y_m(a)$ y la última por $-y_n(a)$ y sumar, se tiene

$$k_2\left[y_n'(a)y_m(a)-y_m'(a)y_n(a)\right]=0. \tag{3.47}$$

Puesto que $k_2\neq 0,$ la expresión entre corchetes debe ser cero. Esta expresión es idéntica a la del último renglón de (3.44). De donde (3.44) es cero y, a partir de (3.43), se obtiene (3.45). Si $k_2=0,$ entonces, por hipótesis $k_1\neq 0$ y el argumento de la demostración es semejante.

Caso 3. Si $r(a)=0$ pero $r(b)\neq 0$ la demostración es semejante a la del caso 2 pero, en lugar de (3.38a), ahora se tiene que utilizar (3.38b).

Caso 4. Si $r(a)\neq 0$ y $r(b)\neq 0,$ se tienen que aplicar las dos condiciones en la frontera (3.38) y proceder como en los casos 2 y 3.

Caso 5. Sea $r(a)=r(b)$; entonces (3.44) toma la forma

$$r(b)\left[y_n'(b)y_m(b) - y_m'(b)y_n(b) - y_n'(a)y_m(a) + y_m'(a)y_n(a)\right]. \qquad (3.48)$$

Es posible utilizar (3.38) como antes y concluir que la expresión entre corchetes es cero. Sin embargo, de inmediato se ve que también se concluiría esto a partir de (3.39), de modo que se puede reemplazar (3.38) por (3.39). Por consiguiente, (3.43) conduce a (3.45), como antes. Esto completa la demostración del Teorema 4.

Una serie generalizada de Fourier (véase la sección 3.2.3), en la cual el conjunto ortogonal es uno de eigenfunciones, se conoce como desarrollo en eigenfunciones.

Los eigenvalores de los problemas de Sturm-Liouville tienen las interesantes propiedades que siguen:

Teorema 5 (Eigenvalores reales)

Si el problema de Sturm-Liouville (3.37), (3.38) satisface la condición enunciada en el teorema 4 y p) es positiva en el intervalo completo $a \le x \le b$ *(o negativa en todo punto de ese intervalo), entonces todos los eigenvalores del problema son reales.*

Demostración.

Sea $l = a + ib$ un eigenvalor del problema y

$$y(x) = u(x) + iv(x) \qquad (3.49)$$

una eigenfunción correspondiente; aquí a, b, u y v son reales. Sustituyendo esto en (3.37), se tiene

$$\left(ru' + irv'\right)' + \left(q + ap + ibp\right)\left(u + iv\right) = 0. \qquad (3.50)$$

Esta ecuación compleja es equivalente al siguiente par de ecuaciones, para las partes reales e imaginarias:

$$\begin{aligned}(ru')'+(q+ap)u-bpv=0.\\(rv')'+(q+ap)v+bpu=0.\end{aligned} \tag{3.51}$$

Al multiplicar la primera ecuación por v y la segunda por $-u$ y sumando, se obtiene

$$\begin{aligned}-b\left(u^2+v^2\right)p &= u(rv')'-v(ru')'\\ &= [(rv')u-(ru')v]'.\end{aligned} \tag{3.52}$$

La expresión entre corchetes es continua sobre $a \le x \le b,$ por razones semejantes a las mencionadas en la demostración anterior. Así, entonces, si se integra sobre $x,$ desde a hasta $b,$ se obtiene

$$-b\int_a^b \left(u^2+v^2\right)p\,dx = \left[r\left(uv'-u'v\right)\right]_a^b. \tag{3.53}$$

Debido a las condiciones en la frontera, el segundo miembro es cero; esto es como en la demostración anterior. Dado que y es una eigenfunción, $u^2+v^2 \neq 0.$ Como y y

p son continuas y $p > 0$ (ó $p < 0$) sobre el intervalo $a \le x \le b,$ la integral de la izquierda es diferente de cero. De donde $b = 0,$ lo cual significa que $l = a$ es real. Esto completa la demostración.

3.3: Ortogonalidad de las Funciones de Bessel

La ortogonalidad de las funciones de Bessel J_n es bastante importante en las aplicaciones de la ingeniería, por ejemplo en relación con las vibraciones de membranas circulares. $J_n(s)$ satisface la ecuación de Bessel (apartado 3.2.2)

$$s^2 \ddot{J}_n + s\dot{J}_n + (s^2 - n^2) J_n = 0 \qquad (3.54)$$

en donde los puntos denotan derivadas con respecto a s. Se supone que n es un entero no negativo. Haciendo $s = lx$, en donde l es una constante diferente de cero, se tiene $dx/ds = 1/ly$, por la regla de la cadena,

$$\dot{J}_n = J_n' / l \qquad \ddot{J}_n = J_n'' / l^2, \qquad (3.55)$$

en donde los apóstrofos denotan derivadas con respecto a x. Al sustituir,

$$x^2 J_n''(lx) + xJ_n'(lx) + (l^2x^2 - n^2) J_n(lx) = 0. \qquad (3.56)$$

Si se divide entre x, esta ecuación se puede escribir en la forma

$$\left[xJ_n'(lx)\right] + \left(-\frac{n^2}{x} + l^2 x\right) J_n(lx) = 0. \qquad (3.57)$$

Se ve que, para cada n fijo, esta es una ecuación de Sturm-Liouville (3.37), apartado 3.2.4, con el parámetro denotado como l^2, en lugar de l, y

$$p(x) = x, \qquad q(x) = -n^2/x, \qquad r(x) = x. \qquad (3.58)$$

Ya que $r(x) = 0$ en $x = 0$, por el teorema 4 de la última sección se deduce que aquellas soluciones de (3.57), sobre un intervalo $0 \le x \le R$ dado, que satisfagan la condición en la frontera

$$J_n(lR) = 0 \qquad (3.59)$$

forman un conjunto ortogonal sobre ese intervalo, con respecto a la función peso $p(x) = x$. (Tenga presente que, para $n \neq 0$, la función q es discontinua en $x = 0$, pero esto no afecta a la demostración de este teorema.) Se puede demostrar que $J_n(s)$ tiene una infinidad de ceros reales: si $a_{1n} < a_{2n} < a_{3n\ldots}$ denotan los ceros positivos de $J_n(s)$: entonces (3.59) se cumple para

$$lR = a_{mn} \qquad or \qquad l = l_{mn} = \frac{a_{mn}}{R} \qquad (m = 1,2,\ldots), \tag{3.60}$$

y se obtiene el resultado siguiente.

Teorema 6 (Ortogonalidad de las Funciones de Bessel)

Para cada n=0, 1, ...fijo, las funciones de Bessel $J_n(l_{1n}x), J_n(l_{2n}x), J_n(l_{3n}x),\ldots,$ *con* l_{mn} *dada por (3.60), forman un conjunto ortogonal sobre el intervalo* $0 \le x \le R$ *con respecto a la función peso* $p(x) = x$, *es decir,*

$$\int_0^R xJ_n(l_{mn}x)J_n(l_{kn}x)dx = 0 \qquad (k \neq m) \tag{3.61}$$

De donde se han obtenido una infinidad de conjuntos ortogonales y cada uno corresponde a uno de los valores fijos de n.

Puesto que $p(x) = x$, por (3.33)-(3.36) de la sección 3.2.3 se ve que, si una función dada $f(x)$ tiene un desarrollo en eigenfunciones, en términos de uno de esos conjuntos ortogonales, el desarrollo es

$$f(x) = c_1 J_n(l_{1n}x) + c_2 J_n(l_{2n}x) + \ldots \tag{3.62}$$

Ésta se conoce como serie de Fourier-Bessel. Se probará que

$$\left\| J_n(l_{mn}x) \right\|^2 = \int_0^R x J_n^2(l_{mn}x)dx = \frac{R^2}{2} J_{n+1}^2(l_{mn}R), \qquad (3.63)$$

en donde $l_{mn} = a_{mn}/R$. Conforme a (3.36) de la sección 3.2.3, esto implica que las constantes de Fourier c_m de f que aparecen en (3.62) son

$$c_m = \frac{2}{R^2 J_{n+1}^2(a_{mn})} \int_0^R x f(x) J_n(l_{mn}x)dx, \qquad m = 1,2,\ldots \qquad (3.64)$$

Se probará (3.63). Multiplicando (3.57) por $2xJ_n'(lx)$, se obtiene una ecuación que se puede escribir

$$\left\{ \left[xJ_n'(lx) \right]^2 \right\} + (l^2x^2 - n^2)\left\{ J_n^2(lx) \right\}' = 0. \qquad (3.65)$$

Si se integra sobre x, desde 0 hasta R, se encuentra

$$\left[xJ_n'(lx) \right]^2 \Big|_0^R = -\int_0^R (l^2x^2 - n^2)\left\{ J_n^2(lx) \right\}' dx. \qquad (3.66)$$

De donde obtenemos

$$-ns^{-n-1}J_n(s)+s^{-n}\,\dot{J}_n(s)=-s^{-n}J_{n+1}(s). \qquad (3.67)$$

Al multiplicar por s^{n+1} y hacer $s=lx$, se obtiene

$$lxJ_n^{'}(lx)\frac{1}{l}=nJ_n(lx)-lxJ_{n+1}(lx), \qquad (3.68)$$

en donde el apóstrofo denota la derivada con respecto a x. De donde, el primer miembro de (3.66) es igual a

$$\left[\left[nJ_n(lx)-lxJ_{n+1}(lx)\right]^2\right]_{x=0}^{R} \qquad (3.69)$$

Si $l=l_{mn,}$ entonces $J_n(lR)$ y, como $J_n(0)=0$ $(n=1,2,\ldots)$, ese primer miembro queda

$$l_{mn}^2R^2J_{n+1}^2(l_{mn}R). \qquad (3.70)$$

Si se integra por partes el segundo miembro de (3.66), se ve que este segundo miembro se convierte en

$$-\left[\left(l^2x^2-n^2\right)J_n^2(lx)\right]_0^R+2l^2\int_0^R xJ_n^2(lx)dx. \qquad (3.71)$$

Cuando $l = l_{mn,}$ la primera expresión es cero en $x=R$; también es cero en $x=0,$ debido a que $l^2x^2-n^2=0$ cuando $n=x=0,$ y $J_n(lx)=0$ cuando $x=0$ y $n=1,2,\ldots$ A partir de esto y (3.71), con facilidad se obtiene (3.63). Esto completa la demostración.

3.4: Transformada de Fourier-Bessel

La Transformada de Fourier-Bessel es equivalente a la Transformada de Fourier en dos dimensiones con un Kernel radialmente simétrico, y también es conocida como Transformada Hankel.

$$g(u,v)=F\left[f(r)\right]=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(r)e^{-2pi(ux+vy)}dxdy. \qquad (3.72)$$

Sea

$$x+iy=re^{iq} \qquad (3.73)$$

$$u+iv=qe^{if} \qquad (3.74)$$

Por tanto

$$x = r\cos q \tag{3.75}$$

$$y = r\sin q \tag{3.76}$$

$$r = \sqrt{x^2 + y^2} \tag{3.77}$$

$$u = q\cos f \tag{3.78}$$

$$v = q\sin f \tag{3.79}$$

$$q = \sqrt{u^2 + v^2} \tag{3.80}$$

De este modo, obtenemos

$$\begin{aligned} g(q) &= \int_0^{\infty}\int_0^{2p} f(r)e^{-2pirq(\cos f\cos q + \sin f\sin q)}rdrdq \\ &= \int_0^{\infty}\int_0^{2p} f(r)e^{-2pirq\cos(q-f)}rdrdq \\ &= \int_0^{\infty}\int_{-f}^{2p-f} f(r)e^{-2pirq\cos q}rdrdq \\ &= \int_0^{\infty}\int_0^{2p} f(r)e^{-2pirq\cos q}rdrdq \\ &= \int_0^{\infty} f(r)\left[\int_0^{2p} e^{-2pirq\cos q}\right]rdr \\ &= 2p\int_0^{\infty} f(r)J_0(2pqr)rdr, \end{aligned} \tag{3.81}$$

donde $J_0(z)$ es una función Bessel de primera especie de orden cero. Por tanto, la pareja de la transformada de Fourier-Bessel es la siguiente[2]

$$g(k) = \int_0^\infty f(x) J_0(kx) x dx \tag{3.82}$$

$$f(x) = \int_0^\infty g(k) J_0(kx) k dk \tag{3.83}$$

3.5: Representación de Señales usando Funciones de Bessel

Una función *x(t)* arbitraria en el intervalo 0<t<a se representa usando la función de Bessel *J1(t)* por la expansión de Fourier – Bessel (FB), dada por

$$x(t) = \sum_{m=1}^{\infty} C_m J_1\left(\frac{x_m}{a} t\right) \tag{3.84}$$

donde x_m, $m = 1, 2, 3, \ldots$ son las raíces de $J_1(t) = 0$.

[2] *Demostración extraída de "Hankel Transform". Eric W. Weisstein, publicada en la página http://www.br.campus.vt.edu/~akrowne/crc/math/h/h056.htm [15]*

Usando la ortogonalidad del conjunto $\{J_1[(x_m/a)t]\}$, los coeficientes de la expansión de Fourier-Bessel se determinan por[3]

$$C_m = \frac{2}{a^2 J_0(x_m)^2} \int_0^a t x(t) J_1\left(\frac{x_m}{a} t\right) dt \qquad (3.85)$$

Como con las series de Fourier, los coeficientes de Fourier-Bessel $\{C_m\}$ son únicos para una $x(t)$ dada. A diferencia de las funciones base senoidales de las series de Fourier, sin embargo, las funciones de Bessel son aperiódicas y decaen dentro del intervalo *a*. El conjunto cuasiperiódico $\{J_1[(x_m/a)t]\}$, tiene similitud estructural con los segmentos de voz de corta duración, con el intervalo entre sucesivos cruces por cero creciendo lentamente con el tiempo, y aproximándose a π en el límite. Por tanto, se puede obtener una representación eficiente de segmentos de voz en términos del conjunto ortogonal usando los coeficientes C_m.

3.6: Representación de Fourier – Bessel de Formas de Onda de Voz

Una señal de voz se representa usando la versión discretizada de (3.85) en intervalos cortos de voz. Los coeficientes $\{C_m, m = 1, 2, \ldots\}$ se obtienen para cada una de las tramas segmentadas y enventanadas de la voz muestreada. Aunque (3.84) es una serie infinita, se aproxima a $x(t)$ rápidamente con un alto pero finito número de coeficientes.

[3] *"A Comparison of Speaker Identification Results Using Features Based on Cepstrum and Fourier-Bessel Expansion". K. Gopalan, Timothy R. Anderson, Edward J. Cupples, Miembros del IEEE. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 3, Mayo, 1999 [1].*

Una representación aproximada de la señal puede, por tanto, ser obtenida con sólo un número finito de términos.

La efectividad de la representación de la voz por funciones de Bessel se estudió usando diferentes números de coeficientes en el sumatorio dado en (3.84). La base de datos usada para este experimento consistió en 54 muestras de la frase en inglés "Estuvimos fuera hace un año" ("We were away a year ago"), hablada por tres hombres y dos mujeres. Se obtuvieron dos conjuntos de grabaciones de señales de voz a la tasa de 11.000 muestras/s. El principio y final de la voz en cada muestra se detectaban usando umbrales de amplitud. Después de experimentar con diferentes intervalos de tramas de voz, con y sin solapamiento, se descubrió que usar 200 muestras/trama con 100 muestras de solapamiento fue computacionalmente óptimo. Cada trama de datos de voz fue después multiplicada por una ventana de Hamming de 200 puntos para preservar las características espectrales de la señal.

Cuando se usaron en el sumatorio de 100 a 150 coeficientes (i.e., $C_m, m = 1,2, \ldots, M$ donde M está entre 100 y 150) para reconstruir la señal de voz, todas las características de la voz original, incluyendo la identidad del locutor, se observaron presentes. La calidad de la voz sintetizada era lo suficientemente buena para que los oyentes, que no estaban familiarizados con el mensaje, entendieran claramente la voz y distinguieran a los locutores. Además del test de audición, la calidad de la voz también se confirmó por el espectrograma de la voz, que estaba muy próximo al de la voz original.

La calidad de la voz comenzó a deteriorarse para valores de M (límite superior de m) entre 30 y 50; la calidad del mensaje, sin embargo, aún era aceptable. En el límite inferior de M, en particular en 20, la identidad del género del locutor era difícil de determinar, a pesar de que el mensaje era aún discernible con ruido de fondo. La voz sintetizada usando sólo los 10 primeros términos tenía una muy baja calidad de mensaje con solo los componentes de baja frecuencia presentes.

La correspondencia entre el número finito M de términos usados en la reconstrucción en (3.84) y el espectro de la señal resultante se establece por el ancho de banda w_{max} de la señal reconstruida. Como el espectro de frecuencia de $J_1[(x_m/a)t]$ está limitado en banda a $w = x_m/a$ para M términos en el sumatorio, la aproximación de la señal dada por

$$x(t) \approx \sum_{m=1}^{M} C_m J_1\left(\frac{x_m}{a} t\right) \qquad (3.86)$$

tiene un ancho de banda máximo de $w_{max} = x_M/a,$ donde x_M es la M-ésima raíz de $J_1(t) = 0.$

Por tanto, usar los M primeros términos del sumatorio en (3.84) limita el espectro de la voz sintetizada a $x_M/a.$ Consecuentemente, para conservar todas las componentes de frecuencia hasta los 3 kHz en la representación, por ejemplo, se debe elegir un M tal que $x_M \approx 2p(3000)a.$ Para la voz original muestreada a 11000 muestras/s, usar 200 muestras por trama da al intervalo un valor de 18.2 ms. Por tanto, $x_M = 342.72.$ Como la raíz 109-ésima de $J_1(t)$ está próxima a este valor (en 343.18), el sumatorio finito debe tener hasta 109 términos para conservar las componentes espectrales hasta los 3 kHz. Para $M \prec 109,$ la voz sintetizada pierde la información de alta frecuencia. En $M = 50,$ por ejemplo, la energía del espectro de la señal reconstituida está limitada por debajo de 1500 Hz, frecuencia que puede ser más baja que el segundo formante para ciertos locutores y/o fonemas; ciertamente no puede tener la calidad original de la voz de las fricativas. El contenido del mensaje se puede aun preservar, aunque con ruido, dependiendo de lo largo que sea M; por tanto, un valor más bajo de M (por debajo de 50) puede ser aceptable para el almacenamiento y transmisión de la voz.

La mejora en la calidad de la voz, en particular para las fricativas, requiere cierta cantidad de información de alta frecuencia que corresponde al valor más alto de M. Si las componentes de alta frecuencia son predominantes en cualquier trama en la forma de onda de la voz original, estas se manifiestan en forma de grandes coeficientes $|C_m|$ para altos valores del índice m. Por tanto, mediante la elección selectiva de los coeficientes en cada trama basada en sus magnitudes relativas, la energía del espectro tanto a altas como a bajas frecuencias se puede conservar en la voz sintetizada. Así, con un ligero incremento del tiempo de procesado (que surge de la elección de $|C_m|$ en cada trama), se puede obtener una calidad más alta de la voz con un número más bajo de coeficientes. Esta afirmación se verificó usando los máximos de 50, 30, 20 y diez coeficientes. Se descubrió que se necesitaba un mínimo de 20 coeficientes seleccionados para una calidad aceptable de la voz sin perder la identidad del locutor. Esto se compara favorablemente con el requerimiento de más de 30 coeficientes no seleccionados para obtener bajo ruido en la voz sintetizada.

Basándonos en los resultados anteriores de la representación de la voz usando $\{J_1(t)\}$ como el conjunto de funciones base, está claro que un conjunto de un número finito de coeficientes de la representación puede usarse para formar un vector característico que caracterice a voz y locutor.

3.6.1: Identificación de Resultados de Características Basadas en la Transformada de FB

Los siguientes conjuntos de características fueron evaluados para dos bases de datos, denominadas ‘Greenflag database’ y ‘NATO database’. La base de datos Greenflag consistió en un total de 173+41 = 214 declaraciones o palabras de 41 locutores (pilotos), que se transmiten del avión a tierra a una tasa de muestreo de 8000 muestras/s. Una palabra de cada piloto se usó como palabra de referencia. Todos los parámetros se evaluaron usando una longitud de trama de 200 muestras con un solape de 100 muestras.

Para reducir el tamaño del vector característico, un conjunto selecto de coeficientes de Fourier-Bessel, como se describió en el apartado 3.6, se puede usar sin pérdida de información. Aunque los coeficientes se pueden seleccionar basándonos en sus amplitudes relativas o en su contenido de frecuencia, se eligió un conjunto de 15 coeficientes que cubría el rango de frecuencia de la banda de paso comenzando con los índices dados por $m = 10, 20, 35, 50, 60, 70, 80, 90, 100, 115, 130, 145, 160, 175, y\ 190$. En cada uno de estos 15 índices, se computaron 5 coeficientes sucesivos. Estos coeficientes (15 x 5) representan una frecuencia de comienzo de 205 Hz (en $m = 10$) que asciende hasta aproximadamente 3905 Hz (en $m = 195$). Podemos advertir que las frecuencias a las que se produce el máximo de energía del espectro para las funciones base, $J_1([x_m / a]t)$, son de alguna forma análogas a los valores de la escala MEL de frecuencia que se elegirían para la extracción de características del dominio Cepstrum. Para reducir más el tamaño del vector característico, se obtuvo una medida de energía en una estrecha banda de frecuencia en las proximidades de cada índice seleccionado. La magnitud de la amplitud de la señal base en cada uno de los cinco índices se sumó para obtener la medida de la energía como

$$e(i) = \sum_{m=k}^{k+5} |C_m| \qquad (3.87)$$

para $i = 1, 2, \ldots, 15$ y $k = 10, 20, 35, 50, 60, 70, 80, 90, 100, 115, 130, 145, 160, 175, y\ 190$, respectivamente, como se muestra en la Fig. 3.

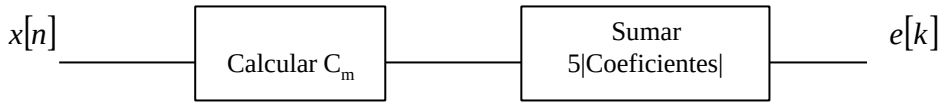


Figura 3.1: Característica de medida de energía usando coeficientes FB.

Se observa que el elemento característico $e(k)$ es análogo a la energía del logaritmo de la salida del filtro de banda crítica en la representación en el dominio Cepstrum. Por tanto, el vector característico de la base de Fourier-Bessel modela la audición perceptiva a las frecuencias seleccionadas. Añadiendo un número fijo de coeficientes (cinco, por ejemplo, aquí), el ancho de banda en cada frecuencia seleccionada se mantiene casi constante con el índice *m*. Para *i*=1 (*k* = 10 y *m* = 10-14), por ejemplo, *e(1)* representa aproximadamente la energía en el ancho de banda de *f1* = *x10/2pa* = 204.93 Hz, a *f2* = *x14/2pa* = 284.95 Hz, u 80 Hz. Para *i* = 10 (*k* = 115 y m = 115-119), *e(10)* representa la energía desde *f1* = 2305 Hz hasta *f2* = 2385 Hz. La medida de la energía *e(i)* como la dada en (3.87) se usó para construir vectores característicos de Fourier-Bessel de 15 elementos para la base de datos Greenflag. Además, también se calcularon vectores característicos de Fourier-Bessel de 20 elementos usando (3.87) con i = 1, 2,..., 20 y k = 5, 15, 25, 35, 45, 65,..., 185, 195. Con el índice del coeficiente de comienzo 5 y el índice final de 195, los vectores de 20 elementos cubrían un rango más ancho de frecuencias que los vectores de 15 elementos tanto en el extremo inferior como en el superior.

Los conjuntos anteriores de características se probaron usando un clasificador comercial de vectores de base cuantizados del Laboratorio de Investigación Entrópico. Usando el mismo clasificador en todas las características y para todas las bases de datos, la efectividad para cada característica puede compararse en términos de la complejidad computacional y de las tasas de identificación. Se observa que el uso de la base de datos Greenflag es ruidoso con ráfagas de ruido de máquinas o motores y chasquidos de micrófono; también, muchas de las palabras de prueba son muy cortas en duración (unas

cuantas centenas de milisegundos) en comparación con las correspondientes palabras de entrenamiento de la base de datos. Sin detección del punto final, por tanto, el tema principal de discriminación de cada locutor sin hacer caso a la duración de la palabra y la calidad de la voz está ante todo en las características o vectores característicos. El funcionamiento del clasificador no parece afectar adversamente a la tasa de identificación.

La elección de los índices del coeficiente en el caso de vectores característicos de la base de Fourier-Bessel, determinó la eficiencia de la característica más que el tamaño de la característica. El vector característico de 20 parámetros cubría aproximadamente las mismas frecuencias centrales y anchos de banda que en el caso usar la Cepstrum de 20 puntos. La inclusión de los componentes de baja frecuencia correspondientes a $m = 5 - 9,$ que cubren aproximadamente 100-200 Hz, elevó ligeramente la tasa.

La energía diferencial en el dominio Fourier-Bessel se obtuvo como la diferencia entre sucesivos elementos de energía, es decir,

$$d(k) = e(k+1) - e(k), \quad k = 1, 2, \ldots, 19. \tag{3.88}$$

Esta característica de energía diferencial es análoga a la energía en una banda diferencial de frecuencias.(Debido al ligero incremento en la tasa de identificación al usar los vectores de energía de 20 elementos de Fourier-Bessel sobre los vectores de 15 elementos, sólo aquel se empleó en construir las características diferenciales de energía.)Como el coeficiente para el índice M es aproximadamente proporcional a la amplitud del espectro a la frecuencia $w_{max} = x_M / a,$ la medida de energía diferencial $d(k)$ es aproximadamente equivalente a la energía del espectro en la banda $w_{k+1} - w_k = (x_{k+1}/a) - (x_k/a).$ Por tanto $d(k)$ se asemeja a la señal de energía filtrada

paso de banda. Claramente, la elección de los coeficientes en la formación de $e(k)$, y por tanto $d(k)$, de nuevo, determina la efectividad del vector característico $d(k)$.

La base de datos NATO

Esta base de datos consistió en palabras cortas de nueve locutores europeos (pilotos), transmitidas del avión a tierra a la tasa de 16000 muestras/s. De un total de 1117 transmisiones, siete palabras de cada locutor se concatenaron para formar palabras de referencia. Las restantes 1054 transmisiones se usaron en la identificación del locutor.

Con 400 muestras/trama, los vectores característicos de energía de 20 elementos basados en Fourier-Bessel cubrían las frecuencias de 105-3905 Hz aproximadamente, usando los índices 5, 15, 25, 35, 45, 65, ..., 185, 195. Mientras que este rango de frecuencias es suficiente para representar la voz, las características del dominio Cepstrum cubren un rango mucho más ancho de 233-7344 Hz. Con la alta tasa de reconocimiento usando este rango de frecuencias en las características del dominio Cepstrum, estaba claro que para esta base de datos, la energía a las frecuencias por encima de 4000 Hz se debía incluir en las características basadas en Fourier-Bessel para la identificación. Por tanto, la longitud de la trama para el uso con la transformada de Fourier-Bessel se cambió a 200 muestras. Para esta longitud de trama, el rango pasó a ser 12.5 ms y la cobertura de frecuencia cambió a 210-7810 Hz. Las dos últimas filas muestran un significativo incremento de la tasa de identificación con la cobertura de frecuencia incrementada. Observamos que usando únicamente cinco coeficientes en el vector característico de energía de la Fourier-Bessel, la banda de frecuencias cubierta en cada elemento, *e(k)*, k = 1, 2, ..., 20, es constante en 210 Hz aproximadamente.

3.7: Discusión

La representación de Fourier-Bessel de la voz es claramente una alternativa a la tradicional representación para aplicaciones de reconstrucción de voz e identificación de locutores basada en la transformada de Fourier. Debido a su comportamiento no

estacionario, similar al de los sonidos de voz incluidos en un intervalo del tono o 'pitch', la representación en términos de los coeficientes de Fourier-Bessel puede ser más compacta que con el uso de las senoides estacionarias. La elección y el número de coeficientes, sin embargo, pueden depender de los sonidos de voz – las vocales requieren coeficientes de índices más bajos y espaciados (o de menor cantidad de índices) mientras que las fricativas requieren coeficientes de índices más altos y en mayor cantidad, por ejemplo.

Como los coeficientes de Fourier-Bessel para el índice *m* representan aproximadamente la amplitud del espectro a la frecuencia $\omega\max = xm/a$, la energía en una banda de frecuencias está íntimamente relacionada con la magnitud de los coeficientes de Fourier-Bessel con índices adyacentes. Computacionalmente, sin embargo, los cálculos en el dominio de Fourier tienen la ventaja de la rapidez y los algoritmos oportunos, que no están disponibles para los cálculos en el dominio de Fourier-Bessel para órdenes de *n* por encima de cero para $J_n(t)$.

Para el aspecto de la comparación con los coeficientes Cepstrum MEL distribuidos basados en la energía en una banda crítica de frecuencias, un creciente número de coeficientes se puede considerar para formar el vector característico *e(k)* en (3.86). A pesar de que esto incrementa el esfuerzo computacional, puede producir tasas de identificación de locutores más altas que son comparables con aquéllas que usan características Cepstrum.

Como conclusión, se ha descrito un método de representación de la señal de voz y extracción de características usando funciones de Bessel de primer orden como funciones base. Se ha mostrado que las características obtenidas de la expansión de Fourier-Bessel de la voz son comparables a las características Cepstrum al representar la energía del espectro. Por tanto, las características basadas en Fourier-Bessel se pueden usar como una alternativa a la representación del espectro de señales de voz. Debido a la naturaleza análoga de las características basadas en Fourier-Bessel con las características del dominio Cepstrum, ambos conjuntos de características producen tasas de identificación de locutores comparables. Con una elección juiciosa de los

coeficientes al formar las características, la expansión de Fourier-Bessel se puede usar eficientemente en aplicaciones de análisis - síntesis de la voz y de identificación de locutores.

3.8: Cuantización Vectorial

La cuantización vectorial se emplea en muchas aplicaciones tales como compresión de imagen y voz, y en reconocimiento de voz y locutores (en general, reconocimiento estadístico de patrones).

Un cuantizador vectorial mapea vectores k-dimensionales en el espacio vectorial R^k en un conjunto finito de vectores $Y = \{y_i : i = 1, 2, \ldots, N\}$. Cada vector y_i se conoce como vector código o palabra de código, y el conjunto de todas las palabra de código se denomina libro de código. Asociado con cada palabra de código, y_{ir} es el más próximo de la región vecina conocida por Región Voronoi, y se define de la forma:

$$V_i = \left\{ x \in R^k : \|x - y_i\| \leq \|x - y_j\|, \ \textit{para todo } j \neq i \right\} \tag{3.89}$$

El conjunto de regiones Voronoi particiona el espacio completo R^k de la forma

$$\bigcup_{i=1}^{N} V_i = R^k$$

$$\textit{para todo } i \neq j \tag{3.90}$$

$$\bigcap_{i=1}^{N} V_i = \varnothing$$

Como ejemplo, tomaremos los vectores del caso de dos dimensiones sin pérdida de generalidad. La Figura 3.2 muestra algunos vectores del espacio. Asociado con cada grupo de vectores está una palabra de código representativa. Cada palabra de código reside en su propia Región Voronoi. Estas regiones se encuentran separadas por líneas imaginarias que se ilustran en la Figura 3.2. Dado un vector de entrada, la palabra de código que se ha elegido para representarlo es una de la misma región Voronoi.

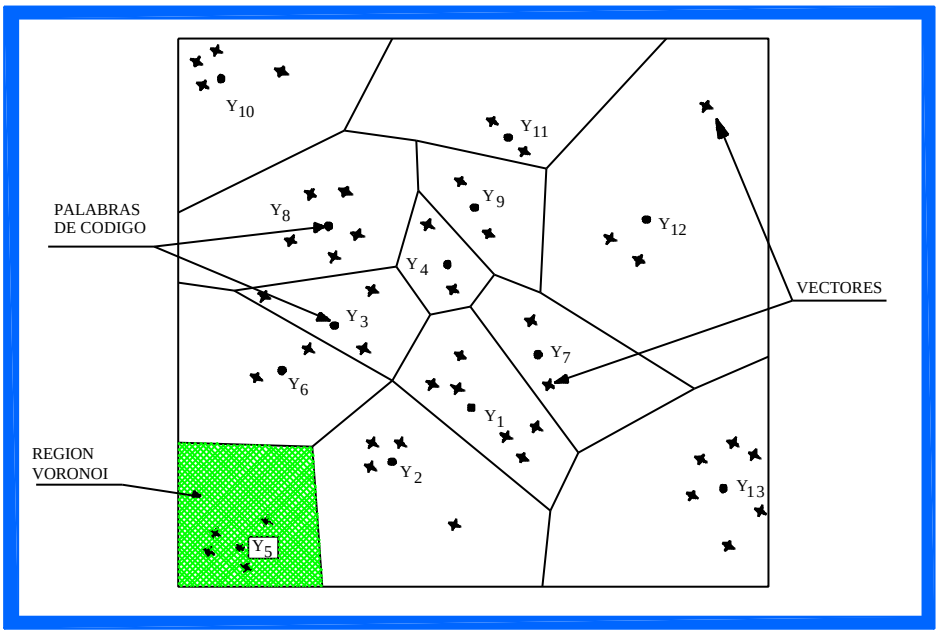


Figura 3.2: Palabras de código en el espacio bidimensional. Los vectores de entrada se marcan con un x, las palabras de código con círculos, y las regiones Voronoi están separadas por líneas frontera.

La palabra de código representativa viene determinada por ser la más próxima en distancia Euclídea al vector de entrada. Se define la distancia Euclídea por:

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k} (x_j - y_{ij})^2} \qquad (3.91)$$

donde x_j es la j-ésima componente del vector de entrada, y y_{ij} es la j-ésima componente de la palabra de código y_i.

Un cuantizador vectorial se compone de dos operaciones. La primera es el codificador, y la segunda es el decodificador. El codificador toma un vector de entrada y da a la salida el índice de la palabra de código que ofrece la distorsión más baja. En este caso, la distorsión más baja se calcula evaluando la distancia Euclídea entre el vector de entrada y cada palabra de código del libro de código[4]. Una vez encontrada la palabra de código más cercana, el índice de esa palabra de código es enviado a través de un canal (el canal podría ser la memoria de un ordenador, canal de comunicaciones,etc). Cuando el codificador recibe el índice de la palabra de código, reemplaza el índice con la palabra asociada. La Figura 3.3 muestra un diagrama de bloques de la operación de codificación y descodificación.

[4] *"Vector Quantization", artículo publicado en la página de referencia http://eewww.eng.ohio-state.edu/~qasemm/vectorquantization/vq.html [13].*

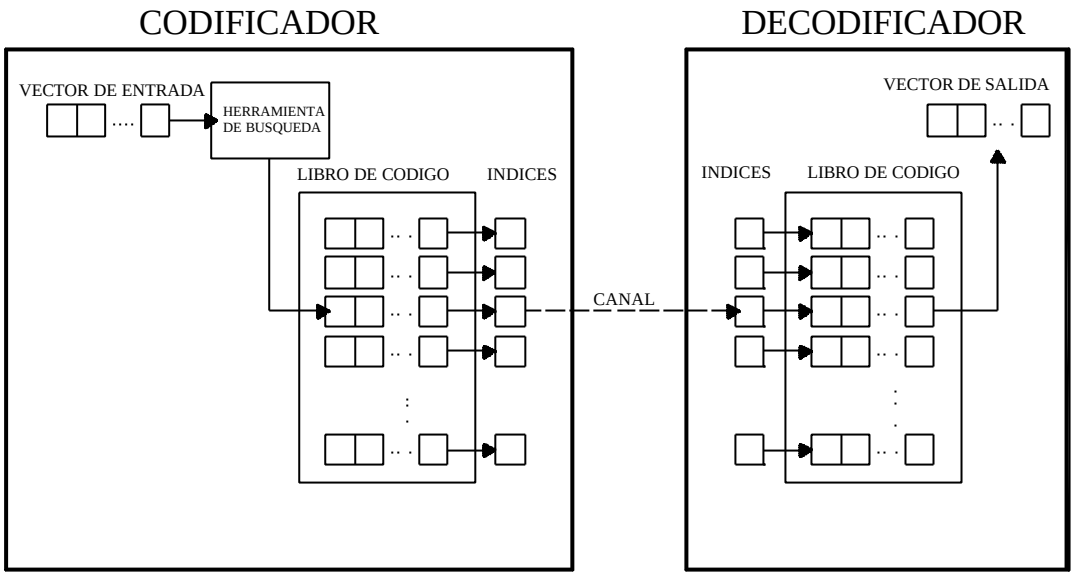


Figura 3.3: El codificador y el decodificador de un Cuantizador Vectorial. Dado un vector de entrada, se encuentra la palabra de código más próxima y se envía el índice de la palabra de código a través de un canal. El decodificador recibe el índice de la palabra de código y da como salida la propia palabra de código.

Hasta ahora hemos hablado del modo en que trabaja el Cuantizador Vectorial, pero no hemos hablado acerca de cómo generar el libro de código. ¿Qué palabras de código representan mejor a un conjunto dado de vectores de entrada? ¿Cuántas deberían de elegirse?

Desafortunadamente, diseñar el libro de código que mejor represente a un conjunto de vectores de entrada es realmente difícil. Esto significa que requiere una búsqueda exhaustiva de las palabras de código mejores posibles del espacio, y la búsqueda se incrementa exponencialmente conforme el número de palabras de código aumenta. Por tanto, se opta por buscar esquemas de diseño de libros de código subóptimos. De entre ellos, el primero que viene a la mente es el más simple. Se conoce por LBG, de Linde-Buzo-Gray, los autores de esta idea. Este algoritmo es similar al algoritmo de media k.

Algoritmo

1. Determinar el número de palabras de código, N, o el tamaño del libro de código.

2. Seleccionar N palabras de código de forma aleatoria, y tómense como libro de código inicial. Las palabras de código iniciales pueden elegirse aleatoriamente de entre el conjunto de vectores de entrada.

3. Usando la medida de la distancia Euclídea, agrupar los vectores cercanos a cada palabra de código. Este paso se realiza tomando cada vector de entrada y encontrando la distancia Euclídea entre éste y cada una de las palabras de código. El vector de entrada pertenece al grupo de la palabra de código que proporciona la mínima distancia.

4. Calcular el nuevo conjunto de palabras de código. Esto se realiza obteniendo el promedio de cada grupo. Sumar las componentes de cada vector y dividir por el número de vectores del grupo.

$$y_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij} \qquad (3.92)$$

donde i es la componente de cada vector (en las direcciones x, y, z, ...), y m es el número de vectores del grupo.

5. Repetir los pasos 2 y 3 hasta que las palabras de código no varíen o bien la variación de las palabras de código sea pequeño.

Este algoritmo es con diferencia el más popular, y esto se debe a su simplicidad. A pesar de ser localmente óptimo, es además muy lento. La razón de esta lentitud es

porque para cada iteración, determinar cada grupo requiere que cada vector de entrada sea comparado con todas las palabras de código del libro de código.

Existen muchos otros métodos para el diseño del libro de código, tales como Pairwise Nearest Neighbor (PNN), Simulated Annealing, Maximun Descent (MD), and Frecuency-Sensitive Competitive Learning (FSCL), etc.

Aunque la cuantización vectorial (VQ) ofrece mayor compresión para la misma tasa de distorsión que la cuantización escalar y la Codificación por Modulación de Pulsos (PCM), aun no está tan ampliamente implementada. Esto se debe a dos cosas. La primera es el tiempo que tarda en generar el libro de código, y la segunda es la velocidad de búsqueda. Se han propuesto muchos algoritmos para incrementar esta velocidad de búsqueda. Algunos de ellos reducen la matemática empleada para determinar la palabra de código que ofrece la mínima distorsión, otros algoritmos preprocesan las palabras de código y explotan la estructura base.

El método de búsqueda más simple, que también es el más lento, es de búsqueda completa. En la búsqueda completa se compara un vector de entrada con cada palabra de código del libro de código. Si hubiera M vectores de entrada, N palabras de código, y cada vector fuera de dimensión k, entonces el número de multiplicaciones resultaría ser kMN, el número de sumas y diferencias $MN((k-1)+k) = MN(2k-1)$, y el número de comparaciones, $MN(k-1)$. Esto convierte la búsqueda completa en un método caro.

¿Cómo marca una tasa el funcionamiento de una imagen o sonido comprimidos usando VQ?. No existe un buen método para medir el funcionamiento de un VQ. Esto se debe a que la distorsión que produce el VQ la evaluaremos los humanos y ésa es una medida subjetiva. No obstante, siempre podemos acudir al error cuadrático medio (Mean Square Error, MSE) y al pico de la relación señal a ruido (Peak Signal to Noise Ratio, PSNR).

MSE se define de la forma:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}\left(\hat{x}_i - x_i\right)^2 \qquad (3.93)$$

donde M es el número de elementos de la señal, o imagen. Por ejemplo, si quisiéramos encontrar la MSE entre la imagen reconstruida y la original, tomaríamos la diferencia entre las dos imágenes pixel a pixel, elevaríamos el resultado al cuadrado, y lo promediaríamos.

La PSNR se define como sigue:

$$PSNR = 10\log_{10}\left(\frac{\left(2^n - 1\right)^2}{MSE}\right) \qquad (3.94)$$

donde N es el número de bits por símbolo. Como ejemplo, si queremos determinar la PSNR entre dos imágenes de nivel Gray 256, haríamos n igual a 8 bits.

A continuación veremos el estudio del algoritmo Linde-Buzo-Gray aplicado a la cuantización vectorial, que ha sido el empleado en el desarrollo de este proyecto.

3.8.1: Algoritmo para Diseño de Cuantizadores Vectoriales

Se trata de desarrollar un algoritmo eficiente e intuitivo para el diseño de cuantizadores vectoriales con bastantes medidas de distorsión general para usar tanto sobre descripciones de fuentes probabilísticas conocidas, como sobre una larga secuencia de datos de entrenamiento. El algoritmo se basa en una aproximación de Lloyd, no es una técnica variacional, y no implica ninguna diferenciación; por tanto, trabaja bien incluso cuando la distribución tiene componentes discretos, como es el caso en el que se usa una distribución de muestras obtenidas de una secuencia de

entrenamiento. Como con las técnicas variacionales comunes, el algoritmo proporciona cuantizadores encontrando condiciones necesarias pero no suficientes para la optimalidad. Normalmente, sin embargo, se asegura al menos optimalidad local en ambas aproximaciones.

Aquí justificamos y describimos el algoritmo, y lo relacionamos con un número de algoritmos similares para casos especiales que han aparecido tanto en la literatura de la cuantización, como en la del análisis de grupo. La operación básica del algoritmo es simple e intuitiva en el caso general que aquí consideramos, y está claro que no se requieren técnicas variacionales para desarrollar o aplicar el algoritmo[5].

Varias de la propiedades básicas del algoritmo se desarrollan usando argumentos heurísticos y se demuestran mediante ejemplo. En estudios teóricos, estas propiedades han dado estamentos matemáticos precisos que se han probado usando argumentos de la teoría de la optimización y de la teoría ergódica. Esos resultados se citan ocasionalmente aquí para caracterizar la generalidad de ciertas propiedades.

En particular, las propiedades de convergencia del algoritmo se demuestran aquí mediante diversos ejemplos. Consideramos caso de prueba habitual para tal diseño de algoritmos-cuantizador para fuentes Gaussianas sin memoria con una medida de la raíz de la media del error de distorsión, pero diseñamos y evaluamos cuantizadores de bloque con una tasa de un bit por símbolo y con longitudes de bloque de 1 hasta 6. La comparación con los límites más bajos hacia la distorsión óptima de dichos cuantizadores de bloques recientemente desarrollados, indica que los cuantizadores resultantes están de hecho cercanos al óptimo y no son simplemente localmente óptimos. También consideramos un caso escalar en el que aparece el óptimo local y muestra cómo conduce una variación del algoritmo al óptimo global.

[5] *"An Algorithm for Vector Quantizer Design". Yoseph Linde, Andrés Buzo, Robert M. Gray, Miembros del IEEE. IEEE Transactions on Communications, Vol. Com-28, NO. 1, Enero, 1980 [2].*

El algoritmo también se usa para diseñar un cuantizador para vectores de 10 dimensiones que surgen en los sistemas de compresión de voz. Se usa una medida de distorsión compleja que no depende simplemente del vector de error. No se asume ningún modelo probabilístico y, por tanto, el cuantizador se debe diseñar basándonos en una secuencia de entrenamiento de voz real. Aquí las propiedades de convergencia para la longitud de la secuencia de entrenamiento y el número de iteraciones del algoritmo se demuestran experimentalmente. No se conoce óptimo teórico alguno para este caso, pero nuestro sistema se empleó para comprimir la salida de un sistema de voz con Codificación Lineal Predictiva (Linear Predictive Coded, LPC) tradicional de 6.000 bit/s a una tasa de 1.400 bit/s, únicamente con una ligera pérdida de calidad tal y como se juzga por oyentes no entrenados en cuestionarios informales subjetivos. Para conocimiento de los autores, la aplicación directa de las técnicas variacionales no han resultado exitosas en el diseño de cuantizadores de bloques para longitudes de bloque tan grandes, ni para medidas de distorsión tan complicadas.

3.8.1.1: Cuantizadores de Bloques

Un cuantizador k-dimensional de nivel N es un mapeado, q, que asigna a cada vector de entrada, $x = (x_0, \ldots, x_{k-1})$, un vector de reproducción, $\hat{x} = q(x)$, extraído de un alfabeto de reproducción finito, $\hat{A} = \{y_i ; i = 1, \ldots, N\}$.. El cuantizador q se describe completamente por el alfabeto de reproducción (o libro de código) $\hat{A}$ junto con la partición, $S = \{S_i ; i = 1, \ldots, N\}$, del espacio de vectores de entrada en los conjuntos $S_i = \{x : q(x) = y_i\}$ de vectores de entrada mapeados sobre el vector de reproducción i-ésimo (o palabra de código). Tales cuantizadores también se denominan cuantizadores de bloques, cuantizadores vectoriales y códigos de bloque fuente.

3.8.1.2: Medidas de Distorsión

Asumimos que la distorsión causada al reproducir un vector de entrada x por un vector de reproducción $\hat{x}$ viene dada por una medida de distorsión no negativa $d(x, \hat{x})$.

Se han propuesto muchas medidas de distorsión de este tipo en la literatura. La más común, por razones de conveniencia matemática, es la distorsión del error cuadrático

$$d(x,\hat{x}) = \sum_{i=0}^{k-1} \left| x_i - \hat{x}_i \right|^2 \tag{3.95}$$

Otras medidas de distorsión comunes son las I_u, o norma Holder,

$$d(x,\hat{x}) = \left\{ \sum_{i=0}^{k-1} \left| x_i - \hat{x}_i \right|^u \right\}^{1/u} \equiv \left\| x - \hat{x} \right\|_u, \tag{3.96}$$

y su potencia u-ésima, la distorsión de ley u-ésima:

$$d(x,\hat{x}) = \sum_{i=0}^{k-1} \left| x_i - \hat{x}_i \right|^u \equiv \left\| x - \hat{x} \right\|_u^u. \tag{3.97}$$

Mientras que, tanto la medida de distorsión (3.96) como la (3.97) dependen de la potencia ν-ésima de los errores de las coordenadas por separado, la medida de (3.96) es a menudo más útil porque es una distancia o métrica y, por tanto, satisface la desigualdad triangular, $d(x,\hat{x}) \le d(x,y) + d(y,\hat{x})$, para todo y. La desigualdad triangular nos permite limitar la distorsión general fácilmente en un sistema multi-paso mediante la suma de las distorsiones individuales incurridas en cada paso. La distorsión

normal de ley u-ésima de (3.97) no tiene esta propiedad. Otras medidas de distorsión son las l_∞ o norma Minkowski,

$$d(x,\hat{x}) = \max_{0\le i\le k-1} |x_i - \hat{x}_i|, \tag{3.98}$$

la distorsión de raíz ponderada,

$$d(x,\hat{x}) = \sum_{i=0}^{k-1} w_i |x_i - \hat{x}_i|^2, \tag{3.99}$$

donde $w_i \ge 0, i = 0, \ldots, k-1,$ y la distorsión cuadrática más general

$$\begin{aligned} d(x,\hat{x}) &= (x-\hat{x})B(x-\hat{x})^t \\ &= \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} B_{i,j}(x_i - \hat{x}_i)(x_j - \hat{x}_j). \end{aligned} \tag{3.100}$$

donde $B = \{B_{i,j}\}$ es una matriz $k \times k$ simétrica definida positiva.

Todas las medidas de distorsión anteriormente descritas tienen la propiedad de que dependen de los vectores x y $\hat{x}$ únicamente a través del vector de error $x - \hat{x}$. Tales medidas de distorsión de la forma $d(x,\hat{x}) = L(x,\hat{x})$ se conocen como medidas de distorsión de diferencia. Las medidas de distorsión que no tienen esta forma pero que dependen de x y $\hat{x}$ de una forma más complicada también se han propuesto para

sistemas de compresión de datos. La de interés aquí es la medida de distorsión de Itakura y Saito y Chaffee que surge en sistemas de compresión de voz y tiene la forma

$$d(x,\hat{x}) = (x-\hat{x})R(x)(x-\hat{x})^t, \qquad (3.101)$$

donde para cada x, $R(x)$ es una matriz $k \times k$ simétrica definida positiva. Esta distorsión recuerda a la distorsión cuadrática de (3.100), pero aquí la matriz de ponderación depende del vector de entrada x.

Hemos tratado aquí la forma y aplicación particulares de esta medida de distorsión más que sus orígenes. Por motivación, sin embargo, describimos ligeramente el contexto en el que se emplea esta medida de distorsión en sistemas de voz. En la aproximación de LPC a compresión de voz, cada trama de voz muestreada se modela como la salida de un filtro todo-polo de orden finito a cuya entrada se tiene bien ruido blanco (sonidos sin voz), o bien un tren de pulsos periódico (sonidos de voz). El análisis LPC tiene, como entrada, una trama de voz y produce parámetros que describen el modelo. Estos parámetros son entonces cuantizados y transmitidos. Una colección de estos parámetros consiste en una decisión de voz/no voz junto con un estimación del pitch para sonidos de voz, un término de ganancia s (relacionado con el volumen), y la respuesta muestreada del filtro inverso normalizado $(1, a_1, a_2, \ldots, a_k)$, es decir, el modelo todo-polo normalizado tiene una función de transferencia o transformada z $\left\{\sum_{k=0}^{K} a_k z^{-k}\right\}^{-1}$. También son posibles otros parámetros de descripción, tales como los coeficientes de reflexión.

En sistemas tradicionales LPC, los diversos parámetros se cuantizan separadamente, pero dichos sistemas han alcanzado efectivamente sus límites teóricos de funcionamiento. Por tanto, es natural considerar cuantización de bloques de estos parámetros y comparar el funcionamiento con las tradicionales técnicas de cuantización

escalar. Aquí consideramos el caso en el que el pitch y la ganancia (como se hace normalmente) se cuantizan separadamente, pero los parámetros que describen el modelo normalizado deben ser cuantizados juntos como un vector. Como el término principal es 1, queremos cuantizar un vector $(a_1, a_2, \ldots, a_K) \equiv x = (x_0, \ldots, x_{K-1})$. Una medida de distorsión, $d(x, \hat{x})$, entre x y una reproducción $\hat{x}$, se puede ver entonces como una medida de distorsión entre dos filtros inversos o modelos normalizados (de ganancia unidad). Una medida de distorsión para tal caso fue propuesta por Itakura y Saito y por Chaffee y tiene la forma de (3.101), siendo $R(x)$ la matriz de autocorrelación $\{r_x(k-j); k = 0,1,\ldots,K-1; j = 0,1,\ldots,K-1\}$ definida por

$$r_x(k) = \int_{-p}^{p} \left| \sum_{m=0}^{K} a_i e^{-imq} \right|^{-2} e^{ikq}\, dq/2p, \qquad (3.102)$$

descrita por x cuando la entrada tiene un espectro plano de amplitud unidad.

Se han desarrollado muchas propiedades y formas alternativas para esta particular medida de distorsión, donde también se muestra que los sistemas estándar LPC implícitamente minimizan esta distorsión, lo cual sugiere que también es una medida de distorsión apropiada para cuantización posterior. Aquí, sin embargo, el hecho importante es que no es una medida de distorsión diferente; es una para la cual la dependencia en x y $\hat{x}$ es bastante complicada.

También observamos que se han propuesto en la literatura varias funciones de las medidas de distorsión previamente definidas, por ejemplo, medidas de distorsión de la forma $\|x - \hat{x}\|^r$ y $r(\|x - \hat{x}\|)$, donde r es una función convexa y la norma es cualquiera de las normas definidas anteriormente. Las técnicas a desarrollar aquí son aplicables a todas estas medidas de distorsión.

3.8.1.3: Funcionamiento

Sea $X = (X_0, \ldots, X_{k-1})$ un vector aleatorio real descrito por una función de distribución cumulativa $F(x) = \Pr\{X_i \leq x_i; i = 0,1, \ldots, k-1\}$.. Una medida del funcionamiento de un cuantizador q aplicado sobre el vector aleatorio X viene dada por la distorsión esperada

$$D(q) = Ed(X, q(X)), \tag{3.103}$$

donde E denota la esperanza con respecto a la distribución fundamental F. Esta medida de funcionamiento está físicamente llena de significado si el cuantizador q se va a usar para cuantizar una secuencia de vectores $X_n = (X_{nK}, \ldots, X_{nK+K-1})$ estacionarios y ergódicos, porque entonces la distorsión promediada en tiempo,

$$n^{-1} \sum_{i=0}^{n-1} d(X_i, q(X_i)), \tag{3.104}$$

converge con probabilidad uno a $D(q)$ conforme $n \to \infty$ (por el Teorema Ergódico), es decir, $D(q)$ describe la distorsión promediada en tiempo a largo plazo.

Una medida alternativa del funcionamiento es el máximo de $d(x, q(x))$ para todo x en A pero nosotros usamos únicamente la distorsión esperada (3.103) porque, en la mayoría de problemas de interés (para nosotros), es la distorsión promedio, y no la distorsión máxima o de pico, la que determina la calidad subjetiva. Además, la distorsión esperada se trata más fácilmente de forma matemática.

3.8.1.4: Cuantización Óptima

Se dice que un cuantizador de nivel N es óptimo (o globalmente óptimo) si minimiza la distorsión esperada, es decir, q^* es óptimo si, para todos los demás cuantizadores q que tengan N vectores de reproducción, $D(q^*) \leq D(q)$. Se dice que un cuantizador es localmente óptimo si $D(q)$ es sólo un mínimo local, es decir, ligeros cambios en q provocan un incremento en la distorsión. El objetivo del diseño de un cuantizador de bloques es obtener un cuantizador óptimo siempre que sea posible, y si no, obtener un cuantizador localmente óptimo y prometedoramente? "bueno". Se han propuesto varios de tales algoritmos en la literatura para diseño asistido por ordenador de cuantizadores localmente óptimos. En pocos casos especiales, ha sido posible demostrar la optimalidad global, bien analíticamente, o bien agotando todos los óptimos locales. En 1957, en un artículo de los Laboratorios Bell, clásico pero desafortunadamente no publicado, S. Lloyd propuso dos métodos para el diseño de cuantizadores para el caso escalar ($k = 1$) con un criterio de distorsión de la raíz del error. Su Método II era una sencilla aproximación variacional en la que tomó derivadas con respecto a los símbolos de reproducción, y_i, y con respecto a los puntos frontera que definen la S_i e igualó estas derivadas a cero. Esto, en general, conduce únicamente a un cuantizador "de punto estacionario" (una derivación multidimensional cero) que satisface necesaria pero no suficientemente las condiciones para la optimalidad. A partir de los argumentos de la segunda derivada, sin embargo, es fácil establecer que dichos cuantizadores de punto-estacionario son al menos localmente óptimos para las medidas de distorsión de la ley de la ν-ésima potencia. Además, Lloyd también demostró la optimalidad global para ciertas distribuciones mediante una técnica de búsqueda exhaustiva de todos los óptimos locales. Esencialmente, se propuso y usó también la misma técnica en el problema paralelo de análisis de grupo, por Dalenius en 1950, Fisher en 1953, y Cox en 1957. La técnica también fue desarrollada independientemente por Max en 1960, y el cuantizador resultante se conoce comúnmente como el cuantizador Lloyd-Max. Esta aproximación resultó bastante útil para el diseño de cuantizadores escalares, con criterios de distorsión de leyes de potencia y con

distribuciones conocidas que tenían un comportamiento lo suficientemente bueno como para asegurar la existencia de las derivadas en cuestión. Además, para este caso, Fleischer pudo demostrar analíticamente que los cuantizadores resultantes eran globalmente óptimos para diversas densidades de probabilidad interesantes.

En algunas situaciones, sin embargo, la aproximación variacional directa no se ha probado con éxito. En primer lugar, si k no es igual a 1 ó 2, los requerimientos computacionales se tornan demasiado complejos. Las combinaciones simples de diferenciación unidimensional no funcionarán debido a los bordes, posiblemente complicados, de las superficies frontera de las celdas de la partición. De hecho, las únicas aplicaciones con éxito, en una aproximación variacional directa a cuantización multidimensional, son para los cuantizadores en los que se requiere que las celdas de partición tengan una forma particular simple, tal como "cubos" multidimensionales o, en dos dimensiones, "trozos de tarta", cada uno de ellos descrito por un radio y dos ángulos. Estos bordes son sensibles para técnicas de diferenciación, pero únicamente conducen a un óptimo local dentro de la clase restringida/específica. En segundo lugar, si además se desea obtener medidas de distorsión más complejas, tales como aquellas de (3.98)-(3.101), la carga computacional asociada a las ecuaciones variacionales puede ser exorbitante. En tercer lugar, si la distribución de probabilidad base tiene componentes discretos, entonces las derivadas requeridas pueden no existir, provocando más problemas computacionales. Por último, si uno carece de una precisa descripción probabilística del vector aleatorio X y debe basar el diseño, en su lugar, en largas secuencias de datos de entrenamiento obsevados, entonces no hay un camino obvio para aplicar la aproximación variacional. Si el proceso base desconocido es estacionario y ergódico, entonces positivamente un sistema diseñado usando una secuencia de entrenamiento suficientemente larga debería también funcionar bien sobre datos futuros. Para aplicar directamente la técnica variacional en este caso, uno debería tener que estimar, en primer lugar, la distribución continua fundamental basada en las observaciones, y después tomar las derivadas apropiadas. Desafortunadamente, sin embargo, la mayoría de las técnicas estadísticas para estimación de la densidad requieren una suposición inicial sobre la clase de densidades permitidas, por ejemplo,

familias exponenciales. Por tanto, estas técnicas no son apropiadas cuando este conocimiento no está disponible. Además, un buen ajuste de un modelo continuo a un histograma de muestras finitas puede tener un mal comportamiento diferencial y, por tanto, puede no producir un buen cuantizador. Para nuestro conocimiento, nadie ha usado una aproximación tal con éxito, ni ha demostrado que esta aproximación conducirá al cuantizador correcto en el límite de una larga secuencia de entrenamiento.

Lloyd también propuso una aproximación no variacional alternativa como su "Método 1". De forma nada sorprendente, ambas aproximaciones conducen al mismo cuantizador para los casos especiales que él consideró, pero discutiremos que una extensión natural e intuitiva de su Método 1 proporciona un eficiente algoritmo para el diseño de buenos cuantizadores vectoriales que superan los problemas de la aproximación variacional. De hecho, se han "descubierto" en diversas ocasiones en la literatura variaciones sobre el Método 1 de Lloyd, para los criterios de distorsión de la raíz del error y la magnitud del error, tanto para el caso escalar como para el multidimensional. El desarrollo básico de Lloyd, sin embargo, que continúa siendo el más simple, aún se extiende fácilmente al caso general considerado aquí.

Para describir el Método 1 de Lloyd en el caso general, asumimos, en primer lugar, que la distribución es conocida, pero permitimos que sea continua o discreta y no hacemos asumpciones que requieran la existencia de derivadas. Dado un cuantizador q, descrito por un alfabeto de reproducción $\hat{A} = \{y_i; i = 1, \ldots, N\}$ y una partición $S = \{S_i; i = 1, \ldots, N\}$, la distorsión esperada $D(\{\hat{A}, S\}) \equiv D(q)$ se puede escribir como

$$\begin{aligned} D(\{\hat{A}, S\}) &= E[d(X, q(X))] = \int_x d(x, q(x)) f(x) \\ &= \sum_{i=1}^{N} E(d(X, y_i) | X \in S_i) \Pr(X \in S_i) \end{aligned} \tag{3.105}$$

donde $E\left(d\left(X, y_i\right) \middle| X \in S_i\right)$ es la distorsión condicional esperada, dado que $X \in S_i$ ó, equivalentemente, dado que $q\left(X\right) = y_i$.

Supóngase que nos dan un alfabeto de reproducción $\hat{A}$, pero no se especifica una partición. Una partición que es óptima para $\hat{A}$ se construye fácilmente mapeando cada x en el $y_i \in \hat{A}$ que minimiza la distorsión $d\left(x, y_i\right)$, es decir, eligiendo la mínima distorsión o la palabra de código vecina más cercana para cada entrada. Se requiere una regla de desempate (tie-breaking) tal como elegir la reproducción con índice más bajo si más de una palabra de código minimiza la distorsión. La partición, $P\left(\hat{A}\right) = \left\{P_i; i = 1, \ldots, N\right\}$ construida de este modo es tal que $X \in P_i$ (o $q\left(x\right) = y_i$) sólo si $d\left(x, y_i\right) \leq d\left(x, y_j\right)$, para todo j, y por tanto

$$D\left(\left\{\hat{A}, P\left(\hat{A}\right)\right\}\right) = E\left(\min_{y \in \hat{A}} d\left(X, y\right)\right) \tag{3.106}$$

lo que, a su vez, implica para cualquier partición S que

$$D\left(\left\{\hat{A}, S\right\}\right) \geq D\left(\left\{\hat{A}, P\left(\hat{A}\right)\right\}\right) \tag{3.107}$$

Por tanto, para un alfabeto de reproducción fijo $\hat{A}$, la partición mejor posible es $P\left(\hat{A}\right)$.

A la inversa, asúmase que se nos da una partición $S = \left\{S_i; i = 1, \ldots, N\right\}$, que describe a un cuantizador. Por el momento, asúmase también que la medida de distorsión y la distribución son tales que, para cada conjunto S con probabilidad

distinta de cero en el espacio Euclídeo k-dimensional, existe un vector de distorsión mínima $\hat{x}(S)$ para el cual

$$E\left(d\left(X,\hat{x}(S)\right)\middle|X\in S\right)=\min_{u}E\left(d\left(X,u\right)\middle|X\in S\right). \tag{3.108}$$

Análogamente al caso de la medida de distorsión de la raíz del error y distribución de probabilidad uniforme, llamamos al vector $\hat{x}(S)$ centroide o centro de gravedad del conjunto S. Si existe este punto, entonces claramente para una partición fija $S=\{S_i; i=1,\ldots,N\}$, ningún alfabeto de reproducción $\hat{A}=\{y_i; i=1,\ldots,N\}$ puede conducir a una distorsión promedio más pequeña que el alfabeto de reproducción $\hat{x}(S)\equiv\{\hat{x}(S_i); i=1,\ldots,N\}$ que contiene los centroides del conjunto de S dado que

$$\begin{aligned} D\left(\left\{\hat{A},S\right\}\right)&=\sum_{i=1}^{N}E\left(d\left(X,y_i\right)\middle|X\in S_i\right)\Pr\left(X\in S_i\right)\\ &\geq\sum_{i=1}^{N}\min_{u}E\left(d\left(X,u\right)\middle|X\in S_i\right)\Pr\left(X\in S_i\right)\\ &=D\left(\left\{\hat{x}(S),S\right\}\right). \end{aligned} \tag{3.109}$$

Se puede demostrar que los centroides de (3.108) existen para todos los conjuntos de S con probabilidad distinta de cero para medidas de distorsión bastante generales, incluyendo todas aquellas consideradas aquí. En particular, si $d(x,y)$ es convexa en y, entonces los centroides se pueden computar usando técnicas de programación estándar convexas. En ciertos casos, se pueden encontrar fácilmente usando técnicas variacionales. Si la probabilidad de un conjunto S es cero, entonces el centroide se

puede definir de una forma arbitraria, dado que entonces la esperanza condicional dada esa S en (3.108) no tiene una única definición.

Las ecuaciones (3.107) y (3.109) sugieren un algoritmo natural para diseñar un buen cuantizador tomando cualquier cuantizador dado y mejorándolo iterativamente:

Algoritmo (Distribución Conocida)

(0) Inicio: Dados N = número de niveles, un umbral de distorsión $e \geq 0,$ y un alfabeto de reproducción de nivel N inicial $\hat{A}_0$ y una distribución F. Hacer $m = 0$ y $D_{-1} = \infty.$

(1) Dado $\hat{A}_m = \{y_i; i = 1, \ldots, N\}$ encontrar su partición de distorsión mínima $P(\hat{A}_m) = \{S_i; i = 1, \ldots, N\}$: $x \in S_i$ si $d(x, y_i) \leq d(x, y_j)$ para todo $j.$ Calcular la distorsión promedio resultante, $D_m = D(\{\hat{A}_m, P(\hat{A}_m)\}) = E\, min_{y \in \hat{A}_m} d(X, y).$

(2) Si $(D_{m-1} - D_m)/D_m \leq e,$ parar con $\hat{A}_m$ y $P(\hat{A}_m)$ describiendo el cuantizador final. En otro caso, continuar.

(3) Encontrar el alfabeto de reproducción óptimo $\hat{x}(P(\hat{A}_m)) = \{\hat{x}(S_i); i = 1, \ldots, N\}$ para $P(\hat{A}_m).$ Hacer $\hat{A}_{m+1} \equiv \hat{x}(P(\hat{A}_m)).$. Reemplazar m por $m+1$ e ir a (1).

Si, en algún punto, la partición óptima $P(\hat{A}_m)$ tiene una celda S_i tal que $\Pr(X \in S_i) = 0,$ entonces el algoritmo, tal como se expuso, asigna un vector arbitrario como centroide y continua (Solución si hay una celda de valor cero). Claramente son posibles reglas alternativas y pueden funcionar mejor en la práctica. Por ejemplo, uno puede simplemente eliminar la celda S_i y el correspondiente símbolo de reproducción del cuantizador sin afectar al funcionamiento, y después continuar con un cuantizador de nivel $(N-1).$ Alternativamente, se podría asignar a S_i su centro de gravedad Euclídeo o el centroide i-ésimo de la iteración previa. También se podría simplemente

reasignar el vector de reproducción correspondiente a S_i a otra celda S_j y continuar el algoritmo. La técnica establecida, se da simplemente por conveniencia, porque las celdas con probabilidad cero no fueron un problema en los ejemplos aquí considerados. Pueden, sin embargo, ocurrir y, en tales situaciones, las técnicas alternativas como las descritas pueden funcionar mejor. En la práctica, una alternativa simple es que, si el cuantizador final producido por el algoritmo tiene una celda con probabilidad cero (por tanto, sin utilidad), simplemente se rearranque/reactive el algoritmo con una condición inicial diferente.

De (3.107) y (3.109), $D_m \leq D_{m-1}$ y por tanto cada iteración del algoritmo debe reducir la distorsión o bien mantenerla inalterada. Mencionaremos más tarde algunos detalles adicionales menores del algoritmo y discutiremos técnicas para elegir la condición inicial, pero la explicación dada previamente contiene las ideas esenciales.

Como D_m es no creciente y no negativa, debe de tener un límite, llamado D_∞, conforme $m \to \infty$. En diversos estudios se muestra que si existe un cuantizador limitante $\hat{A}_\infty$ en el sentido de que $\hat{A}_m \to \hat{A}_\infty$ conforme $m \to \infty$ en el sentido Euclídeo usual, entonces $D\left(\left\{\hat{A}_\infty, P\left(\hat{A}_\infty\right)\right\}\right) = D_\infty$ y $\hat{A}_\infty$ tiene la propiedad de que $\hat{A}_\infty = \hat{x}\left(P\left(\hat{A}_\infty\right)\right)$, es decir, $\hat{A}_\infty$ es exactamente el centroide de su propia partición óptima. En el lenguaje de la teoría de la optimización, $\left\{\hat{A}_\infty, P\left(\hat{A}_\infty\right)\right\}$ es un punto fijo frente a más iteraciones del algoritmo. Por tanto, el cuantizador límite (si existe) se conoce por Cuantizador de Punto-Fijo (en contraste/contraposición con un Cuantizador de Punto-Estacionario obtenido mediante una aproximación variacional). En este sentido, el algoritmo es simplemente una técnica estándar para encontrar un punto fijo a través del método de Aproximaciones Sucesivas. Si $e = 0$ y el algoritmo se detiene para m finito, entonces se ha llegado a dicho punto fijo.

Una condición necesaria para que un cuantizador sea óptimo es que sea un cuantizador de punto-fijo. También se demuestra que, como en el caso de Lloyd, si un cuantizador de punto-fijo es tal que no hay probabilidad en la frontera de las celdas de

partición, es decir si $\Pr(d(X, y_i) = d(X, y_i),$ para algún $i \neq j) = 0,$ entonces el cuantizador es localmente óptimo. Este es siempre el caso con distribuciones continuas, pero puede, en principio ser violado para distribuciones discretas. Sin embargo, nunca se dio el caso en nuestros experimentos. Como Lloyd sugiere, el algoritmo puede modificarse fácilmente para probar un punto fijo para esta condición y si hay probabilidad distinta de cero de un vector sobre una frontera, la estrategia sería reasignar el vector a otra celda de la partición y continuar la iteración.

Para el caso $N = 1$ con un criterio de distorsión de la raíz del error, el algoritmo es simplemente el Método 1 de Lloyd, y sus argumentos se aplican inmediatamente en el caso más general aquí considerado. Una técnica similar fue propuesta anteriormente en 1953 por Fisher en un problema de análisis de grupo usando decisiones de Bayes con un coste de la raíz del error. Para dimensiones mayores y medidas de distorsión de la forma $d(x, \hat{x}) = \|x - \hat{x}\|_2^r, r \geq 1,$ la relaciones (3.107) y (3.109) fueron observadas por Zador y Gersho en su trabajo sobre el comportamiento asintótico de cuantizadores óptimos, y por tanto el algoritmo está realmente implícito en su trabajo. Sin embargo, ellos en realidad no propusieron o aplicaron la técnica para diseñar un cuantizador para N fijo. En 1965, Forgy propuso el algoritmo para análisis de grupo para el caso multidimensional de distorsión la raíz del error y una distribución de muestras. En 1977 Chen propuso esencialmente el mismo algoritmo para el caso multidimensional con la medida de distorsión de la raíz del error, y lo usó para diseñar cuantizadores bidimensionales para vectores uniformemente distribuídos en un círculo.

Dado que el algoritmo no tiene diferenciabilidad de requerimientos, es válido para distribuciones puramente discretas. Esto tienen una importante aplicación al caso en el que uno no posee ‘a priori’ una descripción probabilística de la fuente a comprimir, y por tanto, debe basar su diseño en una larga secuencia de entrenamiento observada de los datos a comprimir. Una aproximación sería usar las técnicas de estadística estándar de estimación de la densidad para obtener una “suave” distribución de F y luego aplicar técnicas variacionales. Como se discutió previamente, no adoptamos esta aproximación porque requiere asumpciones adicionales sobre las densidades permitidas.

En lugar de esto, consideramos la siguiente aproximación: Usar la secuencia de entrenamiento, $\{x_k; k = 0, \ldots, n-1\}$ para formar la distorsión promediada en tiempo

$$\frac{1}{n}\sum_{i=0}^{n-1} d(x_i, q(x_i)) \qquad (3.110)$$

y observar que esta es exactamente la distorsión esperada $E_{G_n} d(X, q(X))$ con respecto a la distribución de muestras G_n determinada por la secuencia de entrenamiento, es decir, la distribución que asigna probabilidad m/n a un vector x que aparece en la secuencia de entrenamiento m veces. Por tanto, podemos diseñar un cuantizador que minimice la distorsión promediada en tiempo para la secuencia de entrenamiento ejecutando el algoritmo sobre la distribución de muestras $G_n^{(1,2)}$. Esto conduce a la siguiente variación del algoritmo:

Algoritmo (Distribución Desconocida)

(0) Inicio: Dados $N =$ número de niveles, un umbral de distorsión $e \geq 0$, y un alfabeto de reproducción de nivel N inicial $\hat{A}_0$, y una secuencia de entrenamiento $\{x_j; j = 0, \ldots, n-1\}$. Hacer $m = 0$ y $D_{-1} = \infty$.

(1) Dado $\hat{A}_m = \{y_i; i = 1, \ldots, N\}$, encontrar la partición de distorsión mínima $P(\hat{A}_m) = \{S_i; i = 1, \ldots, N\}$ de la secuencia de entrenamiento: $x_j \in S_i$ si $d(x_j, y_i) \leq d(x_j, y_l)$, para todo l. Calcular la distorsión promedio

$$D_m = D(\{\hat{A}_m, P(\hat{A}_m)\}) = n^{-1}\sum_{j=0}^{n-1} \min_{y \in \hat{A}_m} d(x_j, y). \qquad (3.111)$$

(2) Si $(D_{m-1} - D_m)/D_m \leq e$, parar con el alfabeto de reproducción final $\hat{A}_m$. En otro caso, continuar.

(3) Encontrar el alfabeto de reproducción óptimo $\hat{x}(P(\hat{A}_m)) = \{\hat{x}(S_i); i = 1, \ldots, N\}$ para $P(\hat{A}_m)$. Hacer $\hat{A}_{m+1} \equiv \hat{x}(P(\hat{A}_m))$. Reemplazar m por $m+1$ e ir a (1).

Obsérvese arriba que mientras se diseña el cuantizador, únicamente se consideran particiones de la secuencia de entrenamiento (el alfabeto de entrada). Una vez que se obtiene el libro de código final $\hat{A}_m$, sin embargo, éste se usa sobre nuevos datos fuera de la secuencia de entrenamiento con la regla óptima del vecino más cercano, esto es, una partición óptima del espacio Euclídeo k-dimensional.

Si la secuencia de vectores aleatorios es estacionario y ergódica, entonces del Teorema Ergódico se sigue que, con probabilidad unidad, G_n se acerca a la verdadera distribución base F conforme $n \to \infty$. Por tanto, si la secuencia de entrenamiento es suficientemente larga, es de esperar que un buen cuantizador para la distribución de muestras G_n debería de ser también bueno para la verdadera distribución F, y por tanto deberá conducir a un buen comportamiento para datos futuros producidos por la fuente. Se demuestra que, sujeto a adaptar asumpciones matemáticas, el cuantizador obtenido al aplicar el algoritmo a G_n converge, conforme $n \to \infty$, al cuantizador obtenido al aplicar el algoritmo a la verdadera distribución base F. Observamos que los autores no conocen resultados análogos para la aproximación de estimación/variacional de la densidad, y que no se requiere la independencia de los sucesivos bloques para estos resultados, únicamente estacionariedad y ergodicidad de bloque.

También hemos extraído que, para distribuciones de alfabeto finitas tales como distribuciones de muestras, el algoritmo siempre converge a un cuantizador de punto-fijo en un número finito de pasos.

Una técnica similar usada en el análisis de grupo con funciones de coste de la raíz del error, fue desarrollada por MacQueen 1967 y es conocida como la aproximación de

medias k. Una técnica más involucrada que usa la aproximación de medias k es la aproximación "ISODATA" de Ball y Hall. La idea básica de encontrar particiones de distorsión mínima y centroides es la misma, los datos de la secuencia de entrenamiento se usan de forma diferente y los cuantizadores resultantes serán, en general, diferentes. Su técnica secuencial incorpora los vectores de entrenamiento uno en un cierto momento, y finaliza cuando se ha incorporado el último vector. Esto contrasta con el algoritmo previo que considera todos los vectores de entrenamiento en cada iteración. El método de las medias k se puede describir como sigue: La meta es producir una partición $S_0 = \{S_0, \ldots, S_{N-1}\}$ del alfabeto de entrenamiento, $A = \{x_i ; i = 0, \ldots, n-1\}$ consistente en todos los vectores de la secuencia de entrenamiento. El alfabeto de reproducción correspondiente $\hat{A}$ será entonces la colección de los centroides Euclídeos de los conjuntos S_i, es decir, el alfabeto de reproducción final será óptimo para la partición final (pero la partición final puede no ser óptima para el alfabeto de reproducción final, excepto cuando $n \to \infty$). Para obtener S, pensamos primero en cada uno de los S_i como un recipiente (bin) en el que ubicar los vectores de la secuencia de entrenamiento hasta que todos estén situados. Inicialmente, comenzamos situando los primeros N vectores en recipientes o regiones (bins) separadas, es decir, $x_i \in S_i, i = 0, \ldots, n-1$. Entonces procedemos como sigue: en cada iteración, se observa un nuevo vector de entrenamiento x_m. Encontramos el conjunto S_i para el cual la distorsión entre x_m y el centroide $\hat{x}(S_i)$ se minimiza y entonces añadimos x_m a este recipiente. Por tanto, en cada iteración, el nuevo vector se añade al recipiente con el centroide más próximo, y por tanto la siguiente vez, este recipiente tendrá un nuevo centroide. Esta operación se continúa hasta que todos los vectores de muestras son incorporados.

Aunque similar en filosofía, el algoritmo de medias k tiene algunas diferencias cruciales. En particular, se ajusta para el caso en el que sólo la secuencia de entrenamiento debe ser clasificada, esto es, donde una larga secuencia de vectores debe ser agrupada en una forma de baja distorsión. El proceso secuencial es computacionalmente eficiente para la agrupación, pero un "cuantizador" no se produce

hasta que el proceso se para. En otras palabras, en el análisis de grupo, uno desea agrupar cosas y los grupos pueden cambiar con el tiempo, pero en cuantización, uno desea fijar los grupos (obtener un cuantizador invariante en el tiempo), y luego usar estos grupos (o el cuantizador) sobre datos futuros fuera de la secuencia de entrenamiento.

Un problema adicional es que los únicos teoremas que garantizan la convergencia, en el límite de una larga secuencia de entrenamiento, requieren la suposición de que los sucesivos vectores son independientes, a diferencia del caso más general para el algoritmo propuesto.

Recientemente Levenson usó una variación de los algoritmos de medias k e ISODATA con una medida de distorsión propuesta por Itakura para determinar patrones de referencia para reconocimiento de palabras de locutores independientes. Ellos usaron, como medida de distorsión, el logaritmo de la distorsión de (3.101) (que es una distorsión de Itakura-Saito de ganancia optimizada –nuestro uso de la medida de distorsión con modelos normalizados de ganancia unidad no dieron como resultado tales funciones logarítmicas). En su técnica, sin embargo, se usó una regla minimax para seleccionar los vectores de reproducción (o puntos de grupo) más que para encontrar el vector centroide "óptimo". Si en su lugar, se usa la medida de distorsión (3.101), entonces los centroides se encuentran fácilmente, como veremos.

3.8.1.5: Elección de $\hat{A}_0$

Hay diversas formas de elegir el alfabeto inicial de reproducción $\hat{A}_0$ requerido por el algoritmo. Un método para usar sobre distribuciones de muestras es aquél del método de medias k, a saber la elección de los primeros N vectores de la secuencia de entrenamiento. Nosotros no tratamos esta aproximación como, intuitivamente, a uno le gustaría que estos vectores fueran bien separados, y N muestras consecutivas pueden no ser. Otros dos métodos resultaron ser útiles en nuestros ejemplos. El primero es usar un cuantizador uniforme para todo o la mayoría del alfabeto fuente (si está limitado). Por ejemplo, si es usado en una distribución de muestras, uno usa un cuantizador

uniforme k-dimensional sobre un cubo Euclídeo k-dimensional que incluya todos o la mayoría de los puntos de la secuencia de entrenamiento. Esta técnica se usó en los ejemplos Gaussianos descritos más adelante.

La segunda técnica es útil cuando uno desea diseñar cuantizadores de tasas sucesivamente más altas hasta obtener un nivel de distorsión aceptable. Aquí consideramos cuantizadores de nivel M con $M = 2^R, R = 0,1, \ldots,$ y continúa hasta que obtenemos un condición inicial para un cuantizador de nivel N como sigue:

3.8.1.6: Condición Inicial por "Splitting"

(0) Inicio: Hacer $M = 1$ y definir $\hat{A}_0(1) = \hat{x}(A),$ el centroide del alfabeto completo (el centroide de la secuencia de entrenamiento si se usa una distribución de muestras)

(1) Dado el alfabeto de reproducción $\hat{A}_0(M)$ conteniendo M vectores $\{y_i; i = 1, \ldots, M\},$ "esplitear" cada vector y_i en dos vectores próximos $y_i + e$ e $y_i - e,$ donde e es un vector de perturbación fijo. El conjunto $\tilde{A}$ de $\{y_i + e, y_i - e, i = 1, \ldots, M\}$ tiene $2M$ vectores. Reemplazar M por $2M$.

(2) ¿Es $M = N$? Si es así, hacer $\hat{A}_0 = \tilde{A}(M)$ y parar. $\tilde{A}_0$ es entonces el alfabeto de reproducción inicial para el algoritmo de cuantización de nivel N. Si no, ejecutar el algoritmo para un cuantizador de nivel M sobre $\tilde{A}(M)$ para producir un buen alfabeto de reproducción $\hat{A}_0(M),$ y después volver al paso (1).

Usando el algoritmo de "splitting" sobre una secuencia de entrenamiento, uno empieza con un cuantizador de nivel uno que consiste en el centroide de la secuencia de entrenamiento. Este vector se esplitea entonces en dos vectores y el algoritmo del cuantizador de nivel 2 se ejecuta sobre esta pareja para obtener un buen cuantizador de nivel 2 (de punto fijo). Cada uno de estos dos vectores se esplitea entonces y el

algoritmo se ejecuta para producir un buen cuantizador de nivel cuatro. En la conclusión, uno tiene cuantizadores de punto fijo de niveles $1, 2, 4, 8, \ldots, N$.

3.8.1.7: Ejemplos

Fuentes Gaussianas

El algoritmo se usó inicialmente para diseñar cuantizadores para el ejemplo clásico de variables aleatorias Gaussianas sin memoria con el criterio de distorsión del error cuadrático de (3.95), basados en una secuencia de datos de entrenamiento. Los datos de entrenamiento y muestras fueron producidos por una secuencia de variables aleatorias Gaussianas sin memoria de media cero y varianza unidad. La condición inicial era un cuantizador de nivel uno sobre el cubo k-dimensional, $\{x : |x_i| \le 4; i = 0, \ldots, k-1\}$. Se usó un umbral de distorsión de 0.1%. El algoritmo general se puede describir como sigue:

(0) Inicio: Fijar $N =$ número de niveles, $k =$ longitud de bloque, $n =$ longitud de la secuencia de entrenamiento, $e = 0.001$. Dada una secuencia de entrenamiento $\{x_j; j = 0, \ldots, n-1\}$. Sea $\hat{A}_0$ un alfabeto de reproducción de cuantizador uniforme de nivel N para el cubo k-dimensional, $\{u : |u_i| \le 4; i = 0, \ldots, k-1\}$. Hacer $m = 0$ y $D_{-1} = \infty$.

(1) Dado $\hat{A}_m = \{y_i; i = 1, \ldots, N\}$, encontrar la partición de distorsión mínima $P(\hat{A}_m) = \{S_i; i = 1, \ldots, N\}$. Por ejemplo, para cada $j = 0, \ldots, n-1$, computar $d(x_j, y_i)$ para $i = 1, \ldots, N$. Si $d(x_j, y_i) \le d(x_j, y_l)$ para todo l, entonces $x_j \in S_i$. Calcular:

$$D_m = D(\{\hat{A}_m, P(\hat{A}_m)\}) = n^{-1} \sum_{j=0}^{n-1} \min_{y \in A_m} d(x_j, y). \qquad (3.112)$$

(2) Si $(D_{m-1} - D_m)/D_m \le e = .001,$ parar con el cuantizador final descrito por $\hat{A}_m$. En otro caso, continuar.

(3) Encontrar el alfabeto de reproducción óptimo $\hat{x}\left(P\left(\hat{A}_m\right)\right) = \{\hat{x}(S_i); i = 1, \ldots, N\}$ para $P\left(\hat{A}_m\right)$. Para el criterio de la raíz del error, $\hat{x}(S_i)$ es el centro de gravedad Euclídeo o el centroide dado por

$$x(S_i) = \frac{1}{\|S_i\|} \sum_{j: x_j \in S_i} x_j, \qquad (3.113)$$

donde $\|S_i\|$ denota el número de vectores de entrenamiento de la celda S_i. Si $\|S_i\| = 0,$ hacer $\hat{x}(S_i) = y_i,$ la antigua palabra de código. Definir $\hat{A}_{m+1} = \hat{x}(P(A_m)),$ reemplazar m por $m+1,$ y volver a (1).

Un cuantizador de dos dimensiones con cuatro niveles se diseña, basándonos en una corta secuencia de entrenamiento de doce vectores de entrenamiento. Debido a la corta secuencia de entrenamiento en este caso, la distorsión final es más baja que la que uno esperaría y el cuantizador final puede no trabajar bien sobre nuevos datos fuera de la secuencia de entrenamiento. Las compensaciones entre la longitud de la secuencia de entrenamiento y el funcionamiento dentro y fuera de la secuencia de entrenamiento se desarrollan más cuidadosamente en el ejemplo de voz.

Si el cuantizador permanece igual para dos iteraciones, entonces las dos distorsiones son iguales y se satisface un umbral "ε = 0".

Una comparación más favorable se obtiene usando un reciente resultado de Yamada, Tazaki, y Gray que proporciona un límite más bajo para el funcionamiento de un cuantizador k-dimensional de nivel N óptimo con una diferencia en la medida de distorsión cuando N es grande. Este límite proporciona una estricta mejora sobre la tasa de distorsión límite para k fijo y tiende a la misma conforme $k \to \infty.$ En el caso que tratamos, el límite tiene la forma

$$D_Q^{(k)}(R) = D(R) \bullet \left\{ \left(\frac{e}{1+k/2} \right) \Gamma(1+k/2)^{2/k} \right\}, \qquad (3.114)$$

donde Γ es la función gamma. Este límite es teóricamente inapropiado para k pequeño, y sin embargo está soprendentemente cerca del resultado para $k = 1,$ que se conoce por ser casi el óptimo. Para $k = 6,$ $N = 2^6 = 64$ es moderadamente grande y la cercanía del funcionamiento actual al límite más bajo, comparado con el funcionamiento proporcionado por $D_Q^{(k)}(1),$ sugiere que el algoritmo está proporcionando de hecho un cuantizador con longitud de bloque 6 y tasa de un bit por símbolo que está cercana al óptimo (a menos de un 6% del óptimo).

Ejemplo de Lloyd

Lloyd proporciona un ejemplo en el que tanto la aproximación variacional como la de punto fijo pueden conducir a cuantizadores localmente óptimos en lugar de a un cuantizador globalmente óptimo. Proponemos a continuación una ligera modificación al algoritmo de punto fijo que de hecho encuentra un cuantizador globalmente óptimo en el ejemplo de Lloyd. Conjeturizamos que esta técnica trabajará más generalmente, pero no hemos podido probar esto teóricamente. Una técnica similar se puede usar con el algoritmo de punto estacionario.

En lugar de usar muestras de la fuente que deseamos cuantizar, usamos muestras corrompidas por ruido aditivo independiente, donde la distribución marginal del ruido es tal que únicamente existe para ella un cuantizador localmente óptimo. Como ejemplo, para cuantización escalar con medida de distorsión la raíz del error, usamos ruido Gaussiano. En este caso, cualquier cuantizador localmente óptimo es también globalmente óptimo. Otras distribuciones, tales como la uniforme o un ruido de

amplitud discreta con un tamaño del alfabeto igual al número de niveles de salida del cuantizador, también se pueden usar.

Cuando la potencia del ruido es mucho mayor que la potencia de la fuente, la distribución de la suma es esencialmente la distribución del ruido. Asumimos que, inicialmente, la potencia del ruido es tan grande que solamente un cuantizador localmente óptimo existe para la suma; por tanto, sin reparar en la condición inicial, el algoritmo convergerá a este óptimo. En el paso siguiente, la potencia del ruido se reduce ligeramente y el cuantizador resultante de la ejecución previa se usa como la condición inicial. Intuitivamente, como el ruido se ha reducido en una pequeña cantidad, el óptimo global para la nueva suma debería estar próximo a aquél de la suma anterior (usamos las mismas muestras de fuente y de ruido con potencia de ruido reducida). Por tanto, esperamos que el algoritmo converja al óptimo global incluso aunque se deberían de haber introducido los nuevos puntos localmente óptimos. Continuamos del mismo modo, reduciendo el ruido gradualmente a cero.

Ejemplo de Voz

En el siguiente ejemplo, consideramos el caso de un sistema de compresión de voz que consiste en un análisis LPC de tramas de voz de 20 mseg de duración que produce decisión de voz/no voz y un pitch, una ganancia y un filtro inverso normalizado como se describió previamente, seguido de cuantización en la que el pitch y la ganancia se cuantizan separadamente como es habitual, pero los coeficientes del filtro normalizado $(a_1, \ldots, a_K) = (x_0, x_1, \ldots, x_{K-1})$ se cuantizan como un vector con $k = 10$, usando la medida de distorsión de (3.101)-(3.102). La secuencia de entrenamiento consistió en una secuencia de vectores parámetros del filtro inverso normalizado (*). El análisis LPC fue digital, y por tanto la secuencia de entrenamiento usada estaba ya "correctamente cuantizada" a 10 bits por muestra o 100 bits por cada vector. La ganancia original requería 12 bits por trama de voz y el pitch usaba 8 bits por trama de voz. La tasa total de la salida del LPC (que queremos comprimir más por cuantización de bloques) es de 6.000 bits/s. No se intentó una mayor compresión de ganancia o del pitch en estos experimentos porque nuestro objetivo era estudiar únicamente la mejora

potencial en la cuantización de bloques de los parámetros de filtros normalizados. El problema más completo incluyendo ganancia y pitch implica muchos otras cuestiones y es el tema de otro artículo aún en elaboración.

Para la medida de distorsión de (3.101)-(3.102), el centroide del subconjunto S de una secuencia de entrenamiento $\{x_j, j = 0,1,\ldots,n-1\}$ es el vector u minimizando

$$\sum_{j:x_j \in S} (x_j - u) R(x_j) (x_j - u)^t . \qquad (3.115)$$

Observamos que la matriz de autocorrelación $R(x)$ es un subproducto natural del análisis LPC y no necesita ser recalculado. Esta minimización, sin embargo, es un problema de minimización de energía residual mínima en el análisis LPC, y puede ser resuelto mediante algoritmos LPC estándar tales como el algoritmo de Levinson. Alternativamente, la teoría de la matriz Toeplitz es un problema de minimización mucho más estudiado, y el centroide se puede obtener mediante técnicas variacionales, para ser

$$\hat{x}(S) = \left\{ \sum_{j:x_j \in S} R(x_j) \right\}^{-1} \sum_{j:x_j \in S} R(x_j) x_j{}^t . \qquad (3.116)$$

Se usó la técnica de ‘splitting’ para la hipótesis de partida y un umbral de distorsión de 0.5%. El algoritmo completo, por ejemplo, puede ser descrito como se muestra a continuación:

(0) Inicio: Fijar $N = 2^R$ con R un entero, donde N es el número mayor deseado de niveles. Fijar $k = 10, n =$ longitud de la secuencia de entrenamiento, $e = .005$. Hacer $M = 1$.

Dada una secuencia de entrenamiento $\{x_j; j = 0, \ldots, n-1\}$, hacer $\{x_j; j = 0, \ldots, n-1\}$, el alfabeto de la secuencia de entrenamiento. Determinar $\hat{A}(1) = \hat{x}(A)$, el centroide de la secuencia de entrenamiento completa usando (3.116) o el algoritmo de Levinson.

(1) (Splitting): Dado $\hat{A}(M) = \{y_i, i = 1, \ldots, M\}$, dividir cada vector de reproducción y_i en $y_i + e$ y $y_i - e$, donde e es un vector de perturbación fijo. Hacer $\hat{A}_0(2M) = \{y_i + e, y_i - e, i = 1, \ldots, M\}$ y después reemplazar M por $2M$.

(2) Hacer $m = 0$ y $D_{-1} = \infty$.

(3) Dado $\hat{A}_m(M) = \{y_1, \ldots, y_M\}$, encontrar su partición óptima $P(\hat{A}_m(M)) = \{S_i; i = 1, \ldots, M\}$, es decir, $x_j \in S_i$ si $d(x_j, y_i) \le d(x_j, y_l)$ para todo l. Calcular la distorsión resultante

$$\begin{aligned} D_m &= D\left(\left\{\hat{A}_m(M), P\left(\hat{A}_m(M)\right)\right\}\right) \\ &= n^{-1} \sum_{j=0}^{n-1} \min_{y \in \hat{A}_m} d(x_j, y). \end{aligned} \tag{3.117}$$

(4) Si $(D_{m-1} - D_m)/D_m \le e = .005$, ir al paso (6). En otro caso, continuar

(5) Encontrar el alfabeto de reproducción óptimo $\hat{A}_{m+1}(M) = \hat{x}\left(P\left(\hat{A}(M)\right)\right) = \{\hat{x}(S_i); i = 1, \ldots, l\}$ para $P\left(\hat{A}_m(M)\right)$. Reemplazar m por $m+1$ y volver a (3).

(6) Hacer $\hat{A}(M) = \hat{A}_m(M)$. El cuantizador de nivel M final viene descrito por $\hat{A}(N)$. En otro caso, ir al paso (1).

Se trata, por tanto, de un método de cuantización vectorial subóptimo. Sin embargo, puede llegar a ser más rápido y proporcionar resultados satisfactorios en comparación con los algoritmos anteriores.

Capítulo 4: Descripción de Subsistemas

4.1: Obtención de Muestras: Creación de la Base de Datos

La base de datos usada para este experimento consistió en declaraciones de 15 locutores, 3 varones y 12 mujeres, y las frases en español con las que se ha experimentado “Uno, dos, tres, cuatro” y “Teoría de la Señal”, constituyendo un total de 90 muestras en cada caso. Estas declaraciones fueron recogidas en algunas ocasiones con el mismo micrófono y en otras con dos micrófonos diferentes, casos que serán objeto de estudio en el apartado de experimentos, junto al resto de la casuística planteada como posible fuente de distorsión en la tasa de reconocimiento. Todas las declaraciones fueron grabadas a una frecuencia de 11KHz con 16 bits/muestra (régimen binario 176 Kbps) en modo mono.

De entre las seis declaraciones por frase y locutor que se han adquirido, cinco se destinaron a la fase de entrenamiento y la última, a la fase de prueba (esto se eligió aleatoriamente) del sistema de reconocimiento.

Una vez recogidas las declaraciones, estas son almacenadas en una base de datos, pero no en el propio formato en que se recepcionan, sino tras un procesamiento que describiremos a continuación.

4.2: Cálculo de Coeficientes de Fourier-Bessel: Extracción de la Característica

Tras la grabación de las declaraciones de los diferentes locutores a través del micrófono y el almacenamiento de las mismas en ficheros de extensión .wav, el primer paso es importarlas al programa para convertirlas a un formato adecuado (secuencia de muestras) para trabajar con ellas desde MATLAB.

Para evitar una excesiva carga computacional y, por tanto, un menor rendimiento del sistema, los coeficientes que se calcularán en principio serán los de la fase inicial o de entrenamiento del identificador, cálculo que se tendrá que realizar una única vez para contar así con la base de datos preparada. Será en el momento en que un locutor se deba someter a una prueba de reconocimiento cuando los coeficientes característicos de su declaración se calculen, tal y como ocurriría en la realidad (ya que el reconocimiento se hace ‘a posteriori’).

Para proceder al cálculo de los coeficientes característicos, seleccionamos de cada declaración un total de 5.000 muestras, con el fin de que el número de tramas en las que se segmente la señal y, con ello, las dimensiones de los vectores característicos, sean similares para todas las declaraciones. Asimismo, se han efectuado pruebas de reconocimiento con un menor número de muestras de voz, en concreto, 4.000 muestras, con el objeto de que el sincronismo entre declaraciones se ajuste lo máximo posible.

Tal y como el método de estudio indica, se considera en el cálculo de los coeficientes de Fourier-Bessel que el número de tramas en que se segmenta la señal de voz debe ser tal que la longitud de cada trama permita considerar la voz estacionaria en dicha trama, y debe existir un solapamiento entre trama y trama segmentada suficiente para que el cálculo de la característica sea contínuo y congruente con las tramas adyacentes. Con estas hipótesis de partida, se opta por tramas de segmentación de 200 muestras solapadas 100 muestras con las tramas adyacentes.

4.2.1: Selección de Coeficientes

En referencia a la selección de coeficientes de Fourier-Bessel a calcular, en las pruebas inciales se optó por calcular 200 coeficientes por trama, solución que no comprime en absoluto la información de partida pero permite comprobar la efectividad del método mediante la reconstrucción de la voz a partir de sus coeficientes característicos. En posteriores pruebas, se han realizado selecciones diversas de estos coeficientes (de forma secuencial, en función de la magnitud relativa, etc.) y, en el grueso del proyecto, se han elegido los coeficientes tomando una solución de compromiso que aporte la ventaja de sintetizar la voz en una cantidad de datos reducida pero que, a la vez, garantice que la pérdida de información a determinadas frecuencias críticas no va a ser significativa. Esta solución resulta de elegir los coeficientes que cubren aproximadamente toda la banda de frecuencia de la señal de voz. Son los dados por el vector de índices siguiente

$$k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190] \quad (4.1)$$

que se compone de 15 elementos, cada uno de ellos correspondiente a un rango de frecuencias significativo dentro de la banda de la señal de voz y siendo la frecuencia inicial que se cubre, con el procesado de 5 coeficientes consecutivos tras el índice marcado, de 205 Hz, y la final, de 3905 Hz, tal y como se describe en el apartado 3.6.1,

lo que garantiza que, con un total de 195 coeficientes de los que se evalúa la energía, tendremos en cuenta todas las frecuencias en las que se transmite la voz humana.

No obstante, se han realizado pruebas en el proyecto con un menor número de coeficientes, con el fin de determinar cuál es el rango de frecuencias en el que reside la información específica del tracto vocal - y, por tanto, del locutor concreto- y de evaluar la variación en la tasa de reconocimiento del sistema al reducir en gran medida la carga computacional. En concreto, los coeficientes tomados en el caso de no abarcar en las pruebas el ancho de banda total de la voz, se reducen a los señalados por el vector de índices siguiente:

K2=[10,20,35,50,60] *(4.2)*

En este caso, se realiza la extracción de la característica de la misma forma, evaluando un total de 65 coeficientes de Fourier-Bessel para determinar la energía en los rangos de frecuencia centrados en las correspondientes a los índices del vector anterior.

Todos estas variantes se detallan en el apartado de Experimentos y Resultados.

4.2.2: Ortogonalidad de las Funciones de Bessel

Para llevar a cabo la obtención de los coeficientes mediante una fórmula compacta, se hace uso de la propiedad de ortogonalidad de las funciones de Bessel, propiedad que ha sido demostrada en el capítulo 3.

Si consideramos $f(x)$ la señal de voz, ésta se puede expresar como combinación lineal de un conjunto ortogonal de la forma

$$f(x) = c_1 J_n(l_{1n} x) + c_2 J_n(l_{2n} x) + \ldots \qquad (4.3)$$

siendo

$$l_{mn} = \frac{a_{mn}}{R} \qquad (m = 1,2,\ldots) \qquad (4.4)$$

y a_{mn} los ceros positivos de la función de Bessel J_n.

La expresión dada por (4.3) se conoce como serie de Fourier-Bessel, y aplicando la propiedad de ortogonalidad de las funciones de Bessel, obtenemos la expresión que determina con exactitud los coeficientes característicos de Fourier-Bessel de una trama de voz,

$$C_m = \frac{2}{a^2 J_0(x_m)^2} \int_0^a t x(t) J_1\left(\frac{x_m}{a} t\right) dt \qquad (4.5)$$

No obstante, realizamos una aproximación para implementar la transformada de Fourier-Bessel en tiempo discreto mediante MATLAB.

4.2.3: Regla de Simpson

Para el cómputo de los coeficientes de Fourier-Bessel, nos encontramos con una integral, para cuya implementación acudimos inicialmente a un método numérico: la Regla de Simpson.

Recordemos que, según la Regla de Simpson, una integral definida I se resolvía numéricamente según la fórmula siguiente:

$$I = \int_a^b f(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] \qquad (4.6)$$

siendo

$x_0 = a$

$x_2 = b$

$x_1 = a + h$

$h = (b - a)/2$

Esta regla se aplicó de la misma forma al cálculo de los coeficientes C_m.

No obstante, la enorme ralentización que este método producía en la ejecución del programa, nos llevó a asumir una aproximación a esta integral mediante una suma discreta de los términos que intervienen en la misma, quedando la expresión que determina los coeficientes de la forma siguiente

$$C_m = \frac{1}{\left\| \sum_{n=1}^{nmuest} J_1\left(n \frac{1}{nmuest} x_m \right) \right\|^2} \hat{x} \sum_{n=1}^{nmuest} J_1\left(n \frac{1}{nmuest} x_m \right) \qquad (4.7)$$

siendo $\hat{x}$, la trama de voz enventanada; *nmuest* la longitud de una trama y x_m el m-ésimo cero de la función de Bessel de primera especie.

Se puede observar la implementación de este método en la función de MATLAB *fb.m*, encargada de calcular los coeficientes de Fourier-Bessel, en el capítulo 7.

4.2.4: Recontrucción de la Señal de Voz

Aunque no fuera el objeto de este estudio, nuestra meta inicial tras el cálculo de los coeficientes de Fourier-Bessel fue realizar la comprobación de que, efectivamente, los coeficientes extraidos de la señal de voz constituían una caracterización de la misma de forma compacta. Por esta razón, la verificación más inmediata de la bondad de estos coeficientes era la reconstrucción de las tramas de voz de las que se extrajeron a partir de los mismos.

La reconstrucción exacta de la señal de voz se obtiene de la siguiente expresión

$$x(t) = \sum_{m=1}^{\infty} C_m J_1\left(\frac{x_m}{a} t\right) \qquad (4.8)$$

Dado que pretendemos reconstruir la señal a partir de un número finito de coeficientes, consideramos la aproximación siguiente

$$x(t) \approx \sum_{m=1}^{M} C_m J_1\left(\frac{x_m}{a} t\right) \qquad (4.9)$$

siendo M el número de coeficientes a partir de los cuales realizamos la reconstrucción.

El resultado de esta prueba, en función del número de coeficientes FB empleados para la misma, se detalla gráficamente en el capítulo de Experimentos y Resultados.

4.2.5: Extracción de la Característica

Se puede comprobar que, una vez computados los coeficientes de Fourier-Bessel de una trama de voz, si seleccionamos coeficientes específicos que consideramos corresponden a frecuencias críticas que caracterizan al locutor, los coeficientes adyacentes a estos no siguen una variación abrupta respecto a los de partida, sino que, en su lugar, la magnitud de los mismos se mantiene a lo largo de un determinado rango de frecuencia. Por esta razón, se considera en el estudio que se aporta una mayor cantidad de información del locutor si se evalúan, con cada coeficiente 'crítico', los cinco coeficientes consecutivos. Puesto que esta información implica la necesidad de una mayor capacidad de almacenamiento de datos y una mayor tasa de transmisión de datos, en el caso de que estos formen parte de un sistema, se opta por evaluar la energía de los coeficientes característicos en un determinado rango de frecuencias, de la forma

$$e(i) = \sum_{m=k}^{k+5} |C_m| \qquad (4.10)$$

para $i = 1, 2, \ldots, 15$ y $k = 10, 20, 35, 50, 60, 70, 80, 90, 100, 115, 130, 145, 160, 175, y\ 190$, respectivamente, tal y como se describe en el capítulo 3. De este modo, conseguimos mantener información de todo el rango de frecuencias que abarca la voz humana en una cantidad reducida de datos.

Esto supone, por tanto, que los vectores de energía de los coeficientes característicos de Fourier-Bessel pasan a ser las características de cada locutor en el sistema de reconocimiento.

4.2.6: Cálculo de Centroides: Cuantizador Vectorial

En el apartado anterior hemos comentado cuáles van a ser los vectores característicos para cada locutor en el proceso de reconocimiento. Sin embargo, se puede observar que, a pesar de haber reducido en gran medida el volumen de información, y según la ecuación x, cada trama de voz de 200 muestras de una declaración se caracterizará por un vector de 15 elementos. Si consideramos que por cada locutor se evalúa una declaración de 5.000 muestras, segmentada en tramas de 200 con 100 muestras de solapamiento entre tramas adyacentes, tenemos un total de 50 vectores aproximadamente de 15 elementos, siendo todos los vectores diferentes entre sí. Se deduce entonces que el procesado de la señal supone una reducción del 85 % sobre el total de la información inicial. A pesar de este alto porcentaje, la cantidad absoluta de muestras a tratar sigue siendo importante, al disponer de un alto número de locutores y de varias declaraciones por locutor, dada la carga computacional que implica la obtención de estas características.

Por esta razón, y para reducir asimismo el procesado en la fase de identificación del locutor de prueba, es conveniente acudir a la cuantización vectorial de dichas características.

El algoritmo de cuantización vectorial empleado, LBG (Linde-Buzo-Gray), descrito en el capítulo 3, permite introducir como parámetro de entrada el número de vectores centroide o palabras de código –y, con ello, el número de regiones Voronoi- en las que queremos que se resuma el alfabeto de entrada. Durante los experimentos, se han obtenido resultados para diferentes cantidades de centroides (en concreto, $N = 10$, 15 ó 20, siendo N el número de vectores centroide). Como se describirá en el siguiente capítulo, este parámetro no ha sido un factor determinante en la tasa de éxito de reconocimiento, por lo que se ha tomado el valor mínimo de entre los empleados en las

pruebas, $N = 10$, para evaluar el efecto de los restantes parámetros sobre dicha tasa en los experimentos sucesivos. Esta dimensión del libro de código de cada uno de los locutores supone ya una reducción en la información almacenada del orden del 96%, ya que, en este caso, se debe contar no solamente con las palabras de código que componen el libro de cada declaración –vectores centroide -, sino también con el vector de índices que señalan a cada vector característico inicial como perteneciente a la región concreta caracterizada por cada palabra de código o centroide.

4.3: Fases del Sistema de Reconocimiento

El sistema de reconocimiento se ha constituido de dos fases principales: la fase de entrenamiento del sistema –o de almacenamiento de los patrones característicos de cada locutor de referencia en la base de datos -, y la fase de prueba de un locutor concreto de identidad desconocida.

Estas dos fases se han diferenciado para las dos modos de reconocimiento planteados en el proyecto:

- Reconocimiento de conjunto cerrado.
- Reconocimiento de conjunto abierto.

Veremos la estructura de las dos fases señaladas para cada uno de los modos implementados, así como los factores fundamentales que marcan la diferencia entre una y otra modalidad en el sistema de reconocimiento.

4.3.1: Modo de Conjunto Cerrado

4.3.1.1: Fase de Entrenamiento

Durante la fase de entrenamiento en el reconocimiento en modo de conjunto cerrado, se realiza la carga de las cinco primeras declaraciones de todos los locutores grabados, y se procede al cálculo de sus coeficientes característicos.

Para realizar el almacenamiento de los vectores característicos de 5 declaraciones diferentes de un mismo locutor, consideramos cada una de estas declaraciones totalmente independientes, aunque todas llevan en última instancia asociado el mismo locutor. Sin embargo, no tratamos de resumir todas las declaraciones en una porque un promedio de los coeficientes perdería toda la información característica almacenada en cada uno de ellos. De ahí que obtengamos cinco patrones, que también nos proporcionan mayor información del locutor.

Los parámetros de libertad de los que disponemos en este análisis son:

- Número de muestras de las declaraciones que van a ser analizadas en el sistema
- Número de muestras por trama en las que va a ser segmentada la señal de voz.
- Número de muestras de solapamiento entre tramas segmentadas adyacentes.
- Número de coeficientes de Fourier-Bessel a calcular.
- Frecuencias críticas o, con más exactitud, rango de frecuencias en los que calcular los coeficientes de Fourier-Bessel.
- Número de centroides en los que se resumirán los vectores característicos de energía para la fase de reconocimiento.
- Método de elección de la condición inicial para el cálculo de estos centroides.

En nuestro estudio, hemos observado el efecto que sobre la tasa de éxito en el reconocimiento tienen las variaciones de los cuatro últimos parámetros, manteniendo el resto de parámetros fijos y actuando sobre uno solo de ellos en cada prueba. De la misma manera, se han realizado pruebas variando el primero de los parámetros, como se ha comentado en anteriores apartados. Los resultados de estas pruebas se describirán detalladamente en el siguiente capítulo.

Una vez obtenidos los vectores de energía de los coeficientes FB, posteriormente cuantizados, tal y como se ha descrito anteriormente, almacenamos la información en tres tablas:

- Centr: Tabla que almacena los vectores centroide de cada locutor por filas.
- Bdatos: Tabla que almacena en cada fila las referencias, índices o punteros de cada vector característico de energía a los centroides de la tabla anterior.
- Bdloc: Tabla que almacena por filas el nombre de cada locutor que corresponda a los datos almacenados en las dos bases de datos anteriores. Asimismo, asocia dicho nombre a una posición en la base de datos del sistema.

Salvando estas tres tablas en memoria, queda finalizada la fase de entrenamiento para el sistema de reconocimiento de conjunto abierto.

4.3.1.2: Fase de Prueba

Durante esta fase, se recoge la grabación de un locutor de identidad desconocida, y se procede a identificarlo como uno de los pertenecientes a la base de datos inicial. Para ello, el proceso de prueba pasa como parámetros a las funciones llamadas las bases de datos creadas durante la fase de entrenamiento. Los valores de los parámetros que se adoptan aquí deben coincidir con los que se tomaron en la fase de entrenamiento, para ser coherentes en la ejecución del programa. No obstante, tras el cálculo de los coeficientes de Fourier-Bessel de la señal de voz de prueba, y de la obtención de los

vectores de energía correspondientes, hemos planteado tres posibles métodos de obtención de la característica y de posterior comparación con los patrones de los locutores de entrenamiento:

1) Calcular los centroides de los vectores característicos de prueba, y comparar el alfabeto de prueba con cada uno de los de entrenamiento a través de la referencia de cada vector característico a su palabra de código o centroide correspondiente. En este método comparamos en resumen centroides, pero de forma ordenada en el tiempo, trama a trama. El inconveniente de esta opción es que se requiere un gran sincronismo entre las declaraciones de entrenamiento y prueba, para que las distorsiones calculadas sean fiables y dependan del locutor, no del texto.

2) No calcular los centroides de los vectores característicos de prueba, y compararlos directamente con el alfabeto que hace referencia a los centroides de los locutores de entrenamiento. En esta opción se realizan comparaciones entre vectores característicos (diferentes entre sí, por tanto, no cuantizados) y centroides, pero también de forma ordenada por tramas. De la misma forma que en el caso anterior, aquí también se requiere un alto sincronismo entre declaraciones para que el cálculo de las distorsiones refleje las diferencias entre los locutores.

3) Calcular los centroides de los vectores característicos y compararlos con los de entrenamiento directamente, no de forma ordenada, con el objeto de asociar los centroides cuya diferencia es mínima dentro del libro de código. Esta opción busca obtener la distorsión mínima en cada comparación, y no requiere sincronismo entre las declaraciones, aunque resulta bastante más costosa computacionalmente por el número de comparaciones que requiere realizar hasta encontrar el óptimo.

La medida de distorsión que se ha tomado como parámetro de decisión a lo largo de este estudio ha sido la Distancia Euclídea.

Obtenida la referencia al locutor de entrenamiento que proporciona la menor distorsión respecto al locutor de prueba, la asociamos, mediante la base de datos bdlocut

pasada como parámetro, al nombre del locutor, y lanzamos un mensaje que indica quién es el locutor.

Con este mensaje finaliza la fase de prueba y, con ello, el proceso de reconocimiento automático de locutores en modo de conjunto cerrado.

4.3.2: Modo de Conjunto Abierto

Conocemos en qué se basa este modo de reconocimiento, ya que ha sido descrito al introducir este estudio. Se pretende en este punto identificar que el locutor de prueba corresponde al locutor de entrenamiento, o bien que no pertenece a la base de datos.

Puesto que la mayor parte de los procesos llevados a cabo en las fases de entrenamiento y prueba para esta modalidad son similares a las de conjunto cerrado, haremos hincapié únicamente en las diferencias respecto a los métodos descritos en el apartado anterior.

4.3.2.1: Fase de Entrenamiento

En esta fase, todo el procesado que sufre la señal de voz tras efectuarse la grabación, es exactamente igual al realizado con el reconocimiento de conjunto cerrado, salvo con la excepción de que el procesado en este caso se realiza para patrones que corresponden al mismo locutor.

Un punto que se añade a este procesado es el cálculo del umbral. Aquí se plantea necesario marcar un umbral de distorsión a partir del cual el locutor se rechaza por considerarlo no perteneciente a la base de datos de este estudio. Por ello, es conocido este parámetro con el nombre de umbral de rechazo.

El umbral se calcula ‘a priori’ conociendo que se establece entre declaraciones del mismo locutor. Entra en juego, por tanto, en esta fase de entrenamiento, el cálculo de distorsión entre patrones de locutores que antes se realizaba exclusivamente en la fase de identificación. Se pueden considerar, de la misma forma, dos opciones para

determinar esta distorsión (las enumeradas en la fase de prueba anterior, a excepción de la segunda, por carecer de sentido realizar comparaciones entre los mismos patrones obtenidos por métodos diferentes), y se tomará el umbral como una función de estas distorsiones mínimas. Hemos analizado diferentes fórmulas para definir el umbral óptimo a partir de las distorsiones obtenidas, con el ánimo de llegar a una solución de compromiso que evite identificar de forma errónea al locutor desconocido como el locutor de prueba por un umbral excesivamente alto, y que, a la vez, evite identificar al locutor de referencia como no perteneciente a la base de datos por ajustar el umbral al mínimo.

Sobre este punto se han realizado numerosas pruebas con el fin de determinar, empíricamente, la expresión óptima del umbral de reconocimiento, pruebas que también serán descritas más adelante.

4.3.2.2: Fase de Prueba

Esta fase también resulta similar a la de conjunto abierto, con la salvedad de que admite como parámetro de entrada el umbral calculado en la fase de entrenamiento, para posteriormente contrastar la distorsión obtenida con el valor de este umbral y, en función de la comparativa realizada con este umbral, emitirá dos tipos de mensajes de texto:

- ‘El locutor es <nombre_del_locutor_de_referencia>’, si umbral es mayor o igual que la distorsión mínima obtenida
- ‘El locutor no pertenece a la base de datos’, si umbral es menor que la distorsión mínima obtenida.

Con este mensaje finaliza la fase de prueba en el reconocimiento de conjunto abierto.

4.4: Descripción Funcional

A continuación describiremos cada una de las funciones que intervienen en el método implementado objeto de estudio en este proyecto.

Train1.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto cerrado (diferentes locutores, un solo micrófono, mic1, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m.

Train2.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto abierto (un único locutor, dos micrófonos, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m y a cumbral.m.

Train3.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto abierto (un único locutor, un solo micrófono, mic2, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m y a cumbral.m.

Train4.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto abierto (un único locutor, un solo micrófono, mic2, 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m y a cumbral.m.

Train5.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto cerrado (diferentes locutores, un solo micrófono, mic1, 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m.

Train6.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto cerrado (diferentes locutores, un solo micrófono, mic2 , 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m.

Train7.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto abierto (un único locutor, un solo micrófono, mic2 , 4.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m y cumbral.m.

Train8.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto cerrado (todos los locutores de la base de datos, dos micrófonos, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m.

Train9.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto cerrado (locutores grabados con micrófono 2, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m.

Traina.m: Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema de reconocimiento para conjunto cerrado (todos los locutores de la base de datos, dos micrófonos, 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los locutores llamando a vtrain.m.

Vtrain.m: Función principal de la fase de entrenamiento en el reconocimiento de conjunto cerrado que realiza llamadas a leedatos.m, para convertir las señales de voz a secuencias de muestras, fb.m, para el cálculo de los coeficientes FB, energia.m, para el cálculo de los vectores característicos de energía y cvect.m, para la cuantización vectorial de estos vectores característicos. Devuelve los vectores característicos y sus centroides para cada declaración para su posterior almacenamiento en la base de datos del sistema.

Leedatos.m: Función que traduce el formato de las ondas de voz grabadas .wav a un formato de secuencia de muestras que permita su procesamiento en MATLAB.

Fb.m: Función que calcula los coeficientes de Fourier-Bessel de un segmento de voz, dado el propio segmento, el número de coeficientes a calcular, la longitud de las tramas de segmentación y el número de muestras de solapamiento entre tramas adyacentes. Para ello realiza una llamada a cerosb.m.

Cerosb.m: Función que calcula los ceros de la función de Bessel de primera clase para una longitud de muestras dada.

Energia.m: Función que determina el vector característico de energía para unos coeficientes de Fourier-Bessel asociados a unas determinadas frecuencias críticas para la identificación del locutor, que se pasan como parámetro por el vector de sus índices.

Cvect.m: Función principal del cuantizador vectorial. Recibe las muestras de los vectores característicos de energía de los coeficientes de Fourier-Bessel y el número de centroides deseado y devuelve los centroides calculados y un vector de índices que marca la correspondencia entre el alfabeto inicial de entrada y dichos centroides.

F1.m: Función que da condiciones iniciales al algoritmo de cuantización vectorial para la primera iteración.

F2.m: Función que define las regiones del cuantizador conocidos sus centroides.

F3.m: Función que asigna un centroide a cada región y evalúa el error total cometido.

F4.m: Función llamada por f3.m para calcular los centroides.

Cumbral.m: Función que determina, dado un conjunto de vectores característicos y centroides de estos correspondientes a las declaraciones de entrenamiento de un solo locutor conocido almacenadas en la base de datos, el nivel de umbral máximo entre declaraciones del mismo locutor, de forma que, si se supera este umbral en la fase de prueba se pueda desechar el locutor desconocido como no perteneciente a la base de

datos. Esta función se aplicará únicamente al reconocimiento de locutores de conjunto abierto. Para determinar esto, realiza llamadas a las funciones disteu1.m, disteu2.m y disteu3.m.

Disteu1.m: Función que determina la distancia euclídea entre cada centroide que corresponde al vector característico de entrenamiento y el correspondiente al vector característico de prueba para cada trama de voz. Para ello se le pasan como parámetros las bases de datos de vectores característicos y de sus centroides correspondientes. Este método de cálculo requiere de la existencia de sincronismo entre las declaraciones a nivel de tramas de voz.

Disteu2.m: Función que determina la distancia euclídea entre cada centroide que corresponde al vector característico de entrenamiento y el vector característico de prueba para cada trama de voz. Para ello se le pasan como parámetros las bases de datos de entrenamiento (vectores característicos y sus centroides correspondientes) y los vectores característicos de la declaración de prueba. Este método de cálculo requiere de la existencia de sincronismo entre las declaraciones a nivel de tramas de voz.

Disteu3.m: Función que determina la mínima distancia euclídea entre centroides correspondientes a los vectores característicos de entrenamiento prueba sin considerar la correspondencia entre tramas. Para ello se le pasan como parámetros los centroides de las declaraciones de entrenamiento y prueba. Este método de cálculo no requiere de la existencia de sincronismo entre las declaraciones a nivel de tramas de voz, pero resulta más costoso a nivel computacional. Se evalúa aquí el grado de similitud global entre declaraciones, seleccionando el mínimo de cada comparación.

Test1.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto cerrado (diferentes locutores, un solo micrófono, mic1, 5.000 muestras de voz por declaración, 195 coeficientes FB) llamando a vtest.m y a identcc.m.

Test2.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto abierto (un único locutor, dos micrófonos, 5.000 muestras de voz por declaración, 195 coeficientes FB) llamando a vtest.m y a identca.m.

Test3.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto abierto (un único locutor, un solo micrófono, mic2, 5.000 muestras de voz por declaración, 195 coeficientes FB) llamando a vtest.m y a identca.m.

Test4.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto abierto (un único locutor, dos micrófonos, 5.000 muestras de voz por declaración, 65 coeficientes FB) llamando a vtest.m y a identca.m.

Test5.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto cerrado (diferentes locutores, un solo micrófono, mic1, 5.000 muestras de voz por declaración, 65 coeficientes FB) llamando a vtest.m y a identcc.m.

Test6.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto cerrado (diferentes locutores, un solo micrófono, mic2, 5.000 muestras de voz por declaración, 65 coeficientes FB) llamando a vtest.m y a identcc.m.

Test7.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto abierto (un único locutor, dos micrófonos, 4.000 muestras de voz por declaración, 195 coeficientes FB) llamando a vtest.m y a identca.m.

Test8.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto cerrado (todos los locutores de la base de datos, dos micrófonos, 5.000 muestras de voz por declaración, 195 coeficientes FB) llamando a vtest.m y a identcc.m.

Test9.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto cerrado (locutores grabados con micrófono 2, 5.000 muestras de voz por declaración, 195 coeficientes FB) llamando a vtest.m y a identcc.m.

Testa.m: Programa 'script' de MATLAB que genera el proceso de prueba del sistema de reconocimiento del locutor desconocido para conjunto cerrado (todos los locutores de la base de datos, dos micrófonos, 5.000 muestras de voz por declaración, 65 coeficientes FB) llamando a vtest.m y a identcc.m.

Vtest.m: Función principal de la fase de prueba en el reconocimiento, tanto de conjunto cerrado como de conjunto abierto, que realiza llamadas a leedatos.m, para convertir las señales de voz a secuencias de muestras, fb.m, para el cálculo de los coeficientes FB, energia.m, para el cálculo de los vectores característicos de energía y cvect.m, en caso de que se realice la cuantización vectorial de estos vectores característicos. Devuelve los vectores característicos y/o sus centroides para cada declaración para su posterior comparación con la base de datos almacenada en el sistema.

Identcc.m: Función que determina, dado un conjunto de vectores característicos correspondientes a una declaración conocida y a un locutor desconocido, la identidad de dicho locutor de entre un conjunto cerrado de locutores cuyos vectores característicos se encuentran almacenados en una base de datos que se pasa como parámetro. Esta identificación la realiza llamando a las funciones disteu1.m, disteu2.m o disteu3.m, dependiendo del tipo de comparación que se haga para calcular la distorsión entre declaraciones, y finalmente, a bdlocut, para identificar la posición del locutor en la base de datos con un nombre concreto.

Identca.m: Función que determina, dado un conjunto de vectores característicos correspondientes a una declaración conocida y a un locutor desconocido, si la identidad de dicho locutor corresponde a alguno de los patrones del locutor de referencia para reconocimiento de conjunto abierto a partir de los vectores característicos que se encuentran almacenados en una base de datos que se pasa como parámetro. Esta identificación la realiza llamando a las funciones disteu1.m, disteu2.m o disteu3.m, dependiendo del tipo de comparación que se haga para calcular la distorsión entre declaraciones. A continuación comparará la distorsión obtenida con la variable global umbral (calculada en la fase de entrenamiento), determinando, de este modo, si el locutor corresponde al de referencia de la base de datos o bien si es un impostor. Para

cada una de las opciones enviará un mensaje de texto que refleje el resultado de la decisión.

Bdlocut.m: Función que recibe como parámetro de entrada el número del locutor identificado en la base de datos, así como la base de datos que asocia un número a cada locutor. Devuelve el nombre del locutor registrado en la base de datos que se identifica con el número dado.

Capítulo 5: Experimentos y Resultados

5.1: Introducción

Los experimentos realizados han consistido en grabar diferentes declaraciones, protagonizadas por diferentes locutores, a través de dos micrófonos.

Como se describió en el capítulo anterior se contemplaron dos frases para realizar las declaraciones. Para el estudio que se expone, nos hemos centrado sobre la primera de ellas: 'Uno, dos, tres, cuatro'. A pesar del hecho de que la misma declaración hace suponer un estudio de reconocimiento con dependencia del texto, veremos que, a partir de estas mismas declaraciones, podemos también estudiar la capacidad de reconocimiento del sistema con independencia del texto, basándonos, para ello, en descartar las frecuencias que albergan principalmente el contenido del mensaje y centrar el análisis sobre aquéllas que caracterizan unívocamente el tracto vocal de cada individuo.

En el apartado de Descripción de Subsistemas, comentábamos los grados de libertad de los que disponemos en el sistema de reconocimiento, con el objetivo de definir el conjunto de valores de los parámetros que nos proporcionan la mayor tasa de éxito en el reconocimiento. Describiremos a continuación, los parámetros sobre los que hemos aplicado variaciones para observar su efecto en la tasa de reconocimiento, así como el significado de cada uno de los valores que toman dichos parámetros.

- En primer lugar, hemos diferenciado aquellos casos en los que las declaraciones tratadas se recogieron con un micrófono (micrófono 1), de aquéllos en los que fue empleado el segundo micrófono para la grabación, ya que se advierte una notable diferencia en el porcentaje de éxito en el reconocimiento obtenido con uno y otro, lo cual nos lleva a observar la importancia del hardware en los resultados, a pesar de ser ésta una mínima parte del sistema global. Asimismo, se mostrarán pruebas en las que intervienen grabaciones procedentes de ambos micrófonos con el fin de comprobar el efecto combinado de los mismos, y la diferencia que se aprecia en los locutores identificados con éxito mediante un receptor y otro. Del mismo modo, se observará que el empleo de ambos afectará, de forma negativa, a los patrones de voz más limpios, en la fase de identificación.

- En cuanto al número de coeficientes de Fourier-Bessel adquiridos para almacenar las características de cada locutor en la base de datos, hemos considerado dos opciones: la primera de ellas, con 195 coeficientes de cálculo centrados en el vector de índices k mostrado en el capítulo anterior, refleja el resultado de pruebas, de conjunto cerrado o abierto, con dependencia del texto, ya que cubre aproximadamente el total del ancho de banda de la voz humana. La segunda opción, que comprende 65 coeficientes centrados en el vector de índices k2, es una muestra del reconocimiento de locutores con independencia del texto, al omitir en el procesado las frecuencias altas del espectro de la voz que son las responsables del contenido del mensaje, y trabajar únicamente con los coeficientes correspondientes a las bajas frecuencias, características del tracto vocal del locutor concreto o, más concretamente, características de la inteligibilidad de la declaración. La opción de un menor número de coeficientes supone una necesidad

menor de capacidad de almacenamiento del sistema, y se debe al hecho de que se ve reducida la dimensión de los vectores característicos de energía de 15 elementos (en el caso de ncoef=195) a 5 en la base de datos (en el caso de ncoef2=65).

- Otro parámetro sobre el que hemos actuado durante las diferentes pruebas ha sido el número de vectores centroide o tamaño N del libro de código en el que resumir las características tras hacerlas pasar por el Cuantizador Vectorial. Dado que el número de vectores característicos de energía calculados para cada declaración de 5.000 muestras se resume en 50, tiene sentido reducir este número en N=10, 15 ó 20 centroides, y han sido estos los valores tomados para este parámetro durante cada una de las pruebas realizadas. Se hará notar el efecto del valor de N sobre el tiempo de obtención de la base de datos de locutores de referencia del sistema, así como el valor añadido que supone un valor de N mayor frente a la carga computacional que representa, dado que la pérdida de información característica del locutor será inferior en estos casos pero, las fases de entrenamiento y prueba se verán igualmente ralentizadas por aumentar el número de vectores a procesar y comparar.

- La dimensión de las declaraciones ha sido un parámetro prácticamente fijo a lo largo de todo el proceso, con un valor de 5.000 muestras por declaración. No obstante, se ha realizado una prueba con 4.000 muestras por declaración con el objeto de observar una posible mejora en el sincronismo entre tramas de los diferentes locutores – y, consecuentemente, en la tasa de reconocimiento -, y la reducción en la carga computacional ya que, este decremento en el número de muestras por declaración supone una disminución en la cantidad de vectores característicos de energía a procesar en el Cuantizador Vectorial, concretamente de 50 a 40 en el paso de 5.000 a 4.000 muestras.

- En referencia al método de cálculo de la distorsión mínima entre declaraciones de entrenamiento y prueba, en el caso de reconocimiento de conjunto cerrado, y entre las propias declaraciones de entrenamiento de un mismo locutor para la

determinación del umbral, en el caso de reconocimiento de conjunto abierto, hemos considerado tres métodos posibles:

1) Calculo=1: Este es el caso convencional, en el que se comparan los vectores centroide calculados a los que hace referencia cada uno de los vectores característicos para cada trama de la declaración. Esto supone una comparación entre características que coinciden en el tiempo, por lo que se asume un total sincronismo entre las declaraciones a nivel de tramas de voz, caso ideal que no se da en la mayoría de las ocasiones, ya que entra en juego en este punto la cadencia del habla de cada locutor particular.

2) Calculo=2: En este caso, se comparan los vectores centroide, a los que hace referencia cada vector característico de la declaración de entrenamiento, directamente con cada uno de los vectores característicos de la declaración de prueba (Esta configuración concreta no se emplea en el modo de reconocimiento de conjunto abierto, en lo referente a la fase de entrenamiento). Evitamos, de este modo, el proceso de Cuantización Vectorial en la fase de identificación, pero almacenamos para ello un mayor número de vectores en memoria. Puesto que la comparación se realiza trama a trama, un funcionamiento correcto exige un nivel de sincronismo entre declaraciones similar al requerido en el método anterior.

3) Calculo=3: Este método requiere de la Cuantización Vectorial en ambas fases del sistema, pero compara directamente los vectores centroide entre sí, sin referencia alguna, de forma que se asociarán aquellos dos vectores que proporcionen una distorsión mínima. Se busca, por tanto, analizar la similitud global entre declaraciones de forma atemporal, lo que nos lleva a pensar que ésta sea la opción ideal para analizar el funcionamiento del sistema con independencia del texto, extrayendo únicamente los coeficientes característicos de las bajas frecuencias. No obstante, se presentarán los resultados para todos los casos planteados.

- En cuanto al funcionamiento interno del Cuantizador Vectorial, hemos distinguido dos opciones que definen la elección del libro de código inicial para ejecutar el algoritmo de Linde-Buzo-Gray. Las dos posibilidades que se plantean para esta condición inicial son:

  1) Metodo=1: El libro de código inicial del algoritmo consistirá en los N primeros vectores del alfabeto de entrada, es decir, los N primeros vectores de energía característicos.

  2) Metodo=2: En este caso, los N componentes del libro de código inicial se elegirán de entre los vectores característicos de energía de forma aleatoria.

- Observar que, este último caso (Metodo=2), combinado con la comparación directa entre centroides para determinar la distorsión mínima entre declaraciones (Calculo=3), es el único cuyo resultado en la identificación puede variar notablemente de una ejecución a otra. Esto se debe a la elección aleatoria de la condición inicial y, fundamentalmente, a que el Cuantizador Vectorial emplea un algoritmo subóptimo, lo que implica que se puede llegar a soluciones de centroides diferentes. Estas diferencias no suelen ser significativas en los casos en que se Calculo=1 ó 2, ya que en ellos se realiza secuencialmente la comparación entre 50 vectores característicos, y el resultado de las mismas salvaría cualquier desviación producida en la obtención de los centroides.

- Concretando en las pruebas de reconocimiento de conjunto abierto, el único parámetro que se nos presenta como grado de libertad, además de los enumerados anteriormente, es la expresión para el cálculo del umbral. Éste resulta ser el parámetro crítico en la tasa de éxito para este modo de reconocimiento, y se han realizado pruebas con diferentes expresiones que son función de la distorsión obtenida entre los patrones de referencia. Se mostrarán los resultados obtenidos para cada una de las expresiones en el apartado de pruebas para esta modalidad en el reconocimiento de locutores.

De forma previa a la presentación de los resultados de las numerosas pruebas realizadas, mostraremos la reconstrucción de la señal de voz a partir de los coeficientes característicos de Fourier-Bessel.

5.2: Reconstrucción de la Señal de Voz

Tal y como hemos comentado anteriormente, la reconstrucción de la señal de voz tiene por objeto únicamente verificar la efectividad de los coeficientes de Fourier-Bessel para caracterizarla. Por ello, únicamente mostraremos un ejemplo gráfico resultado de ejecutar el siguiente código fuente de MATLAB. Dada la señal de voz, x, convertida previamente a una secuencia de muestras:

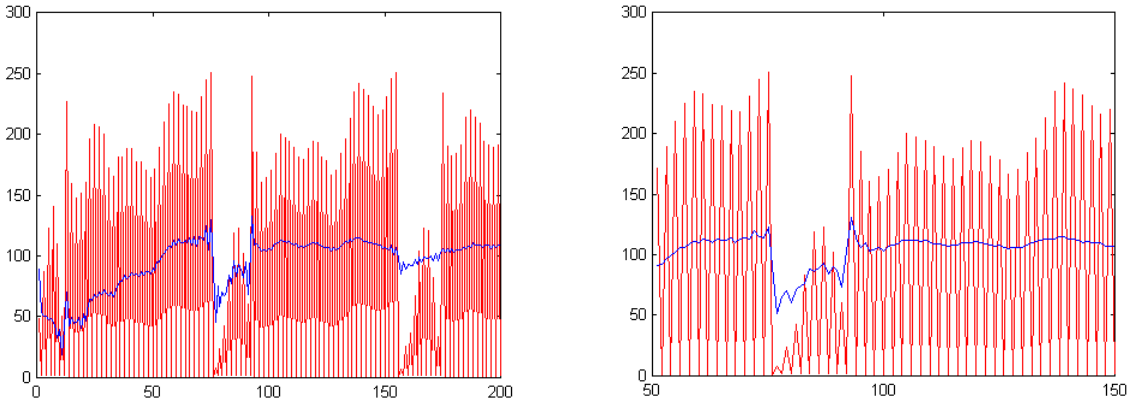
```
function [cm,vozrec]=reconst(x,nmuest,nsolap,ncoef,nrec)
%       cm es la matriz de coeficientes de Fourier-Bessel. Su longitud
%       vendrá  determinada por el número de coeficientes que queramos
%       calcular, ncoef.
%       x es la señal de voz de entrada.
%       nmuest: número de muestras que tendrá  cada una de las tramas
%       en las que segmentemos la señal de voz.
%       nsolap: número de muestras de solapamiento entre trama y trama.
longx=length(x);
%       Calculamos el numero de tramas en que segmentaremos la señal
%       de voz
ntramas=ceil(longx/nsolap)-1;
%       Rellenamos con ceros el vector de voz hasta tener un número
%       entero de tramas
k=(longx+1):(ceil(longx/nsolap)*nsolap);
x(k)=zeros(size(k));
```

```
%           Segmentamos la señal de voz en trozos de nmuest muestras,
%           solapadas nsolap muestras
for i=1:ntramas
  for j=1:nmuest
          trama(i,j)=x(j+((i-1)*nsolap));
  end
end
%           Calculamos la ventana Hamming de nmuest puntos
n=0:(nmuest-1);
w=0.54-(0.46*cos(2*pi*n/(nmuest-1)));
%           Multiplico cada trama de voz por la ventana de Hamming de
%           nmuest puntos
for i=1:ntramas
     vector(i,:)=trama(i,:).*w;
end
%           Calculamos ncoef coeficientes de Fourier-Bessel según la
%           regla de Simpson para cada trama de voz enventanada vector
%           Calculamos los ceros de la función de Bessel de primera especie
ceros=cerosb2(1000);
for n=1:ntramas
     for m=1:ncoef
          xm=ceros(m);
          j=besselj(1,xm*(1:nmuest)/nmuest);
          c(n,m)=vector(n,:)*j'/(j*j');
     end
end
```

```
cm=c;
%          Reconstruimos la señal a partir de los coeficientes para
%          comprobar la eficiencia de este método
longitud=ceil(longx/nsolap)*nsolap;
i=1:longitud;
xrec=zeros(size(i));
n=1;
tram=1;
while(n<=(longitud-nmuest+1)),
    k=1;
    for m=1:nrec
        xm=ceros(m);
        xrec(n:(n+nmuest-1))=xrec(n:(n+nmuest-1))+...
         (c(tram,m).*besselj(1,xm*(1:nmuest)/nmuest));
    end
    n=n+nsolap;
    tram=tram+1;
end
vozrec=xrec;
```

Se puede observar que, en este caso, hemos implementado la transformada de Fourier-Bessel para el cálculo de coeficientes mediante la regla de Simpson. No obstante, en las restantes pruebas llevadas a cabo, se ha optado por simplificar la expresión que determina los coeficientes característicos, en aras de mejorar notablemente el tiempo de ejecución de las pruebas. Algunos ejemploss de los resutados de reconstrucción de la voz se muestran en las Figuras 6.1(a), 6.1(b), en las que se representa la señal de voz reconstruída superpuesta sobre la señal de voz original,

respectivamente para nrec=150 y 175, siendo nrec el número de coeficientes por trama empleados para la reconstrucción de la señal de voz.



(a) **(b)**

Figura 6.1: Comparación de la señal de voz original (rojo) con la señal reconstruída (azul) a partir de sus coeficientes de Fourier-Bessel. En este caso el número de coeficientes FB calculados es igual al número de coeficientes empleado en la reconstrucción ncoef=nrec: (a) nrec=150, (b) nrec=175.

Se observa en la Figura 6.1, que el número de muestras de la señal de voz que se toma para reconstruir es el correspondiente a una trama. Se puede también determinar a partir de las figuras, que se detectan claramente los picos y valles de la señal de voz; aunque no reproduzca exactamente la magnitud, la envolvente de la señal original es seguida casi con exactitud por la señal reconstruida. En la Figura 6.2 se muestra la reconstrucción de una declaración de 300 muestras, donde igualmente se advierte que los máximos y mínimos de la señal son reproducidos fielmente.

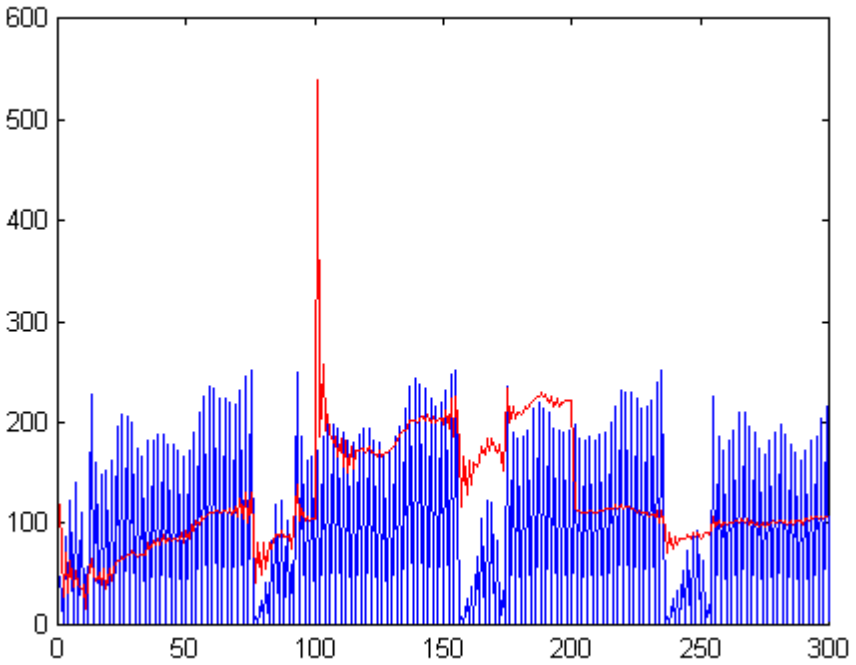


Figura 6.2: Comparación de la señal de voz original (azul) con la señal reconstruída (roja) a partir de sus coeficientes de Fourier-Bessel. En este caso el número de muestras de voz tomadas para su reconstrucción es de 300. nrec=175.

5.3: Pruebas de Reconocimiento de Conjunto Cerrado

Con el objetivo de obtener la combinación de parámetros que nos proporcione una tasa óptima de éxito en el reconocimiento, se han realizado multitud de pruebas, manteniendo, en cada caso, todos los parámetros con un valor constante y variando únicamente uno de ellos, para cada una de los subsistemas descritos en el capítulo anterior.

Del total de resultados obtenidos, cabe destacar el porcentaje de acierto de un 65%, fruto de la combinación de las funciones train6.m y test6.m, funciones que contemplaban el empleo de un solo micrófono, el micrófono 2 , tanto en la fase de

entrenamiento como en la de identificación, 5.000 muestras de voz por declaración, además del cálculo de 65 coeficientes FB centrados a las frecuencias correspondientes a los índices proporcionados por el vector

K2=[10,20,35,50,60]

Este resultado se ha obtenido dando a nuestros parámetros de libertad en el diseño del sistema los valores siguientes:

- N=20, siendo N el número de centroides tomados para la cuantización vectorial de los vectores característicos de energía del patrón correspondiente a cada locutor.
- Metodo=2, variable según la cual, la condición inicial del cuantizador vectorial para el cálculo de centroides consistía en tomar las N palabras de código de partida de forma aleatoria de entre el total de vectores característicos que componen el alfabeto de entrada.
- Calculo=3, de esta forma, el proceso de identificación trataba de obtener las distorsiones entre el locutor de prueba y los de entrenamiento mediante la comparación directa de los centroides obtenidos (20, en este caso), con el fin de asociar dos a dos aquellos cuya diferencia era mínima de forma asíncrona.

El resultado obtenido para la configuración proporcionada nos lleva a pensar que la mejor aplicación para el sistema diseñado es el reconocimiento de conjunto cerrado con independencia del texto, debido al hecho de haber escogido el vector de índices correspondiente a las frecuencias bajas del espectro de la voz, e igualmente, por no necesitar sincronismo alguno a nivel de tramas de voz para determinar la identidad del locutor de prueba.

Este hecho se corrobora por los resultados obtenidos para la misma configuración al disminuir el número de centroides en el que el cuantizador resume la información

característica del locutor, no obstante en menor medida, ya que se obtiene un resultado de identificación del 50% de éxito para N=10 y 15 centroides.

Advertimos asimismo, que emplear el tercer método de comparación mejora en gran medida la tasa de reconocimiento, pero realmente se consigue en combinación con la elección aleatoria del libro de código en el cuantizador, ya que, si hacemos la variable metodo=1, el porcentaje disminuye hasta un 40%, manteniendo la comparación asíncrona de centroides, e independientemente del número de centroides N.

En referencia a las aplicaciones de reconocimiento con dependencia del texto, señalar los resultados obtenidos a partir de las funciones train9.m y test9.m, que se basaban en grabaciones realizadas con el micrófono 2, en 5.000 muestras de voz por declaración, y en el cálculo de 195 coeficientes FB centrados en los índices del vector

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190]

para esta configuración, los mejores resultados obtenidos han sido de un 50%, principalmente en los casos que precisan sincronismo para el cómputo de la distorsión mínima, lo cual guarda una estrecha relación con la dependencia del texto en la identificación.

En las Tabla 5.1 se presentan los resultados más significativos obtenidos de las pruebas realizadas con train6.m y test6.m.

En las tablas siguientes, se detallan los porcentajes resultantes de otras configuraciones, cuyas características se explican en cada una de ellas.

En todos los casos, se presenta el tiempo de cálculo como un parámetro más obtenido en la fase de entrenamiento. Este tiempo, además de darnos una duración absoluta del proceso de carga de datos en el sistema, nos proporciona una medida relativa entre configuraciones, puesto que nos permite ver hasta qué punto merece la

pena transmitir una cierta cantidad de información mínima, siendo el tiempo de cálculo semejante si la información es mayor. Veremos que esto ocurre en determinadas ocasiones.

Porcentaje de éxito en el reconocimiento (%)	**Número de Centroides N=10**	**Número de Centroides N=15**	**Número de Centroides N=20**
Método de cálculo de la Distorsión calculo=1	-	-	-
Método de cálculo de la Distorsión calculo=2	-	-	-
Método de cálculo de la Distorsión calculo=3	40	40	40

Tabla 5.1(a): Resultados de los experimentos para train6.m y test6.m: Metodo=1, elección de la condición inicial a partir de los N primeros vectores característicos.

En la Tabla 5.1 aparecen algunas casillas con resultados de identificación no especificados. Esto también ocurrirá en algunos casos en las tablas siguientes, y es debido a que el porcentaje de reconocimiento obtenidos se considera lo suficientemente bajo como para suponer que los aciertos reportados se han producido de forma aleatoria. De ahí que no aparezcan reflejados en la tabla.

La función train6.m realiza la carga en la base de datos de declaraciones de 4 locutores, sumando un total de 20 declaraciones en la fase de entrenamiento, y 8 declaraciones en la fase de identificación. Puesto que el tiempo de identificación de un locutor no es significativo, y se realiza para un único locutor en cada ejecución del programa, consideramos que conviene evaluar el Tiempo de carga de los patrones

característicos de los locutores de referencia en la fase de entrenamiento. Para este caso concreto, el dado en la Tabla 5.1(a), el tiempo transcurrido hasta finalizar la carga en la base de datos fue de 20.7 segundos en el caso de 10 centroides, 21.5 en el caso de 15, y 24.5 para N=20 centroides.

Porcentaje de éxito en el reconocimiento (%)	**Número de Centroides N=10**	**Número de Centroides N=15**	**Número de Centroides N=20**
Método de cálculo de la Distorsión calculo=1	28	28	-
Método de cálculo de la Distorsión calculo=2	28	28	-
Método de cálculo de la Distorsión calculo=3	50	50	65

(b)

Tabla 5.1(b): Resultados de los experimentos para train6.m y test6.m: Metodo=2, elección aleatoria del libro de código de partida del VQ .

Se puede observar que las Tablas 5.2(a) y 5.2(b) son las que proporcionan los resultados más equiparados para todos los métodos propuestos; a pesar de que el óptimo no se ha obtenido en este caso, tampoco se ha producido resultado alguno despreciable como para considerarlo aleatorio. De esta observación se extrae que, aunque el funcionamiento no supera el 50% de acierto, se adivina que cualquier mejora aplicable en el futuro sobre el sistema repercutirá directamente sobre esta fase, la de reconocimiento de conjunto cerrado con dependencia del texto.

Porcentaje de éxito en el reconocimiento (%)	Centroides N=10	Centroides N=15	Centroides N=20
Distorsión calculo=1	50	40	40
Distorsión calculo=2	50	40	40
Distorsión calculo=3	28	50	50

(a)

Tabla 5.2(a): Resultados de los experimentos para train9.m y test9.m: Locutores grabados con micrófono 2, 5.000 muestras por declaración y 195 coeficientes de FB. Metodo=1, elección de la condición inicial a partir de los N primeros vectores característicos.

Porcentaje de éxito en el reconocimiento (%)	Centroides, N=10	Centroides, N=15	Centroides, N=20
Distorsión, calculo=1	40	40	40
Distorsión, calculo=2	50	40	50
Distorsión, calculo=3	40	28	40

(b)

Tabla 5.2(b): Resultados de los experimentos para train9.m y test9.m: Locutores grabados con micrófono 2, 5.000 muestras por declaración y 195 coeficientes de FB. Metodo=2, elección aleatoria del libro de código de partida del VQ .

Porcentaje de éxito en el reconocimiento (%)	Centroides N=10	Centroides N=15	Centroides N=20
Distorsión calculo=1	35	40	30
Distorsión calculo=2	35	40	30
Distorsión calculo=3	-	-	-

(a)

Tabla 5.3(a): Resultados de los experimentos para traina.m y testa.m: Todos los locutores de la base de datos grabados con ambos micrófonos, 5.000 muestras por declaración y 65 coeficientes de FB. Metodo=1, elección de la condición inicial a partir de los N primeros vectores característicos.

Porcentaje de éxito en el reconocimiento (%)	Centroides, N=10	Centroides, N=15	Centroides, N=20
Distorsión, calculo=1	25	25	30
Distorsión, calculo=2	25	25	25
Distorsión, calculo=3	50	35	30

(b)

Tabla 5.3(b): Resultados de los experimentos para traina.m y testa.m: Todos los locutores de la base de datos grabados con ambos micrófonos, 5.000 muestras por declaración y 65 coeficientes de FB. Metodo=2, elección aleatoria del libro de código de partida del VQ .

Porcentaje de éxito en el reconocimiento (%)	Centroides N=10	Centroides N=15	Centroides N=20
Distorsión calculo=1	35	30	25
Distorsión calculo=2	30	30	20
Distorsión calculo=3	25	-	-

(a)

Tabla 5.4(a): Resultados de los experimentos para train8.m y test8.m: Todos los locutores de la base de datos grabados con ambos micrófonos, 5.000 muestras por declaración y 195 coeficientes de FB. Metodo=1, elección de la condición inicial a partir de los N primeros vectores característicos.

Porcentaje de éxito en el reconocimiento (%)	Centroides, N=10	Centroides, N=15	Centroides, N=20
Distorsión, calculo=1	35	25	20
Distorsión, calculo=2	35	25	-
Distorsión, calculo=3	-	-	-

(b)

Tabla 5.4(b): Resultados de los experimentos para train8.m y test8.m: Todos los locutores de la base de datos grabados con ambos micrófonos, 5.000 muestras por declaración y 195 coeficientes de FB. Metodo=2, elección aleatoria del libro de código de partida del VQ .

Porcentaje de éxito en el reconocimiento (%)	Centroides N=10	Centroides N=15	Centroides N=20
Distorsión calculo=1	32	32	32
Distorsión calculo=2	32	20	20
Distorsión calculo=3	32	-	-

(a)

Tabla 5.5(a): Resultados de los experimentos para train1.m y test1.m: Locutores grabados con micrófono 1, 5.000 muestras por declaración y 195 coeficientes de FB. Metodo=1, elección de la condición inicial a partir de los N primeros vectores característicos.

Porcentaje de éxito en el reconocimiento (%)	Centroides, N=10	Centroides, N=15	Centroides, N=20
Distorsión, calculo=1	20	20	20
Distorsión, calculo=2	32	20	20
Distorsión, calculo=3	20	20	20

(b)

Tabla 5.5(b): Resultados de los experimentos para train1.m y test1.m: Locutores grabados con micrófono 1, 5.000 muestras por declaración y 195 coeficientes de FB. Metodo=2, elección aleatoria del libro de código de partida del VQ .

Porcentaje de éxito en el reconocimiento (%)	Centroides N=10	Centroides N=15	Centroides N=20
Distorsión calculo=1	20	20	20
Distorsión calculo=2	20	20	20
Distorsión calculo=3	-	-	-

(a)

Tabla 5.6(a): Resultados de los experimentos para train5.m y test5.m: Locutores grabados con micrófono 1, 5.000 muestras por declaración y 65 coeficientes de FB. Metodo=1, elección de la condición inicial a partir de los N primeros vectores característicos.

Porcentaje de éxito en el reconocimiento (%)	Centroides, N=10	Centroides, N=15	Centroides, N=20
Distorsión, calculo=1	20	20	20
Distorsión, calculo=2	20	20	20
Distorsión, calculo=3	-	20	32

(b)

Tabla 5.6(b): Resultados de los experimentos para train5.m y test5.m: Locutores grabados con micrófono 1, 5.000 muestras por declaración y 65 coeficientes de FB. Metodo=1, elección aleatoria del libro de código de partida del VQ .

A la vista de los resultados presentados, únicamente presentan un funcionamiento aceptable los sistemas que aparecen en las tablas 5.1 y 5.2, dando el resto de configuraciones un porcentaje final de éxito en el reconocimiento bastante bajo.

Tiempo de Cálculo (seg.)		**N=10**	**N=15**	**N=20**
Train1	Metodo=1	68.6	70	70.1
	Metodo=2	62.7	70.2	74.5
Train5	Metodo=1	31.1	34	34
	Metodo=2	26.2	32.3	33
Train6	Metodo=1	20.7	21.5	24.5
	Metodo=2	23.1	22.2	25.9
Train8	Metodo=1	103.6	108	110.6
	Metodo=2	102.9	118.9	112.8
Train9	Metodo=1	36.8	39	40.1
	Metodo=2	37.1	39.5	40.4
Traina	Metodo=1	41.6	41	42.9
	Metodo=2	39.3	39.8	42.6

Tabla 5.7: Tiempos de obtención de la base de datos en la fase de entrenamiento del sistema de reconocimiento para cada una de las configuraciones planteadas para la ejecución.

En la Tabla 5.7 se muestran los tiempos de cálculo para todas las configuraciones cuyos resultados de reconocimiento aparecen en las anteriores páginas, acompañados de la información referente a número de declaraciones, número de coeficientes a calcular por declaración y número de centroides en los que resumir la característica del locutor, información que justifica, de alguna manera, esta carga computacional y, por tanto, el posible retraso respecto a otros tipos de experimentos.

Características	**Número de declaraciones**	**Número de coeficientes FB**	**Número de muestras de la señal de voz**
Train1	35	195	5.000
Train5	35	65	5.000
Train6	20	65	5.000
Train8	55	195	5.000
Train9	20	195	5.000
Train9	55	65	5.000

Tabla 5.8: Características de cada uno de las configuraciones planteadas para el sistema de reconocimiento que influyen directamente en los tiempos de obtención de la base de datos en la fase de entrenamiento del sistema.

Contrastando los datos aportados en las tablas 5.7 y 5.8 comienzan a tener sentido los tiempos de ejecución que aparecían en la primera de ellas. No obstante, se observa que el incremento en el número de centroides no supone una excesiva ralentización en el funcionamiento del sistema, teniendo en cuenta que en aquellos casos la información

característica de los locutores disponible va a ser ciertamente superior a la que teníamos para un menor número de centroides. Habría que añadir que no sólo influye la elección del número de centroides a tomar en el sistema sobre el tiempo de cálculo de la base de datos del mismo, sino también, y de forma importante, en la necesidad del sistema de una mayor capacidad de almacenamiento de datos, factor que comentamos en el capítulo anterior. De ahí que se deba buscar una solución de compromiso que nos aporte un alto porcentaje de reconocimiento y, a la vez, eficiencia en la ejecución, determinada por tiempo de cómputo y por el proceso de almacenamiento de datos en la memoria del sistema.

Observar que, contrariamente a lo esperado, aumentar el número de centroides no ha mejorado significativamente la tasa del reconocimiento de voz, salvo en los casos puntuales anteriormente señalados. Plantearemos en el siguiente capítulo, posibles mejoras aplicables al sistema de reconocimiento implementado.

5.4: Pruebas de Reconocimiento de Conjunto Abierto

A pesar de que inicialmente se planteó realizar un análisis similar al realizado sobre el modo de conjunto cerrado en la modalidad de conjunto abierto, los resultados obtenidos en el apartado anterior desvelan el bajo rendimiento que podemos obtener de un sistema de reconocimiento de conjunto abierto de estas características. Se han realizado pruebas dependiendo de la expresión tomada para el umbral de rechazo, en términos de las distorsiones determinadas entre diferentes patrones del mismo locutor de referencia, variando igualmente N, número de centroides, calculo, método de comparación de los patrones característicos, y metodo, elección de la condición inicial en el VQ, pero sin llegar a una expresión del umbral (factor determinante en este modo de funcionamiento) que permita considerar no aleatorios los aciertos obtenidos durante las pruebas.

Capítulo 6: Conclusiones y Futuras Líneas de Investigación

6.1: Conclusiones

Se ha planteado un método alternativo al Cepstrum convencionalmente empleado para el Reconocimiento Automático de Locutores, que hace uso de las propiedades de la expansión en serie de Fourier-Bessel basándonos en el principio de que, tanto los segmentos de voz como las funciones de Bessel, presentan cuasiperiodicidad y amplitud decreciente con el tiempo. Por tanto, la representación de señales de voz usando estas funciones base puede terminar en menos componentes que usando las senoidales.

Dada la declaración de un locutor de referencia, la voz se segmenta en tramas con longitud lo suficientemente reducida para poder considerar en ellas la señal de voz estacionaria. Además, entre tramas de voz adyacentes se mantiene un solapamiento determinado que permita no perder la continuidad coherencia global de la señal de voz.

Aplicando la propiedad de ortogonalidad de las funciones de Bessel, se llega a una expresión compacta que determina los coeficientes característicos de Fourier-Bessel, de los cuales seleccionamos aquellos que corresponden a rangos de frecuencias críticas en el proceso de transmisión de la voz humana.

Estos coeficientes se calcularán para cada una de las tramas segmentadas de la declaración.

Con el objeto de reducir en cierta medida la cantidad de datos representativos de un locutor concreto, se adopta como patrón característico del locutor un vector para cada trama de voz cuyos elementos contienen la energía acumulada de cinco coeficientes de Fourier-Bessel consecutivos a partir de cada índice de los señalados como críticos.

Estos vectores de energía característicos se hacen pasar por un Cuantizador Vectorial que reduce la variabilidad de los datos a un determinado número de centroides que constituyen el libro de código o patrón de cada locutor.

Implementado este estudio, llegamos a resultados de identificación del orden del 65% en el modo de reconocimiento de conjunto cerrado, concretamente en el caso de considerar únicamente los coeficientes característicos del locutor que abarcan la banda de bajas frecuencias del espectro de voz humana, lo cual se considera una aplicación de reconocimiento con independencia del texto. Para otro tipo de configuraciones, consistentes en variar el número de centroides en el que resumir la información característica del locutor, el número de coeficientes FB a evaluar, el método de cálculo de la distancia entre locutores, o bien la elección de la condición inicial del cuantizador vectorial, los resultados obtenidos son del orden del 50% en el mejor de los casos.

En el caso de reconocimiento de conjunto abierto, y dados los resultados obtenidos en el modo de conjunto cerrado, no se ha determinado la expresión óptima del umbral que permita obtener una tasa de éxito fiable en la identificación.

Una posible mejora o vía alternativa en el desarrollo de este método podría consistir en realizar la cuantización vectorial de los coeficientes característicos de cada trama concreta, en lugar de hacerlo a lo largo del tiempo. La cuantización consistiría en extraer como centroides los coeficientes críticos de un solo vector característico de energía correspondiente a una trama, repitiendo el proceso para todas las tramas que constituyen la declaración. Se contaría de este modo con la información esencial de cada trama, que idealmente caracterizaría, de forma unívoca, un determinado fonema del mensaje.

No obstante, se plantean en los siguientes apartados algunas líneas de investigación sobre las que puede continuar el estudio de este método.

6.2: Futuras Líneas de Investigación

Aunque se han conseguido recientemente muchos avances y éxitos en el reconocimiento de locutores, aun existen muchos problemas para los que queda encontrar buenas soluciones. La mayoría de estos problemas surgen de la variabilidad, incluyendo la variabilidad generada por el locutor y la variabilidad en las condiciones de grabación y del canal. Es muy importante investigar parámetros característicos que sean estables en el tiempo, insensibles a la variación del modo de hablar, incluyendo la cadencia y nivel del habla, y robustos frente a variaciones en la calidad de la voz debidas a causas tales como cambio en la voz o resfriados. También es importante desarrollar un método para vencer al problema de la distorsión debida a los terminales telefónicos y canales, y a los ruidos de fondo y del canal.

Desde el punto de vista del interfaz humano, es importante considerar cómo se deberían proponer los textos a los usuarios, y cómo deberían tratarse los errores en el reconocimiento. Recientemente han surgido estudios sobre modos de extracción automática de los periodos de voz de cada persona separadamente de un diálogo que contiene a más de dos personas (Separación de fuentes de voz), como una extensión de la tecnología del reconocimiento de locutores.

En referencia a nuestro estudio concreto, planteamos a continuación posibles líneas de mejora sobre los métodos implementados, en concreto, sobre la cuantización vectorial, y sobre las características almacenadas de los locutores.

6.3: Técnicas de Normalización

El factor más significativo que afecta al funcionamiento del Reconocimiento Automático de Locutores (*Automatic Speaker Recognition,* ASR) es la variación de las características de la señal de una prueba a otra (variabilidad de intersección y variabilidad sobre el tiempo). Las variaciones surgen de los propios locutores, de diferencias en las condiciones de grabación y transmisión, y del ruido de fondo. Los locutores no pueden repetir una declaración precisamente de la misma forma de una prueba a otra. Es bien sabido que las muestras de la misma declaración grabada en una sesión están mucho más altamente correladas que muestras grabadas en sesiones diferentes. También hay cambios de las voces a largo plazo.

Es importante para los sistemas de reconocimiento de locutores adaptarse a estas variaciones. Se plantean dos tipos de técnicas de normalización; una en el dominio paramétrico, la otra, en el dominio de la distancia/similitud[1].

6.3.1: Normalización en el Dominio Paramétrico

La ecualización espectral, el conocido por método de ecualización ciega, es una técnica típica de normalización en el dominio paramétrico que ha sido confirmada como efectiva en la reducción de los efectos lineales del canal y de las variaciones espectrales a largo plazo. Este método es especialmente efectivo para aplicaciones de reconocimiento de locutores con dependencia del texto que emplean declaraciones lo suficientemente largas. Los coeficientes Cepstrum se promedian sobre la duración de

[1] *Estudio extraído del artículo "Speaker Recognition", desarrollado por Sadoki Furui, NTT Human Interface Laboratories, Tokyo, Japan, publicado en la dirección de internet* http://www.cslu.cse.ogi.edu/HLTsurvey/ch1node9.html *[12].*

una declaración completa y los valores promediados se restan a los coeficientes Cepstrum de cada trama. La variación aditiva en el dominio espectral logarítmico se puede compensar bastante bien con este método. Sin embargo, elimina inevitablemente algunas características específicas dependientes del texto y del locutor; por tanto, es inapropiada para declaraciones cortas en aplicaciones de reconocimiento de locutores.

6.3.2: Normalización en el Dominio Distancia/Similitud

Se propone un método de normalización por estimación de distancia (similitud) que emplea una relación de parecido. La relación de parecido se define como la relación de dos probabilidades condicionales de las medidas observadas de la declaración: la primera probabilidad es el parecido de los datos acústicos dada la identidad demandada del locutor, y la segunda es el parecido dado que el locutor es un impostor. La normalización por la relación de parecido aproxima el acierto óptimo según la ley de Bayes.

También se ha propuesto un método de normalización basado en una probabilidad *a posteriori*. La diferencia entre el método de normalización basado en la relación de parecido y el método basado en la probabilidad *a posteriori* es que el locutor esperado sea o no incluido en el conjunto para la normalización; el conjunto de locutores usado en el método basado en la relación de parecido no incluye al locutor esperado, mientras que el término de normalización para el método basado en la probabilidad *a posteriori*, se calcula empleando todos los locutores de referencia, incluido el locutor esperado.

Resultados experimentales indican que los dos métodos de normalización son casi igualmente efectivos. Ambos mejoran la separabilidad de locutores y reducen la necesidad de umbrales dependientes del texto o del locutor, como se comparó con la tasa usando únicamente el modelo del locutor correcto.

Recientemente se ha propuesto un nuevo método en el que el término de normalización se aproxima por la similitud de un solo modelo mixto que representa la distribución de parámetros para todos los locutores de referencia. Una ventaja de este

método es que el coste computacional del cálculo del término de normalización es muy pequeño, y se ha confirmado que este método da mucho mejores resultados que cualquiera de los métodos de normalización anteriormente mencionados.

6.4: Cuantización Binaria de Vectores Característicos para Identificación Robusta de Locutores con Texto Independiente

Presentamos una nueva aproximación a la cuantización de vectores en la que el vector característico viene representado por un vector binario. Se conoce como *Cuantización Binaria* $(BQ)^2$. El criterio de funcionamiento de la cuantización de vectores, la medida de la distorsión (distancia), se empleó para investigar la efectividad de BQ. Para tramas de análisis de 12 bits, la distorsión promedio causada por un BQ es aún menor que la distancia promedio intra-locutor entre dos repeticiones de la misma palabra (después del alineamiento DTW). Como la salida del BQ es una secuencia binaria, es posible combinarla con un Clasificador de Red de Hamming Directo. Con referencia a la idea del modelo jerárquico para describir las características individuales de un locutor, se desarrolló un sistema de identificación de locutores de texto independiente. Resultados experimentales muestran que el funcionamiento del sistema es muy bueno. No sólo priman los pequeños requerimientos de memoria y de cálculo en el sistema de identificación sino, de forma más importante, muestra fuerte robustez frente al ruido Gaussiano aditivo.

[2] *Estudio extraído del artículo "Binary Quantization of Feature Vectors for Robust Text-Independent Speaker Identification", elaborado por Zhong-Xuan Yuan, miembro asociado del IEEE, y Bo-Ling Xu, y perteneciente a la publicación IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 1, Enero 1999 [16].*

6.4.1: Introducción

La cuantización de vectores (VQ) es una técnica de compresión muy eficiente que proporciona baja tasa de codificación de voz. La bien conocida técnica de VQ es llamada el algoritmo de Linde-Buzo-Gray (LBG) [REF!!], que se basa en dos condiciones necesarias para la optimalidad:

- el centroide y
- el vecino más cercano.

El algoritmo LBG se usa para obtener un alfabeto de reproducción finito (libro de código), $\hat{x} = \{x_m; m = 1, \ldots, M_{VQ}\}$, representativo de una larga secuencia de entrenamiento del espectro de voz. El libro de código se diseña para minimizar la distorsión promedio que sure de representar esta secuencia de entrenamiento. Los VQ se han empleado tanto en codificadores de voz de pocos bits, como en sistemas de reconocimiento de voz (incluyendo el locutor). Al usar el VQ, se reduce la cantidad de datos a procesar.

Para mejorar la robustez de un sistema de reconocimiento de locutores y disminuir más la complejidad computacional y el almacenamiento, proponemos una nueva aproximación para cuantizar el vector característico, que se representa por un vector binario. Llamamos a este tipo de cuantización de vectores *Cuantización Binaria* (BQ). Se propone un nuevo modelo de reconocimiento de locutores, combinando el BQ con la Red de Hamming Directa. El modelo está motivado parcialmente por el éxito del reconocimiento de voz y locutores basado en el VQ. En el primer caso, Rabiner usó un cuantizador de vectores para transferir el conjunto de vectores característicos (LPC) a un conjunto de observación finito (los índices de las entradas del libro de código). El conjunto discreto y finito de observación se usó como datos de entrada al reconocedor del Modelo Oculto de Markov (*Hidden Markov Model*, HMM). En el segundo caso, Buck usó libros de código VQ multisección para reemplazar sus previos libros de código para reconocimiento de palabras aisladas (*Isolated Word Reconition*, IWR) y reconocimiento de locutores con texto dependiente. En este estudio, se usa el BQ para

encontrar un particionado del espacio de vectores característicos multidimensional $\{S_0, S_1, \ldots, S_{M_{BQ-1}}\}$. El espacio de vectores completo se representa por $S = S_0 \cup S_1 \cup \ldots \cup S_{M_{BQ-1}}$. Cada partición S_m constituye un hipercubo no superpuesto, y todo vector dentro de S_m se representa por un vector binario b_m. No sólo el vector binario b_m representa todo vector dentro de S_m, sino también el valor decimal de la secuencia binaria de b_m iguala al índice *m* de la partición S_m. El método de BQ integra la representación binaria de todo vector interior a la partición S_m con el índice de la partición, y separa la representación de los vectores dentro de la partición del centroide $\hat{x}_m$ de la partición. La propiedad especial del BQ hace posible usarlo como la entrada del Clasificador de Red de Hamming Directo. El vector centroide $\hat{x}_m$ de S_m sólo se usa como el vector de reproducción, un sustituto para cada vector interior. El conjunto de vectores centroide x_m, $m = 0, 1, \ldots, M_{BQ}$, se usará para investigar la distorsión promedio causada por el BQ. Sabemos que las características de la característica individual del locutor residen en su palabra hablada. En pequeños vocabularios de texto fijo, tal como los dígitos, el patrón de un locutor se describe por los vectores acústicos derivados de cada palabra, cada palabra se representa por una secuencia de vectores binarios ordenada en el tiempo. Se emplea un modelo jerárquico para implementar la identificación de locutores en el modo de texto independiente en vocabulario digital.

6.4.2: Cuantización Binaria

Dado un conjunto de vectores característicos de entrenamiento x_i, $i = 1, 2, \ldots, I$, en el que el espacio original Cartesiano de vectores multidimensional, el vector centroide x_c se deriva del conjunto de entrenamiento

$$x_{cj} = \frac{1}{I}\sum_{i=1}^{I} x_{ij} \qquad 1 \le j \le I, \tag{6.1}$$

donde J es el número de componentes de un vector característico, x_{cj} es la j-ésima componente de x_c e I es el número de vectores en la secuencia de entrenamiento. A continuación, el origen de coordenadas se traslada a x_c. De esta forma queda determinado el nuevo sistema de coordenadas, coordenadas del centro de masas. Hay *J* planos coordenados en el nuevo espacio Cartesiano de vectores. Los *J* planos coordenados, junto con el máximo y el mínimo valor de cada componente de los vectores característicos, forman un conjunto de particiones en el nuevo espacio de vectores Cartesiano, $\{S_0, S_1, \ldots, S_{MQ-1}\}$. Es obvio que M_{BQ} está determinado por J y $M_{BQ} = 2^J$. Cada partición S_m forma una región hipercúbica sin solapamiento. Todo vector característico dentro de S_m se representa por un vector binario b_m. El valor decimal de la secuencia binaria de b_m iguala al índice m de la partición S_m. En sistemas de identificación de locutores, un vector característico del análisis *front-end* se preprocesa y representa por un vector binario. El vector binario se usa como la entrada de un identificador de red de Hamming directo. Para obtener una representación binaria b_i adecuada para un vector x_i, se compara cada componente del vector, x_{ij}, con la correspondiente componente de x_c, x_{cj}

$$b_{ij} = \begin{cases} 1 & x_{ij} - x_{cj} \geq 0 \\ 0 & x_{ij} - x_{cj} < 0 \end{cases} \quad 1 \leq j \leq J \tag{6.2}$$

Cuando $x_{ij} - x_{cj} \geq 0$, la componente j-ésima x_{ij}, de x_i, se representa por un uno binario, es decir $b_{ij} = 1$. Cuando $x_{ij} - x_{cj} < 0$, la j-ésima componente x_{ij}, de x_i, se representa por un cero binario, es decir $b_{ij} = 0$. Así, el vector $\mathbf{x}_{ij}$ se puede representar por un vector binario $\{b_{ij}\}$. $\{b_{ij}\}$ va desde $\underbrace{\{00\ldots0\}}_{J}$ a $\underbrace{\{11\ldots1\}}_{J}$.

El valor decimal del vector binario $\{b_{ij}\}$ satisface la relación siguiente:

$$0 \le valor\ de\ \{b_{ij}\} \le 2^J - 1 \qquad (6.3)$$

El vector binario $\{b_{ij}\}$ corresponde justo al índice del hipercubo S_m al que pertenece el vector, x_i. Basándonos en la discusión anterior, el vector binario $\{b_{ij}\}$ se usa como la salida del cuantizador binario. También está claro que el valor decimal del vector binario, b_i, es justamente igual al índice del hipercubo S_m en el que x_i reside.

Promediando los vectores que pertenecen al conjunto de entrenamiento y caen en el interior del hipercubo m, se obtienen los vectores centroide $\hat{x}_m$, $m = 0, 1, \ldots, M_{BQ-1}$ de la región

$$\hat{x}_{mj} = \frac{1}{N_m} \sum_{i'=1}^{N_m} x_{i'j} \qquad 1 \le j \le J \qquad (6.4)$$

donde N_m es el número de vectores, $x_{i'}$, dentro del hipercubo m. Todo vector en S_m se reproduce por $\hat{x}_m$. El conjunto de $\hat{x}_m$ se usará para investigar la distorsión media provocada por el BQ en el apartado 6.3.3.

Comparando BQ con VQ, podemos ver que ambos cuantizadores se entrenan para encontrar particiones del espacio de vectores característicos. El cuantizador de vectores es entrenado para obtener el conjunto óptimo de vectores del libro de código $\hat{x}_m$, $m = 1, 2, \ldots, M_{VQ}$. A medida que se incrementa el nivel de cuantización M_{VQ}, se incrementa rápidamente la carga computacional para entrenar el cuantizador de vectores

óptimos. Como el pequeño decremento en la distorsión media de $M_{VQ} = 128$ a $M_{VQ} = 256$ no justificaba el incremento en la carga computacional correspondiente al gran libro de código, muchos sistemas de reconocimiento de voz (locutores incluidos) basados en el VQ, decidieron implementar el cuantizador de vectores usando un $M_{VQ} \leq 128$. Un cuantizador binario se entrena para encontrar el origen del nuevo sistema de coordenadas. La carga computacional para encontrar x_c en (6.1) únicamente iguala a aquélla del nivel 1 de entrenamiento $(M_{VQ} = 1)$ del cuantizador de vectores. El centroide $\hat{x}_m$ de cada partición S_m se obtiene promediando los vectores dentro de la región en términos de (6.4). Aparentemente, el procedimiento de cálculo es muy simple. Los procedimientos del algoritmo de división y cálculo desarrollados para producir un cuantizador de vectores óptimo de nivel más alto $(M_{VQ} > 1)$, lo que provocaba un gran número de operaciones, se omitieron en el entrenamiento de un cuantizador binario. Con el fin de obtener un buen funcionamiento de la cuantización, el espacio de vectores se divide en un gran número de vectores – muchas más que en el VQ, es decir $M_{BQ} >> M_{VQ}$-. Si el BQ se aplica a un sistema de codificación de voz, la debilidad del BQ es que el número de vectores centroide (palabras de código) en el BQ es mucho mayor que en el VQ. Afortunadamente, los vectores centroide se utilizan como vectores de reproducción cuando investigamos la distorsión causada por el BQ, porque el criterio de funcionamiento de la cuantización de vectores es la medida de la distorsión (distancia) [REF!!]. En el siguiente apartado, se usa esta distorsión media para investigar el funcionamiento del BQ.

6.4.3. Investigación de la Distorsión Media

Para tener un mejor entendimiento de la distorsiones medias provocadas por un cuantizador binario, se investigarán también las distorsiones medias provocadas por un cuantizador de vectores y las distancias promedia intra-locutor entre dos repeticiones de una palabra con alineamiento de torsión en tiempo dinámico (*Dynamic Time Warping*, DTW).

Asumimos que la distorsión provocada por reproducir un vector de entrada x_i por un vector de reproducción $\hat{x}_m$ viene dada por la medida de distancia Euclídea. Para cada declaración, se define la distorsión promedio provocada por el BQ como

$$\left\| D_{BQ} \right\| = \frac{1}{I} \sum_{i=1}^{I} d(\hat{x}_m, x_i) \qquad (6.5)$$

donde I es el número de vectores característicos en una declaración, y m es el valor decimal del vector binario b_m que denota al hipercubo m, en el que x_i reside. Si el número de componentes de un vector característico en una trama de análisis es 12 (por ejemplo, para una trama de 240 muestras), es decir $j = 12$ en (6.1)-(6.4), el valor binario del vector b_m va de 0000 0000 0000 a 1111 1111 1111 correspondiendo a 2^{12}=4096 hipercubos. $\hat{x}_m$ es el vector de reproducción (centroide) del hipercubo m, que viene determinado por (6.4). Si el hipercubo m está vacío, es decir $N_m = 0$ en (6.4), $\hat{x}_m$ se reemplaza por x_c derivado de (6.1). d denota la distorsión cuando un vector característico x_i en el hipercubo m es sustituido por el vector de reproducción $\hat{x}_m$ del hipercubo.

Para cada declaración de prueba, se define la distorsión media provocada por el cuantizador de vectores como

$$\left\| D_{VQ} \right\| = \frac{1}{I} \sum_{i=1}^{I} \min_{1 \le m \le M_{VQ}} \left[d(\hat{x}_m, x_i) \right] \qquad (6.6)$$

donde M_{VQ} es el nivel de cuantización del VQ.

La distancia promedio intra-locutor entre dos repeticiones de una misma palabra se define como

$$\|D_{DTW}\| = \frac{1}{I} \min_{v(i)} \sum_{i=1}^{I} d(x_i, y_{v(i)}) \tag{6.7}$$

donde $x_i = 1, 2, \ldots, I$ son los vectores característicos pertenecientes a una palabra. $y_m, m = v(i)$ y $m = 1, 2, \ldots, M$ son los vectores característicos que pertenecen a la declaración repetida de la misma palabra. Ambas declaraciones se realizan por el mismo locutor. El modo de alineamiento viene restringido por las condiciones de (6.8) y (6.9)

$$v(1) = 1 \qquad v(I) = M \tag{6.8}$$

$$v(i+1) - v(i) = \begin{cases} 0, 1, 2 & v(i) \neq v(i-1) \\ 1, 2 & v(i) = v(i-1) \end{cases} \tag{6.9}$$

Para cada declaración del conjunto de prueba, se calcula un $\|D_{BQ}\|$, un $\|D_{VQ}\|$ y un $\|D_{DTW}\|$. Se puede comprobar que la distorsión promedio de BQ es más baja que la de VQ. Además la distorsión promedio de BQ incluso cae por debajo de la distancia promedio intra-locutor entre dos repeticiones de la misma palabra con alineamiento DTW. Esto significa que la precisión de la cuantización binaria es lo suficientemente alta y el BQ puede llevarse a la práctica porque la gran variabilidad existe en las señales de voz, incluso en las repeticiones de la misma palabra pronunciadas por el mismo locutor. A la vista del patrón estadístico de reconocimiento, la distorsión promedio más

pequeña que la distancia promedio intra-locutor de repetición de la misma palabra no es muy importante ni llena de significado.

6.4.4: Esquema de Identificación de Locutores

En la Figura 6.1 se muestra un diagrama de bloques del sistema de reconocimiento de locutores basado en el BQHN. El patrón de un locutor viene descrito con el modelo jerárquico mostrado en la Figura 6.2.

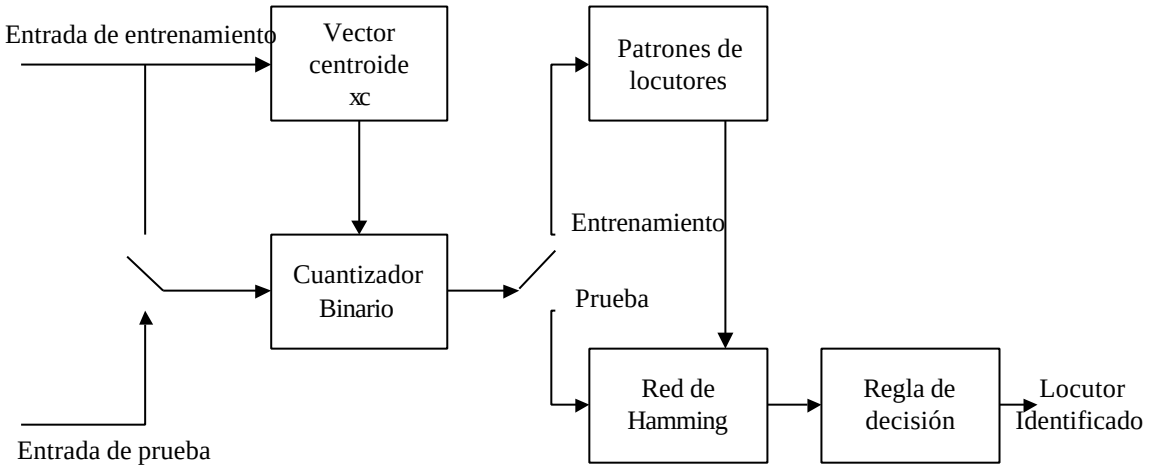


Figura 6.1: Diagrama de bloques de un identificador de locutores basado en BQHN

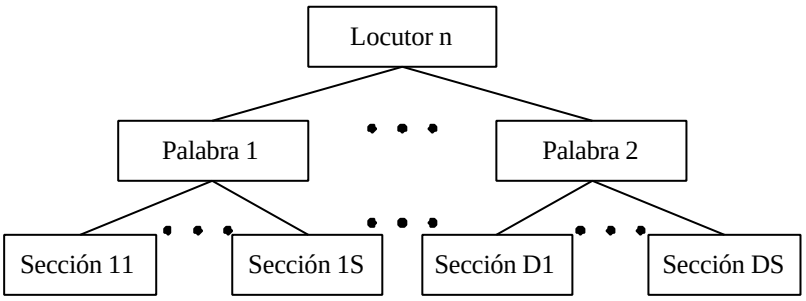


Figura 6.2: Modelo jerárquico para la descripción de las características individuales del locutor n.

Este modelo está basado en el hecho de que las características individuales del locutor residen en cada palabra hablada del vocabulario. Así, el reconocedor opera como identificador de texto independiente (en vocabulario digital). El identificador se ejecuta en primer lugar en un procedimiento de entrenamiento para proporcionar el vector centroide, x_c, para que el cuantizador binario y los vectores binarios contengan el patrón de cada locutor. A continuación, se muestran los pasos para formar los patrones de referencia binarios de cada palabra para el patrón de un locutor

1) Cada repetición de una palabra hablada pronunciada por el locutor en el conjunto de entrenamiento, se divide en S secciones en secuencia temporal. Si una declaración de entrenamiento tiene L tramas $(F_0, F_1, \ldots, F_{L-1}),$, el número de tramas de cada sección es

$$L_s = INT(L/S + 1) \tag{6.10}$$

La primera trama de cada sección F_S

$$F_S = \begin{cases} INT(L * s)/S & s = 0, 1, \ldots, S-2 \\ F_{L-1} - L_S & s = S-1 \end{cases} \tag{6.11}$$

En las secciones adyacentes, pueden existir una o dos tramas de solapamiento.

2) Se cargan los vectores característicos de las correspondientes secciones de las declaraciones de entrenamiento de la misma palabra dentro de un grupo,

entonces, para promediar los vectores de este grupo, se obtiene el vector centroide de la sección, x_S, $l \le s \le S$..

3) Se compara cada x_S con x_C para obtener la representación binaria, b_S, $1 \le s \le S,$, de la sección s en términos de (6.2)

4) Disponer el vector binario S b_S en una secuencia binaria que se usa como la referencia r_d de la palabra d. Cada componente de r_d es r_{dk}, $1 \le k \le K = S \times J$, $1 \le d \le D,$, donde S es el número de secciones de una palabra hablada, J es el número de componentes de un vector característico y D es el número de palabras del vocabulario.

Estos pasos se repetirán para toda palabra del vocabulario para el patrón de cada uno de los locutores hasta que se determinen los patrones del total de locutores.

En la fase de identificación, se dividen, en primer lugar, los vectores característicos del locutor desconocido en S secciones. Después de que los vectores de cada sección se hayan promediado, se obtiene el vector centroide de cada sección. Los vectores centroide se envían al cuantizador binario para obtener la representación binaria de los vectores característicos de entrada i_k, $1 \le k \le K = S \times J$, tal como se hizo en la fase de entrenamiento. Entonces, se mide la distancia entre cada palabra de referencia, r_{dk}, del patrón de un locutor y la secuencia de entrada binaria, i_k para dar lugar a una distancia para cada palabra de referencia usando la distancia de Hamming modificada de la red de Hamming directa

$$\left\| D_{Hamming} \right\|_d = \sum_{k=1}^{K} r_{dk} \oplus i_k \qquad (6.12)$$

donde $\oplus$ es el operador OR exclusiva. De (6.12) podemos ver que $\left\| D_{Hamming} \right\|_d$ iguala al número de componentes diferentes entre la palabra de referencia r_d y el patrón de entrada binario i_k. La distancia más pequeña se elige como la salida del patrón del locutor. El proceso y la red de Hamming directa se ilustran en la Figura 6.3.

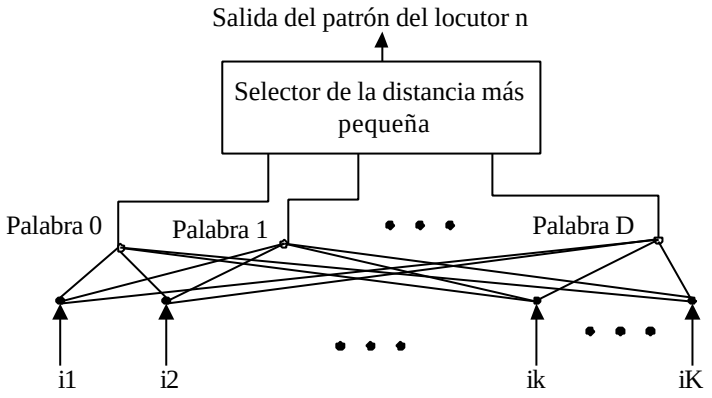


Figura 6.3: Ilustración de la Red de Hamming Directa y medida de la distancia entre la palabra de referencia $\mathbf{r_{dk}}$ y el patrón de entrada binario $\mathbf{i_k}$.

La regla de decisión elige al locutor cuyo patrón da la distancia más pequeña

$$locutor^{*} = \arg \min_{1 \le i \le N} \left\{ \min_{1 \le d \le D} \left\| D_{Hamming} \right\|_d^i \right\} \qquad (6.13)$$

donde D es el número de palabras del vocabulario y N, el número de locutores registrados en el sistema.

6.4.5: Estimación del Almacenamiento y Cálculo

Además del funcionamiento robusto del identificador basado en el BQHN con ruido blanco Gaussiano aditivo, merece la pena estimar 'grosso modo' el almacenamiento y la carga computacional que se necesitan para diseñar el cuantizador binario, en la fase de cuantización y en la fase de identificación. Para diseñar el cuantizador binario, que lo que necesita hacer es evaluar el vector centroide, x_C, en (6.1), se requieren J(I-1) sumas y J divisiones. Sólo se necesitan 4 • J = 4 • 12 =48, J =12 y 4 bytes para almacenar un dato en punto flotante, los bytes de almacenamiento se requieren para guardar J componentes en punto flotante del vector, donde J es el número de componentes de un vector, e I es el número de vectores en un conjunto de entrenamiento.

Para obtener la representación binaria de un vector característico, solamente se requieren J comparaciones entre el vector de entrada y el vector centroide, x_C, [véase (6.2)]. Si S es el número de secciones en el que se divide una palabra hablada, D es el número de palabras del vocabulario y N es el número de locutores registrados en el identificador, se requieren S • D • J • N bits para guardar los patrones. Por ejemplo, S=10, D=10, J=12, N=42,

S • D • J • N = 10 • 10 • 12 • 42 = 50400 bits = 6300 bytes

Para la evaluación de la distancia de/a un locutor en una red de Hamming directa en la fase de identificación, se necesitan

S • J • D = 1200 operaciones XOR

y D=10 comparaciones. La regla de decisión necesita N=42 comparaciones.

Se debería advertir que en nuestra ruda evaluación de la carga computacional, no hemos incluido el cómputo del análisis *front-end*. Este cálculo es necesario en cualesquiera otros identificadores. Si comparamos el cálculo y el almacenamiento requeridos en el identificador basado en el BQHN con los requeridos en cualesquiera otros identificadores basados en el VQ, podemos ver que el cálculo y almacenamiento necesitados en el identificador basado en el BQHN es mucho más pequeño.

6.4.6: Sumario

Hemos propuesto una aproximación para cuantizar vectores característicos: El Cuantizador Binario. Para demostrar la efectividad del BQ, se ha empleado el criterio de funcionamiento de la cuantización de vectores, la medida de distorsión (distancia). La distorsión promedio causada por el BQ es más baja que la causada por el VQ cuando $M_{VQ} = 128$ e incluso más baja que la distancia promedio intra-locutor entre dos repeticiones de la misma palabra.

Como la salida del BQ es una secuencia binaria, nos es posible combinarla con el clasificador de red de Hamming directo. En relación con la idea de modelo jerárquico para describir las características individuales de un locutor, se diseña un nuevo sistema de identificación de locutores que opera como identificador de locutores de texto independiente en vocabulario digital. Los resultados experimentales muestran que el funcionamiento del identificador es muy bueno. Además del funcionamiento robusto del identificador basado en el BQHN, la pequeña carga computacional y pequeño almacenamiento requeridos en el identificador son las otras dos ventajas.

Capítulo 7: Planos:Listado de los Programas

7.1: Fase de Entrenamiento

7.1.1: Generación de la Base de Datos

```
%%%%FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN1)%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto cerrado (diferentes locutores, un solo micrófono,
%mic1, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos
%de los locutores llamando a vtrain.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz
%
nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;
```

```
%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%centr: base de datos de vectores centroide de cada declaración

centr=[];

%bdatos: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos=[];

%bdloc: base de datos que asocia a cada locutor un número

bdloc=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Número de centroides

N=10;

[centr(1:N,1:15),bdatos(1,1:49)]=vtrain('c:\matlab\lina11.wav',nmuest,nsolap,ncoef,k,
5101,10100);
bdloc(1,1:18)='El locutor es Lina';
```

```
[centr(1:N,16:30),bdatos(2,1:49)]=vtrain('c:\matlab\lina12.wav',nmuest,nsolap,ncoef,k
,4501,9500);
bdloc(2,1:18)='El locutor es Lina';

[centr(1:N,31:45),bdatos(3,1:49)]=vtrain('c:\matlab\lina13.wav',nmuest,nsolap,ncoef,k
,7601,12600);
bdloc(3,1:18)='El locutor es Lina';

[centr(1:N,46:60),bdatos(4,1:49)]=vtrain('c:\matlab\lina14.wav',nmuest,nsolap,ncoef,k
,5001,10000);
bdloc(4,1:18)='El locutor es Lina';

[centr(1:N,61:75),bdatos(5,1:49)]=vtrain('c:\matlab\lina15.wav',nmuest,nsolap,ncoef,k
,6501,11500);
bdloc(5,1:18)='El locutor es Lina';

[centr(1:N,76:90),bdatos(6,1:49)]=vtrain('c:\matlab\jordi11.wav',nmuest,nsolap,ncoef,
k,2301,7300);
bdloc(6,1:19)='El locutor es Jordi';

[centr(1:N,91:105),bdatos(7,1:49)]=vtrain('c:\matlab\jordi12.wav',nmuest,nsolap,ncoef
,k,3001,8000);
bdloc(7,1:19)='El locutor es Jordi';

[centr(1:N,106:120),bdatos(8,1:49)]=vtrain('c:\matlab\jordi13.wav',nmuest,nsolap,nco
ef,k,3501,8500);
bdloc(8,1:19)='El locutor es Jordi';

[centr(1:N,121:135),bdatos(9,1:49)]=vtrain('c:\matlab\jordi14.wav',nmuest,nsolap,nco
ef,k,7601,12600);
bdloc(9,1:19)='El locutor es Jordi';
```

```
[centr(1:N,136:150),bdatos(10,1:49)]=vtrain('c:\matlab\jordi15.wav',nmuest,nsolap,ncoef,k,3001,8000);
bdloc(10,1:19)='El locutor es Jordi';

[centr(1:N,151:165),bdatos(11,1:49)]=vtrain('c:\matlab\yo11.wav',nmuest,nsolap,ncoef,k,6251,11250);
bdloc(11,1:21)='El locutor es Arancha';

[centr(1:N,166:180),bdatos(12,1:49)]=vtrain('c:\matlab\yo12.wav',nmuest,nsolap,ncoef,k,7951,12950);
bdloc(12,1:21)='El locutor es Arancha';

[centr(1:N,181:195),bdatos(13,1:49)]=vtrain('c:\matlab\yo13.wav',nmuest,nsolap,ncoef,k,7751,12750);
bdloc(13,1:21)='El locutor es Arancha';

[centr(1:N,196:210),bdatos(14,1:49)]=vtrain('c:\matlab\yo14.wav',nmuest,nsolap,ncoef,k,7501,12500);
bdloc(14,1:21)='El locutor es Arancha';

[centr(1:N,211:225),bdatos(15,1:49)]=vtrain('c:\matlab\yo15.wav',nmuest,nsolap,ncoef,k,4501,9500);
bdloc(15,1:21)='El locutor es Arancha';

[centr(1:N,256:270),bdatos(16,1:49)]=vtrain('c:\matlab\juanjo11.wav',nmuest,nsolap,ncoef,k,2201,7200);
bdloc(16,1:20)='El locutor es Juanjo';

[centr(1:N,271:285),bdatos(17,1:49)]=vtrain('c:\matlab\juanjo12.wav',nmuest,nsolap,ncoef,k,11501,16500);
bdloc(17,1:20)='El locutor es Juanjo';
```

```
[centr(1:N,286:300),bdatos(18,1:49)]=vtrain('c:\matlab\juanjo13.wav',nmuest,nsolap,n
coef,k,10001,15000);
bdloc(18,1:20)='El locutor es Juanjo';

[centr(1:N,301:315),bdatos(19,1:49)]=vtrain('c:\matlab\juanjo14.wav',nmuest,nsolap,n
coef,k,11751,16750);
bdloc(19,1:20)='El locutor es Juanjo';

[centr(1:N,316:330),bdatos(20,1:49)]=vtrain('c:\matlab\juanjo15.wav',nmuest,nsolap,n
coef,k,9901,14900);
bdloc(20,1:20)='El locutor es Juanjo';

[centr(1:N,376:390),bdatos(21,1:49)]=vtrain('c:\matlab\angel11.wav',nmuest,nsolap,n
coef,k,7001,12000);
bdloc(21,1:19)='El locutor es Angel';

[centr(1:N,391:405),bdatos(22,1:49)]=vtrain('c:\matlab\angel12.wav',nmuest,nsolap,n
coef,k,6501,11500);
bdloc(22,1:19)='El locutor es Angel';

[centr(1:N,406:420),bdatos(23,1:49)]=vtrain('c:\matlab\angel13.wav',nmuest,nsolap,n
coef,k,3801,8800);
bdloc(23,1:19)='El locutor es Angel';

[centr(1:N,421:435),bdatos(24,1:49)]=vtrain('c:\matlab\angel14.wav',nmuest,nsolap,n
coef,k,4101,9100);
bdloc(24,1:19)='El locutor es Angel';

[centr(1:N,436:450),bdatos(25,1:49)]=vtrain('c:\matlab\angel15.wav',nmuest,nsolap,n
coef,k,3901,8900);
bdloc(25,1:19)='El locutor es Angel';
```

```
[centr(1:N,451:465),bdatos(26,1:49)]=vtrain('c:\matlab\carmen11.wav',nmuest,nsolap,
ncoef,k,4501,9500);
bdloc(26,1:20)='El locutor es Carmen';

[centr(1:N,466:480),bdatos(27,1:49)]=vtrain('c:\matlab\carmen12.wav',nmuest,nsolap,
ncoef,k,6301,11300);
bdloc(27,1:20)='El locutor es Carmen';

[centr(1:N,481:495),bdatos(28,1:49)]=vtrain('c:\matlab\carmen13.wav',nmuest,nsolap,
ncoef,k,6901,11900);
bdloc(28,1:20)='El locutor es Carmen';

[centr(1:N,496:510),bdatos(29,1:49)]=vtrain('c:\matlab\carmen14.wav',nmuest,nsolap,
ncoef,k,6301,11300);
bdloc(29,1:20)='El locutor es Carmen';

[centr(1:N,511:525),bdatos(30,1:49)]=vtrain('c:\matlab\carmen15.wav',nmuest,nsolap,
ncoef,k,3001,8000);
bdloc(30,1:20)='El locutor es Carmen';

[centr(1:N,526:540),bdatos(31,1:49)]=vtrain('c:\matlab\monica11.wav',nmuest,nsolap,
ncoef,k,5501,10500);
bdloc(31,1:20)='El locutor es Monica';

[centr(1:N,541:555),bdatos(32,1:49)]=vtrain('c:\matlab\monica12.wav',nmuest,nsolap,
ncoef,k,5101,10100);
bdloc(32,1:20)='El locutor es Monica';

[centr(1:N,556:570),bdatos(33,1:49)]=vtrain('c:\matlab\monica13.wav',nmuest,nsolap,
ncoef,k,3001,8000);
bdloc(33,1:20)='El locutor es Monica';
```

```
[centr(1:N,571:585),bdatos(34,1:49)]=vtrain('c:\matlab\monica14.wav',nmuest,nsolap,
ncoef,k,2701,7700);
bdloc(34,1:20)='El locutor es Monica';

[centr(1:N,586:600),bdatos(35,1:49)]=vtrain('c:\matlab\monica15.wav',nmuest,nsolap,
ncoef,k,3301,8300);
bdloc(35,1:20)='El locutor es Monica';

save bdloc
save centr
save bdatos
```

```
%%%%FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN2)%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto abierto (un único locutor, dos micrófonos, 5.000
%muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los
%locutores llamando a vtrain.m y a cumbral.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada
```

```
nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
% para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%centr2: base de datos de vectores centroide de cada declaración

centr2=[];

%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos2=[];

%bdloc2: base de datos que asocia a cada locutor un número

bdloc2=[];

%umbral: máxima distorsión permitida entre patrones de un mismo
%locutor de referencia.

umbral=[];

%Añadimos cada una de las declaraciones de referencia de los
```

```
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro
%Número de centroides

N=10;

%Método de cálculo de la distorsión

calculo=1;

[centr2(1:N,1:15),bdatos2(1,1:49)]=vtrain('c:\matlab\yo11.wav',nmuest,nsolap,ncoef,k,
6251,11250);
bdloc2(1,1:21)='El locutor es Arancha';

[centr2(1:N,16:30),bdatos2(2,1:49)]=vtrain('c:\matlab\yo12.wav',nmuest,nsolap,ncoef,
k,7951,12950);
bdloc2(2,1:21)='El locutor es Arancha';

[centr2(1:N,31:45),bdatos2(3,1:49)]=vtrain('c:\matlab\yo13.wav',nmuest,nsolap,ncoef,
k,7751,12750);
bdloc2(3,1:21)='El locutor es Arancha';

[centr2(1:N,46:60),bdatos2(4,1:49)]=vtrain('c:\matlab\yo14.wav',nmuest,nsolap,ncoef,
k,7501,12500);
bdloc2(4,1:21)='El locutor es Arancha';

[centr2(1:N,61:75),bdatos2(5,1:49)]=vtrain('c:\matlab\yo15.wav',nmuest,nsolap,ncoef,
k,4501,9500);
bdloc2(5,1:21)='El locutor es Arancha';
```

```
[centr2(1:N,76:90),bdatos2(6,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,n
coef,k,6001,11000);
bdloc2(6,1:21)='El locutor es Arancha';

[centr2(1:N,91:105),bdatos2(7,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,
ncoef,k,7201,12200);
bdloc2(7,1:21)='El locutor es Arancha';

[centr2(1:N,106:120),bdatos2(8,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap
,ncoef,k,6401,11400);
bdloc2(8,1:21)='El locutor es Arancha';

[centr2(1:N,121:135),bdatos2(9,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap
,ncoef,k,7001,12000);
bdloc2(9,1:21)='El locutor es Arancha';

[centr2(1:N,136:150),bdatos2(10,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsola
p,ncoef,k,6201,11200);
bdloc2(10,1:21)='El locutor es Arancha';

save bdloc2
save centr2
save bdatos2
umbral=cumbral(centr2,bdatos2,calculo);
save umbral
```

*%%%%**FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN3)**%%%%*
%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto abierto (un único locutor, un solo micrófono, mic2,
%5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los
%locutores llamando a vtrain.m y a cumbral.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
% para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%centr2: base de datos de vectores centroide de cada declaración

centr2=[];

```
%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos2=[];

%bdloc2: base de datos que asocia a cada locutor un número

bdloc2=[];

% umbral: máxima distorsión permitida entre patrones de un mismo
%locutor de referencia.

umbral=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

% Número de centroides

N=10;

%Método de cálculo de la distorsión

calculo=1;


[centr2(1:N,1:15),bdatos2(1,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,nc
oef,k,6001,11000);
bdloc2(1,1:21)='El locutor es Arancha';
```

```
[centr2(1:N,16:30),bdatos2(2,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,n
coef,k,7201,12200);
bdloc2(2,1:21)='El locutor es Arancha';

[centr2(1:N,31:45),bdatos2(3,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap,n
coef,k,6401,11400);
bdloc2(3,1:21)='El locutor es Arancha';

[centr2(1:N,46:60),bdatos2(4,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap,n
coef,k,7001,12000);
bdloc2(4,1:21)='El locutor es Arancha';

[centr2(1:N,61:75),bdatos2(5,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap,n
coef,k,6201,11200);
bdloc2(5,1:21)='El locutor es Arancha';


save bdloc2
save centr2
save bdatos2
umbral=cumbral(centr2,bdatos2,calculo);
save umbral
```

*%%%%**FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN4)**%%%%*
%%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema %de reconocimiento para conjunto abierto (un único locutor, un solo micrófono, mic2, %5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los %locutores llamando a vtrain.m y a cumbral.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%centr2: base de datos de vectores centroide de cada declaración

centr2=[];

```
%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos2=[];

%bdloc2: base de datos que asocia a cada locutor un número

bdloc2=[];

% umbral: máxima distorsión permitida entre patrones de un mismo
%locutor de referencia.

umbral=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

% Número de centroides

N=10;

%Método de cálculo de la distorsión

calculo=1;

[centr2(1:N,1:5),bdatos2(1,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,nco
ef2,k2,6001,11000);
bdloc2(1,1:21)='El locutor es Arancha';
```

```
[centr2(1:N,6:10),bdatos2(2,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,nc
oef2,k2,7201,12200);
bdloc2(2,1:21)='El locutor es Arancha';

[centr2(1:N,11:15),bdatos2(3,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap,n
coef2,k2,6401,11400);
bdloc2(3,1:21)='El locutor es Arancha';

[centr2(1:N,16:20),bdatos2(4,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap,n
coef2,k2,7001,12000);
bdloc2(4,1:21)='El locutor es Arancha';

[centr2(1:N,21:25),bdatos2(5,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap,n
coef2,k2,6201,11200);
bdloc2(5,1:21)='El locutor es Arancha';

save bdloc2
save centr2
save bdatos2
umbral=cumbral(centr2,bdatos2,calculo);
save umbral
```

```
%%%%FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN5)%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto cerrado (diferentes locutores, un solo micrófono,
%mic1, 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos
%de los locutores llamando a vtrain.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%centr: base de datos de vectores centroide de cada declaración

centr=[];
```

```
%bdatos: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos=[];

%bdloc: base de datos que asocia a cada locutor un número

bdloc=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Número de centroides

N=10;

[centr(1:N,1:5),bdatos(1,1:49)]=vtrain('c:\matlab\lina11.wav',nmuest,nsolap,ncoef2,k2
,5101,10100);
bdloc(1,1:18)='El locutor es Lina';

[centr(1:N,6:10),bdatos(2,1:49)]=vtrain('c:\matlab\lina12.wav',nmuest,nsolap,ncoef2,k
2,4501,9500);
bdloc(2,1:18)='El locutor es Lina';

[centr(1:N,11:15),bdatos(3,1:49)]=vtrain('c:\matlab\lina13.wav',nmuest,nsolap,ncoef2,
k2,7601,12600);
bdloc(3,1:18)='El locutor es Lina';

[centr(1:N,16:20),bdatos(4,1:49)]=vtrain('c:\matlab\lina14.wav',nmuest,nsolap,ncoef2,
k2,5001,10000);
```

```
bdloc(4,1:18)='El locutor es Lina';

[centr(1:N,21:25),bdatos(5,1:49)]=vtrain('c:\matlab\lina15.wav',nmuest,nsolap,ncoef2,
k2,6501,11500);
bdloc(5,1:18)='El locutor es Lina';

[centr(1:N,26:30),bdatos(6,1:49)]=vtrain('c:\matlab\jordi11.wav',nmuest,nsolap,ncoef2
,k2,2301,7300);
bdloc(6,1:19)='El locutor es Jordi';

[centr(1:N,31:35),bdatos(7,1:49)]=vtrain('c:\matlab\jordi12.wav',nmuest,nsolap,ncoef2
,k2,3001,8000);
bdloc(7,1:19)='El locutor es Jordi';

[centr(1:N,36:40),bdatos(8,1:49)]=vtrain('c:\matlab\jordi13.wav',nmuest,nsolap,ncoef2
,k2,3501,8500);
bdloc(8,1:19)='El locutor es Jordi';

[centr(1:N,41:45),bdatos(9,1:49)]=vtrain('c:\matlab\jordi14.wav',nmuest,nsolap,ncoef2
,k2,7601,12600);
bdloc(9,1:19)='El locutor es Jordi';

[centr(1:N,46:50),bdatos(10,1:49)]=vtrain('c:\matlab\jordi15.wav',nmuest,nsolap,ncoef
2,k2,3001,8000);
bdloc(10,1:19)='El locutor es Jordi';

[centr(1:N,51:55),bdatos(11,1:49)]=vtrain('c:\matlab\yo11.wav',nmuest,nsolap,ncoef2,
k2,6251,11250);
bdloc(11,1:21)='El locutor es Arancha';
```

```
[centr(1:N,56:60),bdatos(12,1:49)]=vtrain('c:\matlab\yo12.wav',nmuest,nsolap,ncoef2,
k2,7951,12950);
bdloc(12,1:21)='El locutor es Arancha';

[centr(1:N,61:65),bdatos(13,1:49)]=vtrain('c:\matlab\yo13.wav',nmuest,nsolap,ncoef2,
k2,7751,12750);
bdloc(13,1:21)='El locutor es Arancha';

[centr(1:N,66:70),bdatos(14,1:49)]=vtrain('c:\matlab\yo14.wav',nmuest,nsolap,ncoef2,
k2,7501,12500);
bdloc(14,1:21)='El locutor es Arancha';

[centr(1:N,71:75),bdatos(15,1:49)]=vtrain('c:\matlab\yo15.wav',nmuest,nsolap,ncoef2,
k2,4501,9500);
bdloc(15,1:21)='El locutor es Arancha';

[centr(1:N,86:90),bdatos(16,1:49)]=vtrain('c:\matlab\juanjo11.wav',nmuest,nsolap,nco
ef2,k2,2201,7200);
bdloc(16,1:20)='El locutor es Juanjo';

[centr(1:N,91:95),bdatos(17,1:49)]=vtrain('c:\matlab\juanjo12.wav',nmuest,nsolap,nco
ef2,k2,11501,16500);
bdloc(17,1:20)='El locutor es Juanjo';

[centr(1:N,96:100),bdatos(18,1:49)]=vtrain('c:\matlab\juanjo13.wav',nmuest,nsolap,nc
oef2,k2,10001,15000);
bdloc(18,1:20)='El locutor es Juanjo';

[centr(1:N,101:105),bdatos(19,1:49)]=vtrain('c:\matlab\juanjo14.wav',nmuest,nsolap,n
coef2,k2,11751,16750);
bdloc(19,1:20)='El locutor es Juanjo';
```

```
[centr(1:N,106:110),bdatos(20,1:49)]=vtrain('c:\matlab\juanjo15.wav',nmuest,nsolap,n
coef2,k2,9901,14900);
bdloc(20,1:20)='El locutor es Juanjo';

[centr(1:N,126:130),bdatos(21,1:49)]=vtrain('c:\matlab\angel11.wav',nmuest,nsolap,n
coef2,k2,7001,12000);
bdloc(21,1:19)='El locutor es Angel';

[centr(1:N,131:135),bdatos(22,1:49)]=vtrain('c:\matlab\angel12.wav',nmuest,nsolap,n
coef2,k2,6501,11500);
bdloc(22,1:19)='El locutor es Angel';

[centr(1:N,136:140),bdatos(23,1:49)]=vtrain('c:\matlab\angel13.wav',nmuest,nsolap,n
coef2,k2,3801,8800);
bdloc(23,1:19)='El locutor es Angel';

[centr(1:N,141:145),bdatos(24,1:49)]=vtrain('c:\matlab\angel14.wav',nmuest,nsolap,n
coef2,k2,4101,9100);
bdloc(24,1:19)='El locutor es Angel';

[centr(1:N,146:150),bdatos(25,1:49)]=vtrain('c:\matlab\angel15.wav',nmuest,nsolap,n
coef2,k2,3901,8900);
bdloc(25,1:19)='El locutor es Angel';

[centr(1:N,151:155),bdatos(26,1:49)]=vtrain('c:\matlab\carmen11.wav',nmuest,nsolap,
ncoef2,k2,4501,9500);
bdloc(26,1:20)='El locutor es Carmen';

[centr(1:N,156:160),bdatos(27,1:49)]=vtrain('c:\matlab\carmen12.wav',nmuest,nsolap,
ncoef2,k2,6301,11300);
bdloc(27,1:20)='El locutor es Carmen';
```

```
[centr(1:N,161:165),bdatos(28,1:49)]=vtrain('c:\matlab\carmen13.wav',nmuest,nsolap,
ncoef2,k2,6901,11900);
bdloc(28,1:20)='El locutor es Carmen';

[centr(1:N,166:170),bdatos(29,1:49)]=vtrain('c:\matlab\carmen14.wav',nmuest,nsolap,
ncoef2,k2,6301,11300);
bdloc(29,1:20)='El locutor es Carmen';

[centr(1:N,171:175),bdatos(30,1:49)]=vtrain('c:\matlab\carmen15.wav',nmuest,nsolap,
ncoef2,k2,3001,8000);
bdloc(30,1:20)='El locutor es Carmen';

[centr(1:N,176:180),bdatos(31,1:49)]=vtrain('c:\matlab\monica11.wav',nmuest,nsolap,
ncoef2,k2,5501,10500);
bdloc(31,1:20)='El locutor es Monica';

[centr(1:N,181:185),bdatos(32,1:49)]=vtrain('c:\matlab\monica12.wav',nmuest,nsolap,
ncoef2,k2,5101,10100);
bdloc(32,1:20)='El locutor es Monica';

[centr(1:N,186:190),bdatos(33,1:49)]=vtrain('c:\matlab\monica13.wav',nmuest,nsolap,
ncoef2,k2,3001,8000);
bdloc(33,1:20)='El locutor es Monica';

[centr(1:N,191:195),bdatos(34,1:49)]=vtrain('c:\matlab\monica14.wav',nmuest,nsolap,
ncoef2,k2,2701,7700);
bdloc(34,1:20)='El locutor es Monica';

[centr(1:N,196:200),bdatos(35,1:49)]=vtrain('c:\matlab\monica15.wav',nmuest,nsolap,
ncoef2,k2,3301,8300);
bdloc(35,1:20)='El locutor es Monica';
```

save bdloc
save centr
save bdatos

*%%%%**FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN6)**%%%%*
%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema %de reconocimiento para conjunto cerrado (diferentes locutores, un solo micrófono, %mic2 , 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos %de los locutores llamando a vtrain.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de índices correspondientes a frecuencias críticas para

```
%las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%centr: base de datos de vectores centroide de cada declaración

centr=[];

%bdatos: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos=[];

%bdloc: base de datos que asocia a cada locutor un número

bdloc=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Número de centroides

N=10;

[centr(1:N,1:5),bdatos(1,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,ncoef2
,k2,6001,11000);
bdloc(1,1:21)='El locutor es Arancha';

[centr(1:N,6:10),bdatos(2,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,ncoef
2,k2,7201,12200);
```

```
bdloc(2,1:21)='El locutor es Arancha';

[centr(1:N,11:15),bdatos(3,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap,nco
ef2,k2,6401,11400);
bdloc(3,1:21)='El locutor es Arancha';

[centr(1:N,16:20),bdatos(4,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap,nco
ef2,k2,7001,12000);
bdloc(4,1:21)='El locutor es Arancha';

[centr(1:N,21:25),bdatos(5,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap,nco
ef2,k2,6201,11200);
bdloc(5,1:21)='El locutor es Arancha';

[centr(1:N,26:30),bdatos(6,1:49)]=vtrain('c:\matlab\elena211.wav',nmuest,nsolap,ncoe
f2,k2,9501,14500);
bdloc(6,1:19)='El locutor es Elena';

[centr(1:N,31:35),bdatos(7,1:49)]=vtrain('c:\matlab\elena212.wav',nmuest,nsolap,ncoe
f2,k2,6001,11000);
bdloc(7,1:19)='El locutor es Elena';

[centr(1:N,36:40),bdatos(8,1:49)]=vtrain('c:\matlab\elena213.wav',nmuest,nsolap,ncoe
f2,k2,5001,10000);
bdloc(8,1:19)='El locutor es Elena';

[centr(1:N,41:45),bdatos(9,1:49)]=vtrain('c:\matlab\elena214.wav',nmuest,nsolap,ncoe
f2,k2,4001,9000);
bdloc(9,1:19)='El locutor es Elena';
```

```
[centr(1:N,46:50),bdatos(10,1:49)]=vtrain('c:\matlab\elena215.wav',nmuest,nsolap,ncoef2,k2,2001,7000);
bdloc(10,1:19)='El locutor es Elena';

[centr(1:N,51:55),bdatos(11,1:49)]=vtrain('c:\matlab\mang211.wav',nmuest,nsolap,ncoef2,k2,11001,16000);
bdloc(11,1:23)='El locutor es M.Angeles';

[centr(1:N,56:60),bdatos(12,1:49)]=vtrain('c:\matlab\mang212.wav',nmuest,nsolap,ncoef2,k2,8501,13500);
bdloc(12,1:23)='El locutor es M.Angeles';

centr(1:N,61:65),bdatos(13,1:49)]=vtrain('c:\matlab\mang213.wav',nmuest,nsolap,ncoef2,k2,11001,16000);
bdloc(13,1:23)='El locutor es M.Angeles';

[centr(1:N,66:70),bdatos(14,1:49)]=vtrain('c:\matlab\mang214.wav',nmuest,nsolap,ncoef2,k2,8101,13100);
bdloc(14,1:23)='El locutor es M.Angeles';

[centr(1:N,71:75),bdatos(15,1:49)]=vtrain('c:\matlab\mang215.wav',nmuest,nsolap,ncoef2,k2,11901,16900);
bdloc(15,1:23)='El locutor es M.Angeles';

[centr(1:N,76:80),bdatos(16,1:49)]=vtrain('c:\matlab\maria211.wav',nmuest,nsolap,ncoef2,k2,6301,11300);
bdloc(16,1:19)='El locutor es Maria';

[centr(1:N,81:85),bdatos(17,1:49)]=vtrain('c:\matlab\maria212.wav',nmuest,nsolap,ncoef2,k2,6001,11000);
bdloc(17,1:19)='El locutor es Maria';
```

```
[centr(1:N,86:90),bdatos(18,1:49)]=vtrain('c:\matlab\maria213.wav',nmuest,nsolap,nc
oef2,k2,7501,12500);
bdloc(18,1:19)='El locutor es Maria';

[centr(1:N,91:95),bdatos(19,1:49)]=vtrain('c:\matlab\maria214.wav',nmuest,nsolap,nc
oef2,k2,6401,11400);
bdloc(19,1:19)='El locutor es Maria';

[centr(1:N,96:100),bdatos(20,1:49)]=vtrain('c:\matlab\maria215.wav',nmuest,nsolap,n
coef2,k2,6501,11500);
bdloc(20,1:19)='El locutor es Maria';

save bdloc
save centr
save bdatos
```

```
%%%%FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN7)%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto abierto (un único locutor, un solo micrófono, mic2 ,
%4.000 muestras de voz por declaración, 65 coeficientes FB) y la base de datos de los
%locutores llamando a vtrain.m y cumbral.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;
```

```
%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Número de coeficientes de Fourier-Bessel a partir
%de los cuales reconstruiremos la señal de voz

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%centr2: base de datos de vectores centroide de cada declaración

centr2=[];

%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados

bdatos2=[];

%bdloc2: base de datos que asocia a cada locutor un número

bdloc2=[];

% umbral: máxima distorsión permitida entre patrones de un mismo
%locutor de referencia.
```

```
umbral=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Número de centroides

N=10;

%Método de cálculo de la distorsión

calculo=1;

[centr2(1:N,1:15),bdatos2(1,1:39)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,nc
oef,k,6301,10300);
bdloc2(1,1:21)='El locutor es Arancha';

[centr2(1:N,16:30),bdatos2(2,1:39)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,n
coef,k,7501,11500);
bdloc2(2,1:21)='El locutor es Arancha';

[centr2(1:N,31:45),bdatos2(3,1:39)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap,n
coef,k,6801,10800);
bdloc2(3,1:21)='El locutor es Arancha';

[centr2(1:N,46:60),bdatos2(4,1:39)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap,n
coef,k,7601,11600);
bdloc2(4,1:21)='El locutor es Arancha';
```

```
[centr2(1:N,61:75),bdatos2(5,1:39)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap,n
coef,k,6301,10300);
bdloc2(5,1:21)='El locutor es Arancha';

save bdloc2
save centr2
save bdatos2

umbral=cumbral(centr2,bdatos2,calculo);
save umbral
```

```
%%%%FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN8)%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa ‘script’ de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto cerrado (todos los locutores de la base de datos,
%dos micrófonos, 5.000 muestras de voz por declaración, 195 coeficientes FB) y la
%base de datos de los locutores llamando a vtrain.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;
```

```
%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de Índices correspondientes a frecuencias críticas
%para las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%centr: Base de datos de vectores centroides de cada
%declaración.

centr=[];

%bdatos: Base de datos de correspondencia entre el alfabeto
%de vectores característicos y los vectores centroide calculados.

bdatos=[];


%bdloc: Base de datos que asocia a cada locutor un número.

bdloc=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Número de centroides
```

```
%N=10;
%N=15;
N=20;

[centr(1:N,1:15),bdatos(1,1:49)]=vtrain('c:\matlab\lina11.wav',nmuest,nsolap,ncoef,k,
5101,10100);

bdloc(1,1:18)='El locutor es Lina';

[centr(1:N,16:30),bdatos(2,1:49)]=vtrain('c:\matlab\lina12.wav',nmuest,nsolap,ncoef,k
,4501,9500);

bdloc(2,1:18)='El locutor es Lina';

[centr(1:N,31:45),bdatos(3,1:49)]=vtrain('c:\matlab\lina13.wav',nmuest,nsolap,ncoef,k
,7601,12600);

bdloc(3,1:18)='El locutor es Lina';

[centr(1:N,46:60),bdatos(4,1:49)]=vtrain('c:\matlab\lina14.wav',nmuest,nsolap,ncoef,k
,5001,10000);

bdloc(4,1:18)='El locutor es Lina';

[centr(1:N,61:75),bdatos(5,1:49)]=vtrain('c:\matlab\lina15.wav',nmuest,nsolap,ncoef,k
,6501,11500);

bdloc(5,1:18)='El locutor es Lina';

[centr(1:N,76:90),bdatos(6,1:49)]=vtrain('c:\matlab\jordi11.wav',nmuest,nsolap,ncoef,
k,2301,7300);
```

bdloc(6,1:19)='El locutor es Jordi';

[centr(1:N,91:105),bdatos(7,1:49)]=vtrain('c:\matlab\jordi12.wav',nmuest,nsolap,ncoef,k,3001,8000);

bdloc(7,1:19)='El locutor es Jordi';

[centr(1:N,106:120),bdatos(8,1:49)]=vtrain('c:\matlab\jordi13.wav',nmuest,nsolap,ncoef,k,3501,8500);

bdloc(8,1:19)='El locutor es Jordi';

[centr(1:N,121:135),bdatos(9,1:49)]=vtrain('c:\matlab\jordi14.wav',nmuest,nsolap,ncoef,k,7601,12600);

bdloc(9,1:19)='El locutor es Jordi';

[centr(1:N,136:150),bdatos(10,1:49)]=vtrain('c:\matlab\jordi15.wav',nmuest,nsolap,ncoef,k,3001,8000);

bdloc(10,1:19)='El locutor es Jordi';

[centr(1:N,151:165),bdatos(11,1:49)]=vtrain('c:\matlab\yo11.wav',nmuest,nsolap,ncoef,k,6251,11250);

bdloc(11,1:21)='El locutor es Arancha';

[centr(1:N,166:180),bdatos(12,1:49)]=vtrain('c:\matlab\yo12.wav',nmuest,nsolap,ncoef,k,7951,12950);

bdloc(12,1:21)='El locutor es Arancha';

[centr(1:N,181:195),bdatos(13,1:49)]=vtrain('c:\matlab\yo13.wav',nmuest,nsolap,ncoef,k,7751,12750);

bdloc(13,1:21)='El locutor es Arancha';

[centr(1:N,196:210),bdatos(14,1:49)]=vtrain('c:\matlab\yo14.wav',nmuest,nsolap,ncoef,k,7501,12500);

bdloc(14,1:21)='El locutor es Arancha';

[centr(1:N,211:225),bdatos(15,1:49)]=vtrain('c:\matlab\yo15.wav',nmuest,nsolap,ncoef,k,4501,9500);

bdloc(15,1:21)='El locutor es Arancha';

[centr(1:N,256:270),bdatos(16,1:49)]=vtrain('c:\matlab\juanjo11.wav',nmuest,nsolap,ncoef,k,2201,7200);

bdloc(16,1:20)='El locutor es Juanjo';

[centr(1:N,271:285),bdatos(17,1:49)]=vtrain('c:\matlab\juanjo12.wav',nmuest,nsolap,ncoef,k,11501,16500);

bdloc(17,1:20)='El locutor es Juanjo';

[centr(1:N,286:300),bdatos(18,1:49)]=vtrain('c:\matlab\juanjo13.wav',nmuest,nsolap,ncoef,k,10001,15000);

bdloc(18,1:20)='El locutor es Juanjo';

```
[centr(1:N,301:315),bdatos(19,1:49)]=vtrain('c:\matlab\juanjo14.wav',nmuest,nsolap,ncoef,k,11751,16750);

bdloc(19,1:20)='El locutor es Juanjo';

[centr(1:N,316:330),bdatos(20,1:49)]=vtrain('c:\matlab\juanjo15.wav',nmuest,nsolap,ncoef,k,9901,14900);

bdloc(20,1:20)='El locutor es Juanjo';

[centr(1:N,376:390),bdatos(21,1:49)]=vtrain('c:\matlab\angel11.wav',nmuest,nsolap,ncoef,k,7001,12000);

bdloc(21,1:19)='El locutor es Angel';

[centr(1:N,391:405),bdatos(22,1:49)]=vtrain('c:\matlab\angel12.wav',nmuest,nsolap,ncoef,k,6501,11500);

bdloc(22,1:19)='El locutor es Angel';

[centr(1:N,406:420),bdatos(23,1:49)]=vtrain('c:\matlab\angel13.wav',nmuest,nsolap,ncoef,k,3801,8800);

bdloc(23,1:19)='El locutor es Angel';

[centr(1:N,421:435),bdatos(24,1:49)]=vtrain('c:\matlab\angel14.wav',nmuest,nsolap,ncoef,k,4101,9100);

bdloc(24,1:19)='El locutor es Angel';
```

[centr(1:N,436:450),bdatos(25,1:49)]=vtrain('c:\matlab\angel15.wav',nmuest,nsolap,ncoef,k,3901,8900);

bdloc(25,1:19)='El locutor es Angel';

[centr(1:N,451:465),bdatos(26,1:49)]=vtrain('c:\matlab\carmen11.wav',nmuest,nsolap,ncoef,k,4501,9500);

bdloc(26,1:20)='El locutor es Carmen';

[centr(1:N,466:480),bdatos(27,1:49)]=vtrain('c:\matlab\carmen12.wav',nmuest,nsolap,ncoef,k,6301,11300);

bdloc(27,1:20)='El locutor es Carmen';

[centr(1:N,481:495),bdatos(28,1:49)]=vtrain('c:\matlab\carmen13.wav',nmuest,nsolap,ncoef,k,6901,11900);

bdloc(28,1:20)='El locutor es Carmen';

[centr(1:N,496:510),bdatos(29,1:49)]=vtrain('c:\matlab\carmen14.wav',nmuest,nsolap,ncoef,k,6301,11300);

bdloc(29,1:20)='El locutor es Carmen';

[centr(1:N,511:525),bdatos(30,1:49)]=vtrain('c:\matlab\carmen15.wav',nmuest,nsolap,ncoef,k,3001,8000);

bdloc(30,1:20)='El locutor es Carmen';

[centr(1:N,526:540),bdatos(31,1:49)]=vtrain('c:\matlab\monica11.wav',nmuest,nsolap, ncoef,k,5501,10500);

bdloc(31,1:20)='El locutor es Monica';

[centr(1:N,541:555),bdatos(32,1:49)]=vtrain('c:\matlab\monica12.wav',nmuest,nsolap, ncoef,k,5101,10100);

bdloc(32,1:20)='El locutor es Monica';

[centr(1:N,556:570),bdatos(33,1:49)]=vtrain('c:\matlab\monica13.wav',nmuest,nsolap, ncoef,k,3001,8000);

bdloc(33,1:20)='El locutor es Monica';

[centr(1:N,571:585),bdatos(34,1:49)]=vtrain('c:\matlab\monica14.wav',nmuest,nsolap, ncoef,k,2701,7700);

bdloc(34,1:20)='El locutor es Monica';

[centr(1:N,586:600),bdatos(35,1:49)]=vtrain('c:\matlab\monica15.wav',nmuest,nsolap, ncoef,k,3301,8300);

bdloc(35,1:20)='El locutor es Monica';

[centr(1:N,601:615),bdatos(36,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap, ncoef,k,6001,11000);

bdloc(36,1:21)='El locutor es Arancha';

[centr(1:N,616:630),bdatos(37,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap, ncoef,k,7201,12200);

bdloc(37,1:21)='El locutor es Arancha';

[centr(1:N,631:645),bdatos(38,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap, ncoef,k,6401,11400);

bdloc(38,1:21)='El locutor es Arancha';

[centr(1:N,646:660),bdatos(39,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap, ncoef,k,7001,12000);

bdloc(39,1:21)='El locutor es Arancha';

[centr(1:N,661:675),bdatos(40,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap, ncoef,k,6201,11200);

bdloc(40,1:21)='El locutor es Arancha';

[centr(1:N,676:690),bdatos(41,1:49)]=vtrain('c:\matlab\elena211.wav',nmuest,nsolap, ncoef,k,9501,14500);

bdloc(41,1:19)='El locutor es Elena';

[centr(1:N,691:705),bdatos(42,1:49)]=vtrain('c:\matlab\elena212.wav',nmuest,nsolap, ncoef,k,6001,11000);

bdloc(42,1:19)='El locutor es Elena';

[centr(1:N,706:720),bdatos(43,1:49)]=vtrain('c:\matlab\elena213.wav',nmuest,nsolap, ncoef,k,5001,10000);

bdloc(43,1:19)='El locutor es Elena';

[centr(1:N,721:735),bdatos(44,1:49)]=vtrain('c:\matlab\elena214.wav',nmuest,nsolap, ncoef,k,4001,9000);

bdloc(44,1:19)='El locutor es Elena';

[centr(1:N,736:750),bdatos(45,1:49)]=vtrain('c:\matlab\elena215.wav',nmuest,nsolap, ncoef,k,2001,7000);

bdloc(45,1:19)='El locutor es Elena';

[centr(1:N,751:765),bdatos(46,1:49)]=vtrain('c:\matlab\mang211.wav',nmuest,nsolap, ncoef,k,11001,16000);

bdloc(46,1:23)='El locutor es M.Angeles';

[centr(1:N,766:780),bdatos(47,1:49)]=vtrain('c:\matlab\mang212.wav',nmuest,nsolap, ncoef,k,8501,13500);

bdloc(47,1:23)='El locutor es M.Angeles';

[centr(1:N,781:795),bdatos(48,1:49)]=vtrain('c:\matlab\mang213.wav',nmuest,nsolap, ncoef,k,11001,16000);

bdloc(48,1:23)='El locutor es M.Angeles';

[centr(1:N,796:810),bdatos(49,1:49)]=vtrain('c:\matlab\mang214.wav',nmuest,nsolap, ncoef,k,8101,13100);

bdloc(49,1:23)='El locutor es M.Angeles';

[centr(1:N,811:825),bdatos(50,1:49)]=vtrain('c:\matlab\mang215.wav',nmuest,nsolap, ncoef,k,11901,16900);

bdloc(50,1:23)='El locutor es M.Angeles';

[centr(1:N,826:840),bdatos(51,1:49)]=vtrain('c:\matlab\maria211.wav',nmuest,nsolap, ncoef,k,6301,11300);

bdloc(51,1:19)='El locutor es Maria';

[centr(1:N,841:855),bdatos(52,1:49)]=vtrain('c:\matlab\maria212.wav',nmuest,nsolap, ncoef,k,6001,11000);

bdloc(52,1:19)='El locutor es Maria';

[centr(1:N,856:870),bdatos(53,1:49)]=vtrain('c:\matlab\maria213.wav',nmuest,nsolap, ncoef,k,7501,12500);

bdloc(53,1:19)='El locutor es Maria';

[centr(1:N,871:885),bdatos(54,1:49)]=vtrain('c:\matlab\maria214.wav',nmuest,nsolap, ncoef,k,6401,11400);

bdloc(54,1:19)='El locutor es Maria';

[centr(1:N,886:900),bdatos(55,1:49)]=vtrain('c:\matlab\maria215.wav',nmuest,nsolap, ncoef,k,6501,11500);

bdloc(55,1:19)='El locutor es Maria';

save bdloc
save centr
save bdatos

*%%%%**FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAIN9)**%%%%*
%%
% Programa 'script' de MATLAB que genera el proceso de entrenamiento del sistema %de reconocimiento para conjunto cerrado (locutores grabados con micrófono 2, %5.000 muestras de voz por declaración, 195 coeficientes FB) y la base de datos de los %locutores llamando a vtrain.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos

```
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de Índices correspondientes a frecuencias críticas
%para las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%centr: Base de datos de vectores centroides de cada
%declaración.

centr=[];

%bdatos: Base de datos de correspondencia entre el alfabeto
%de vectores característicos y los vectores centroide calculados.

bdatos=[];


%bdloc: Base de datos que asocia a cada locutor un número.

bdloc=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Número de centroides

%N=10;
```

```
%N=15;
N=20;

[centr(1:N,1:15),bdatos(1,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,ncoef,k,6001,11000);

bdloc(1,1:21)='El locutor es Arancha';

[centr(1:N,16:30),bdatos(2,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,ncoef,k,7201,12200);

bdloc(2,1:21)='El locutor es Arancha';

[centr(1:N,31:45),bdatos(3,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap,ncoef,k,6401,11400);

bdloc(3,1:21)='El locutor es Arancha';

[centr(1:N,46:60),bdatos(4,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap,ncoef,k,7001,12000);

bdloc(4,1:21)='El locutor es Arancha';

[centr(1:N,61:75),bdatos(5,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap,ncoef,k,6201,11200);

bdloc(5,1:21)='El locutor es Arancha';

[centr(1:N,76:90),bdatos(6,1:49)]=vtrain('c:\matlab\elena211.wav',nmuest,nsolap,ncoef,k,9501,14500);
```

```
bdloc(6,1:19)='El locutor es Elena';

[centr(1:N,91:105),bdatos(7,1:49)]=vtrain('c:\matlab\elena212.wav',nmuest,nsolap,ncoef,k,6001,11000);

bdloc(7,1:19)='El locutor es Elena';

[centr(1:N,106:120),bdatos(8,1:49)]=vtrain('c:\matlab\elena213.wav',nmuest,nsolap,ncoef,k,5001,10000);

bdloc(8,1:19)='El locutor es Elena';

[centr(1:N,121:135),bdatos(9,1:49)]=vtrain('c:\matlab\elena214.wav',nmuest,nsolap,ncoef,k,4001,9000);

bdloc(9,1:19)='El locutor es Elena';

[centr(1:N,136:150),bdatos(10,1:49)]=vtrain('c:\matlab\elena215.wav',nmuest,nsolap,
ncoef,k,2001,7000);

bdloc(10,1:19)='El locutor es Elena';

[centr(1:N,151:165),bdatos(11,1:49)]=vtrain('c:\matlab\mang211.wav',nmuest,nsolap,
ncoef,k,11001,16000);

bdloc(11,1:23)='El locutor es M.Angeles';

[centr(1:N,166:180),bdatos(12,1:49)]=vtrain('c:\matlab\mang212.wav',nmuest,nsolap,
ncoef,k,8501,13500);

bdloc(12,1:23)='El locutor es M.Angeles';
```

[centr(1:N,181:195),bdatos(13,1:49)]=vtrain('c:\matlab\mang213.wav',nmuest,nsolap, ncoef,k,11001,16000);

bdloc(13,1:23)='El locutor es M.Angeles';

[centr(1:N,196:210),bdatos(14,1:49)]=vtrain('c:\matlab\mang214.wav',nmuest,nsolap, ncoef,k,8101,13100);

bdloc(14,1:23)='El locutor es M.Angeles';

[centr(1:N,211:225),bdatos(15,1:49)]=vtrain('c:\matlab\mang215.wav',nmuest,nsolap, ncoef,k,11901,16900);

bdloc(15,1:23)='El locutor es M.Angeles';

[centr(1:N,226:240),bdatos(16,1:49)]=vtrain('c:\matlab\maria211.wav',nmuest,nsolap, ncoef,k,6301,11300);

bdloc(16,1:19)='El locutor es Maria';

[centr(1:N,241:255),bdatos(17,1:49)]=vtrain('c:\matlab\maria212.wav',nmuest,nsolap, ncoef,k,6001,11000);

bdloc(17,1:19)='El locutor es Maria';

[centr(1:N,256:270),bdatos(18,1:49)]=vtrain('c:\matlab\maria213.wav',nmuest,nsolap, ncoef,k,7501,12500);

bdloc(18,1:19)='El locutor es Maria';

```
[centr(1:N,271:285),bdatos(19,1:49)]=vtrain('c:\matlab\maria214.wav',nmuest,nsolap,
ncoef,k,6401,11400);

bdloc(19,1:19)='El locutor es Maria';

[centr(1:N,286:300),bdatos(20,1:49)]=vtrain('c:\matlab\maria215.wav',nmuest,nsolap,
ncoef,k,6501,11500);

bdloc(20,1:19)='El locutor es Maria';

save bdloc
save centr
save bdatos




%%%%FASE DE ENTRENAMIENTO DEL IDENTIFICADOR (TRAINa)%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa ‘script’ de MATLAB que genera el proceso de entrenamiento del sistema
%de reconocimiento para conjunto cerrado (todos los locutores de la base de datos,
%dos micrófonos, 5.000 muestras de voz por declaración, 65 coeficientes FB) y la base
%de datos de los locutores llamando a vtrain.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
```

```
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de Índices correspondientes a frecuencias críticas
%para las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%centr2: Base de datos de vectores centroides de cada
%declaración.

centr=[];

%bdatos2: Base de datos de correspondencia entre el alfabeto
%de vectores característicos y los vectores centroide calculados.

bdatos=[];

%bdloc2: Base de datos que asocia a cada patrón del locutor
%un número

bdloc=[];

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
```

```
%a los archivos de voz como parámetro

%Número de centroides

%N=10;
%N=15;
N=20;

[centr(1:N,1:5),bdatos(1,1:49)]=vtrain('c:\matlab\lina11.wav',nmuest,nsolap,ncoef2,k2
,5101,10100);

bdloc(1,1:18)='El locutor es Lina';

[centr(1:N,6:10),bdatos(2,1:49)]=vtrain('c:\matlab\lina12.wav',nmuest,nsolap,ncoef2,k
2,4501,9500);

bdloc(2,1:18)='El locutor es Lina';

[centr(1:N,11:15),bdatos(3,1:49)]=vtrain('c:\matlab\lina13.wav',nmuest,nsolap,ncoef2,
k2,7601,12600);

bdloc(3,1:18)='El locutor es Lina';

[centr(1:N,16:20),bdatos(4,1:49)]=vtrain('c:\matlab\lina14.wav',nmuest,nsolap,ncoef2,
k2,5001,10000);

bdloc(4,1:18)='El locutor es Lina';

[centr(1:N,21:25),bdatos(5,1:49)]=vtrain('c:\matlab\lina15.wav',nmuest,nsolap,ncoef2,
k2,6501,11500);
```

```
bdloc(5,1:18)='El locutor es Lina';

[centr(1:N,26:30),bdatos(6,1:49)]=vtrain('c:\matlab\jordi11.wav',nmuest,nsolap,ncoef2
,k2,2301,7300);

bdloc(6,1:19)='El locutor es Jordi';

[centr(1:N,31:35),bdatos(7,1:49)]=vtrain('c:\matlab\jordi12.wav',nmuest,nsolap,ncoef2
,k2,3001,8000);

bdloc(7,1:19)='El locutor es Jordi';

[centr(1:N,36:40),bdatos(8,1:49)]=vtrain('c:\matlab\jordi13.wav',nmuest,nsolap,ncoef2
,k2,3501,8500);

bdloc(8,1:19)='El locutor es Jordi';

[centr(1:N,41:45),bdatos(9,1:49)]=vtrain('c:\matlab\jordi14.wav',nmuest,nsolap,ncoef2
,k2,7601,12600);

bdloc(9,1:19)='El locutor es Jordi';

[centr(1:N,46:50),bdatos(10,1:49)]=vtrain('c:\matlab\jordi15.wav',nmuest,nsolap,ncoef
2,k2,3001,8000);

bdloc(10,1:19)='El locutor es Jordi';

[centr(1:N,51:55),bdatos(11,1:49)]=vtrain('c:\matlab\yo11.wav',nmuest,nsolap,ncoef2,
k2,6251,11250);

bdloc(11,1:21)='El locutor es Arancha';
```

```
[centr(1:N,56:60),bdatos(12,1:49)]=vtrain('c:\matlab\yo12.wav',nmuest,nsolap,ncoef2,
k2,7951,12950);

bdloc(12,1:21)='El locutor es Arancha';

[centr(1:N,61:65),bdatos(13,1:49)]=vtrain('c:\matlab\yo13.wav',nmuest,nsolap,ncoef2,
k2,7751,12750);

bdloc(13,1:21)='El locutor es Arancha';

[centr(1:N,66:70),bdatos(14,1:49)]=vtrain('c:\matlab\yo14.wav',nmuest,nsolap,ncoef2,
k2,7501,12500);

bdloc(14,1:21)='El locutor es Arancha';

[centr(1:N,71:75),bdatos(15,1:49)]=vtrain('c:\matlab\yo15.wav',nmuest,nsolap,ncoef2,
k2,4501,9500);

bdloc(15,1:21)='El locutor es Arancha';

[centr(1:N,86:90),bdatos(16,1:49)]=vtrain('c:\matlab\juanjo11.wav',nmuest,nsolap,nco
ef2,k2,2201,7200);

bdloc(16,1:20)='El locutor es Juanjo';

[centr(1:N,91:95),bdatos(17,1:49)]=vtrain('c:\matlab\juanjo12.wav',nmuest,nsolap,nco
ef2,k2,11501,16500);

bdloc(17,1:20)='El locutor es Juanjo';
```

[centr(1:N,96:100),bdatos(18,1:49)]=vtrain('c:\matlab\juanjo13.wav',nmuest,nsolap,ncoef2,k2,10001,15000);

bdloc(18,1:20)='El locutor es Juanjo';

[centr(1:N,101:105),bdatos(19,1:49)]=vtrain('c:\matlab\juanjo14.wav',nmuest,nsolap,ncoef2,k2,11751,16750);

bdloc(19,1:20)='El locutor es Juanjo';

[centr(1:N,106:110),bdatos(20,1:49)]=vtrain('c:\matlab\juanjo15.wav',nmuest,nsolap,ncoef2,k2,9901,14900);

bdloc(20,1:20)='El locutor es Juanjo';

[centr(1:N,126:130),bdatos(21,1:49)]=vtrain('c:\matlab\angel11.wav',nmuest,nsolap,ncoef2,k2,7001,12000);

bdloc(21,1:19)='El locutor es Angel';

[centr(1:N,131:135),bdatos(22,1:49)]=vtrain('c:\matlab\angel12.wav',nmuest,nsolap,ncoef2,k2,6501,11500);

bdloc(22,1:19)='El locutor es Angel';

[centr(1:N,136:140),bdatos(23,1:49)]=vtrain('c:\matlab\angel13.wav',nmuest,nsolap,ncoef2,k2,3801,8800);

bdloc(23,1:19)='El locutor es Angel';

```
[centr(1:N,141:145),bdatos(24,1:49)]=vtrain('c:\matlab\angel14.wav',nmuest,nsolap,n
coef2,k2,4101,9100);

bdloc(24,1:19)='El locutor es Angel';

[centr(1:N,146:150),bdatos(25,1:49)]=vtrain('c:\matlab\angel15.wav',nmuest,nsolap,n
coef2,k2,3901,8900);

bdloc(25,1:19)='El locutor es Angel';

[centr(1:N,151:155),bdatos(26,1:49)]=vtrain('c:\matlab\carmen11.wav',nmuest,nsolap,
ncoef2,k2,4501,9500);

bdloc(26,1:20)='El locutor es Carmen';

[centr(1:N,156:160),bdatos(27,1:49)]=vtrain('c:\matlab\carmen12.wav',nmuest,nsolap,
ncoef2,k2,6301,11300);

bdloc(27,1:20)='El locutor es Carmen';

[centr(1:N,161:165),bdatos(28,1:49)]=vtrain('c:\matlab\carmen13.wav',nmuest,nsolap,
ncoef2,k2,6901,11900);

bdloc(28,1:20)='El locutor es Carmen';

[centr(1:N,166:170),bdatos(29,1:49)]=vtrain('c:\matlab\carmen14.wav',nmuest,nsolap,
ncoef2,k2,6301,11300);

bdloc(29,1:20)='El locutor es Carmen';
```

[centr(1:N,171:175),bdatos(30,1:49)]=vtrain('c:\matlab\carmen15.wav',nmuest,nsolap, ncoef2,k2,3001,8000);

bdloc(30,1:20)='El locutor es Carmen';

[centr(1:N,176:180),bdatos(31,1:49)]=vtrain('c:\matlab\monica11.wav',nmuest,nsolap, ncoef2,k2,5501,10500);

bdloc(31,1:20)='El locutor es Monica';

[centr(1:N,181:185),bdatos(32,1:49)]=vtrain('c:\matlab\monica12.wav',nmuest,nsolap, ncoef2,k2,5101,10100);

bdloc(32,1:20)='El locutor es Monica';

[centr(1:N,186:190),bdatos(33,1:49)]=vtrain('c:\matlab\monica13.wav',nmuest,nsolap, ncoef2,k2,3001,8000);

bdloc(33,1:20)='El locutor es Monica';

[centr(1:N,191:195),bdatos(34,1:49)]=vtrain('c:\matlab\monica14.wav',nmuest,nsolap, ncoef2,k2,2701,7700);

bdloc(34,1:20)='El locutor es Monica';

[centr(1:N,196:200),bdatos(35,1:49)]=vtrain('c:\matlab\monica15.wav',nmuest,nsolap, ncoef2,k2,3301,8300);

bdloc(35,1:20)='El locutor es Monica';

```
[centr(1:N,201:205),bdatos(36,1:49)]=vtrain('c:\matlab\yomic211.wav',nmuest,nsolap,
ncoef2,k2,6001,11000);

bdloc(36,1:21)='El locutor es Arancha';

[centr(1:N,206:210),bdatos(37,1:49)]=vtrain('c:\matlab\yomic212.wav',nmuest,nsolap,
ncoef2,k2,7201,12200);

bdloc(37,1:21)='El locutor es Arancha';

[centr(1:N,211:215),bdatos(38,1:49)]=vtrain('c:\matlab\yomic213.wav',nmuest,nsolap,
ncoef2,k2,6401,11400);

bdloc(38,1:21)='El locutor es Arancha';

[centr(1:N,216:220),bdatos(39,1:49)]=vtrain('c:\matlab\yomic214.wav',nmuest,nsolap,
ncoef2,k2,7001,12000);

bdloc(39,1:21)='El locutor es Arancha';

[centr(1:N,221:225),bdatos(40,1:49)]=vtrain('c:\matlab\yomic215.wav',nmuest,nsolap,
ncoef2,k2,6201,11200);

bdloc(40,1:21)='El locutor es Arancha';

[centr(1:N,226:230),bdatos(41,1:49)]=vtrain('c:\matlab\elena211.wav',nmuest,nsolap,
ncoef2,k2,9501,14500);

bdloc(41,1:19)='El locutor es Elena';
```

[centr(1:N,231:235),bdatos(42,1:49)]=vtrain('c:\matlab\elena212.wav',nmuest,nsolap, ncoef2,k2,6001,11000);

bdloc(42,1:19)='El locutor es Elena';

[centr(1:N,236:240),bdatos(43,1:49)]=vtrain('c:\matlab\elena213.wav',nmuest,nsolap, ncoef2,k2,5001,10000);

bdloc(43,1:19)='El locutor es Elena';

[centr(1:N,241:245),bdatos(44,1:49)]=vtrain('c:\matlab\elena214.wav',nmuest,nsolap, ncoef2,k2,4001,9000);

bdloc(44,1:19)='El locutor es Elena';

[centr(1:N,246:250),bdatos(45,1:49)]=vtrain('c:\matlab\elena215.wav',nmuest,nsolap, ncoef2,k2,2001,7000);

bdloc(45,1:19)='El locutor es Elena';

[centr(1:N,251:255),bdatos(46,1:49)]=vtrain('c:\matlab\mang211.wav',nmuest,nsolap, ncoef2,k2,11001,16000);

bdloc(46,1:23)='El locutor es M.Angeles';

[centr(1:N,256:260),bdatos(47,1:49)]=vtrain('c:\matlab\mang212.wav',nmuest,nsolap, ncoef2,k2,8501,13500);

bdloc(47,1:23)='El locutor es M.Angeles';

[centr(1:N,261:265),bdatos(48,1:49)]=vtrain('c:\matlab\mang213.wav',nmuest,nsolap, ncoef2,k2,11001,16000);

bdloc(48,1:23)='El locutor es M.Angeles';

[centr(1:N,266:270),bdatos(49,1:49)]=vtrain('c:\matlab\mang214.wav',nmuest,nsolap, ncoef2,k2,8101,13100);

bdloc(49,1:23)='El locutor es M.Angeles';

[centr(1:N,271:275),bdatos(50,1:49)]=vtrain('c:\matlab\mang215.wav',nmuest,nsolap, ncoef2,k2,11901,16900);

bdloc(50,1:23)='El locutor es M.Angeles';

[centr(1:N,276:280),bdatos(51,1:49)]=vtrain('c:\matlab\maria211.wav',nmuest,nsolap, ncoef2,k2,6301,11300);

bdloc(51,1:19)='El locutor es Maria';

[centr(1:N,281:285),bdatos(52,1:49)]=vtrain('c:\matlab\maria212.wav',nmuest,nsolap, ncoef2,k2,6001,11000);

bdloc(52,1:19)='El locutor es Maria';

[centr(1:N,286:290),bdatos(53,1:49)]=vtrain('c:\matlab\maria213.wav',nmuest,nsolap, ncoef2,k2,7501,12500);

bdloc(53,1:19)='El locutor es Maria';

[centr(1:N,291:295),bdatos(54,1:49)]=vtrain('c:\matlab\maria214.wav',nmuest,nsolap, ncoef2,k2,6401,11400);

bdloc(54,1:19)='El locutor es Maria';

[centr(1:N,296:300),bdatos(55,1:49)]=vtrain('c:\matlab\maria215.wav',nmuest,nsolap, ncoef2,k2,6501,11500);

bdloc(55,1:19)='El locutor es Maria';

save bdloc
save centr
save bdatos

7.1.2: Extracción de la Característica: Cálculo de los Coeficientes de Fourier-Bessel

function [centr,bdatos]=vtrain(ruta,nmuest,nsolap,ncoef,k,a,b)

% Función principal de la fase de entrenamiento en el reconocimiento de conjunto %cerrado que realiza llamadas a leedatos.m, para convertir las señales de voz a %secuencias de muestras, fb.m, para el cálculo de los coeficientes FB, energia.m, para %el cálculo de los vectores característicos de energía y cvect.m, para la cuantización %vectorial de estos vectores característicos. Devuelve los vectores característicos y sus %centroides para cada declaración para su posterior almacenamiento en la base de %datos del sistema.

```
if (ruta(1)~='/0')
    [voz,fs]=leedatos(ruta);
    [cmvoz,vozrec]=fb(voz(a:b),nmuest,nsolap,ncoef,k);
    clear voz
    envoz=energia(cmvoz,k);
    clear cmvoz
end

%Primera llamada a centroide para todos los vectores característicos de
%la declaración

[ncoef,nvect]=size(envoz');
metodo=1;

%Comprimimos la información característica a 10 vectores característicos
%o centroides
%10 CENTROIDES%
```

```
N=10;

%15 CENTROIDES%
%N=15;
%20 CENTROIDES%
%N=20;

[A,S,D]=cvect(N,envoz',metodo);

[fil,col]=size(A');
[totcol]=length(S');

centr(1:fil,1:col)=A';
bdatos(1,1:totcol)=S';
```

function [bdatos,fs]=leedatos(ruta)

```
%Función que toma como parámetro de entrada la ruta de acceso de una señal de voz,
%y la carga en la base de datos que tendremos almacenada para hacer la comparación
%entre declaraciones de locutores. Realizamos la carga del archivo de voz en el
%programa renombrándolo como una señal con un número de muestras determinado
%obtenidas a una frecuencia de muestreo fs.

[bdatos,fs]=wavread(ruta);
```

function [cm,vozrec]=fb(x,nmuest,nsolap,ncoef,indices)

% Función que calcula los coeficientes de Fourier-Bessel de un segmento de voz, dado %el propio segmento, el número de coeficientes a calcular, la longitud de las tramas de %segmentación y el número de muestras de solapamiento entre tramas adyacentes. %Para ello realiza una llamada a cerosb.m.

%cm: matriz de coeficientes de Fourier-Bessel. Su longitud
%vendrá determinada por el número de coeficientes que queramos
%calcular, ncoef.
%x: señal de voz de entrada.
%nmuest: número de muestras que tendrá cada una de las tramas
%en las que segmentemos la señal de voz.
%nsolap: número de muestras de solapamiento entre trama y trama.

%CÁLCULO DE COEFICIENTES DE FOURIER-BESSEL
%Determinamos la longitud de la secuencia de muestras que compone la
%señal de voz.

```
longx=length(x);

%Calculamos el número de tramas en que segmentaremos la señal de voz

ntramas=ceil(longx/nsolap)-1;

%Rellenamos con ceros el vector de voz hasta tener un número entero de
%tramas

k=(longx+1):(ceil(longx/nsolap)*nsolap);
x(k)=zeros(size(k));

%Segmentamos la señal de voz en trozos de nmuest muestras, solapadas
```

```
%nsolap muestras

x=x'; %<- Esto es nuevo

%Agrupamos la señal de voz en tramas, asignando una trama a cada fila
%de la matriz trama.

trama(1,:)=x(1:nmuest);

for i=2:ntramas
    trama(i,:)=x(1+(nmuest-nsolap)*(i-1):nmuest+(nmuest-nsolap)*(i-1));
end             %Fin de 'for'

%Calculamos la ventana Hamming de nmuest puntos

n=0:(nmuest-1);
w=0.54-(0.46*cos(2*pi*n/(nmuest-1)));

%Multiplico cada trama de voz por la ventana de Hamming de nmuest
%puntos
%w=vector fila

mauxiliar=ones(ntramas,1)*w;
vector=trama.*mauxiliar;

%Calculamos ncoef coeficientes de Fourier-Bessel según la regla de
%Simpson para cada trama de voz enventanada vector
%Extremos de cada intervalo de voz o trama
%Calculamos los ceros de la función de Bessel de primera especie

ceros=ceros(1000);
```

```
%ceros: vector de índices que marcan los ceros de la función de Bessel de
%primera especie para las 1000 primeras muestras
%m: vector de ceros 1 x ncoef

m=zeros(1,1:ncoef);
in=length(indices);

for n=1:in      %Para cada índice del vector pasado como parámetro
    j=indices(n);
            %Asigno al vector m 5 índices consecutivos a partir del dado por el
%vector de índices
    m(j:j+5)=j:j+5;
end

dis=find(m~=0);
for aux=1:length(dis)
    J(dis(aux),:)=besselj(1,ceros(dis(aux))*(1:nmuest)/nmuest);
    J(dis(aux),:)=J(dis(aux),:)/norm(J(dis(aux),:))^2;
end

%J: matriz de filas que son las funciones de Bessel

for n=1:ntramas
    c(n,:)=(J*(vector(n,:))')';
end
cm=c;
```

```
function ceros=ceros(nmuest)
%Función que calcula los ceros de la función de Bessel de %primera especie para un
%número de muestras nmuest.

%Creamos un vector de longitud nmuest

n=1:1:nmuest;

%Generamos la función de Bessel de primera especie de longitud nmuest.

x=besselj(1,n);
signo=sign(x);

for i=2:nmuest
%Para cada muestra buscamos los cambios de signo de la
%función.
%Si existe cambio, entonces vector(i-1)= 0
%Si no existe cambio, entonces vector(i-1)≠ 0

    vector(i-1)=signo(i)-signo(i-1);
end                         %Fin de 'for'

%Almacenamos los puntos en los que se produce un cambio de
%signo en la función de Bessel.

ceros=[1,find(vector~=0)];
```

function e=energia(coefm,indice)

%Función que determina el vector característico de energía para unos coeficientes %asociados a unas determinadas frecuencias críticas para la identificación del locutor, %que se pasan como parámetro por el vector de sus índices.

%Calculamos el tamaño de la matriz pasada como parámetro
%ntram: número de tramas
%orden: número de coeficientes de FB calculados para cada trama

[trama,orden]=size(coefm);

%Calculamos el valor absoluto de los coeficientes característicos

mcoefm=abs(coefm);

%Para cada trama

for n=1:trama

%Para cada índice del vector pasado como parámetro

for k=1:length(indice)

%Agrupamos en un vector el coeficiente de índice k y los 5 coeficientes de
%índices consecutivos a partir de éste.

vector=[mcoefm(n,indice(k)),mcoefm(n,indice(k)+1),...
mcoefm(n,indice(k)+2),mcoefm(n,indice(k)+3),...
mcoefm(n,indice(k)+4),mcoefm(n,indice(k)+5)];

%Realizamos la suma de los elementos del vector obtenido

```
suma=cumsum(vector);

%Nos quedamos con el último elemento: suma acumulada de todos los
%coeficientes señalados en valor absoluto.

en(n,k) =suma(length(vector));
    end     %Fin de 'for'
end         %Fin de 'for'

%Obtenemos el vector de energías: cada elemento es la suma acumulada de la energía
%de 5 coeficientes de FB consecutivos. Por tanto, cada elemento es la energía
%acumulada en un determinado rango de frecuencias alrededor de una frecuencia
%crítica dada.
%En cada fila tendremos el vector característico de energía para cada trama de la
%declaración.

e=en;
```

7.1.3: Cuantización Vectorial

```
function [A,S,D]=cvect(N,x,sel);
%Función principal del cuantizador vectorial. Recibe las muestras de x y el número de
%centroides deseado y devuelve estos

%CUANTIZADOR VECTORIAL.

%DEFINICIÓN DE VARIABLES:

CIERTO=1;
FALSO=0;
GRANDE=Inf;
D=[];

%err = Umbral relativo de decisión.

err=0.01;

%d = Tamaño de la secuencia de aprendizaje,
%k = Dimensión de los vectores.

[k,d]=size(x);

%Elección del alfabeto inicial.
%Como hay varios m,todos distintos, se implementan en una
%función aparte, para ganar en claridad ahora.

A=f1(x,N,sel);
```

```
%BUCLE:

m=2;
D(1)=GRANDE;
iteramos=CIERTO;
while(iteramos)

%Encontrar la partición de distorsión mínima para el alfabeto
%actual.

[S,D(m)]=f2(x,A);

%¿Es la mejora relativa mejor que el umbral? Si es así, paramos:

if D(m)==0|((D(m-1)-D(m))/D(m)<=err)|(m>50)
iteramos=FALSO;
else
A=f3(x,S,N);    %Recalculamos el alfabeto.
end     %fin de "if"

m=m+1;

end     %fin de "while"
```

```
function A=f1(X,N,i);
%Función que da condiciones iniciales al algoritmo para la primera iteración. Escoge
%un alfabeto inicial de 'N' elementos dados los vectores de entrada 'X' y devuelve su
%salida en 'A'.

%i = Indica el método a usar
%i = 1 <-- Se escogen de forma aleatoria.
%i = 2 <-- Se distribuyen alrededor del vector medio.
%CUIDADO: 'X' es una matriz cuyas COLUMNAS son los vectores.
%Lo mismo podemos decir de 'A'.

[k,d]=size(X);

%Versión 1: Escogemos los N primeros elementos de x.
if (i==1)
A=X(:,1:N);
end

%Versión 2: Los tomamos distribuidos alrededor de la media.
if (i==2)
A=zeros(k,N);
vm=mean(X')';
A(:,1)=vm;
desvest=std(X')';
j=2;
while j<=N
A(:,j)=vm+desvest.*randn(k,1);
j=j+1;
end
end
```

function [S,D]=f2(X,A);

%Función que define las regiones del cuantizador conocidos sus centroides.
%Encuentra la partición de distorsión mínima para el alfabeto actual.

%ENTRADAS:
%A = Matriz cuyas columnas son el alfabeto.
%X = Matriz cuyas columnas son los vectores de aprendizaje.
%SALIDAS:
%S = Vector colunma. Su elemento j indica a qué vector del alfabeto
%se asocia el vector de entrada j.
%D = Distorsión.

GRANDE=Inf;

%k = Dimensión de los vectores. N = Tamaño del alfabeto.

[k,N]=size(A);

%d = Número de vectores de entrada=número de columnas de X.

d=length(X(1,:));

%Definición de S

S=zeros(d,1);

%Bucle para asignar cada vector de entrada a uno del alfabeto.
%i = Índice del bucle.

i=1;

```
while(i<=d)

%Cojo el vector y lo guardo en 'v'
v=X(:,i);

%Segundo BUCLE para escoger el vector del alfabeto
j=1;
distancia=GRANDE;
indice=1;
while(j<=N)
dist=norm(v-A(:,j));
if(dist<distancia)

%A igualdad, cojo el índice menor
indice=j;
distancia=dist;
end
j=j+1;
end
S(i)=indice;
i=i+1;
end

%Cómputo de la distorsión total, como suma de las distorsiones de
%cada vector.
%La distorsión se mide como la Distancia Euclídea.

i=1;
Daux=0;

while(i<=d)
```

```
Daux=Daux+norm(X(:,i)-A(:,S(i)))^2;
i=i+1;
end

%Cálculo de la distorsión D:

D=Daux/d;
```

```
function A=f3(X,S,N);
%Función que asigna un centroide a cada región y evalúa el error total cometido.
%A=f3(X,S,N) escoge el mejor alfabeto 'A' para la partición de los vectores 'X'
%especificada en 'S'. 'N' es el número de elementos que queremos en el alfabeto.

GRANDE=Inf;

%k = Dimensión de los vectores. d = Número de vectores de entrada.

[k,d]=size(X);

%Declaración de A

A=ones(k,N);

%BUCLE. Crearemos N particiones a las que voy a asociar las que me
%vienen en S. Como puede que en S haya menos de N particiones,
%entonces algunas de las que creemos estar n vacías.
%Después, calcularemos el centroide de cada partición.
```

```
i=1;
while(i<=N)
V=[];
m=min(S);       %Elijo una partición de S.

if(m<GRANDE)
%Si se da esta condición es que hay una partición
                %en S. Busco los índices en los que S=m

indice=find(S==m);

 %y los borro de S

S(indice)=ones(length(indice),1)*GRANDE;

 %ahora recojo los vectores columna de X
                %indicados por 'indice', porque todos ellos
                %pertenecen a la misma partición.

V=X(:,indice);
                %calculo su centroide y lo guardo en A
A(:,i)=f4(V);

      else
%En este caso, no hay m s particiones en S
%He de elegir arbitrariamente el centroide de particiones
%vacías, y lo hago reasignándolo a uno de los vectores iniciales.
A(:,i)=X(:,i);
      end
      i=i+1;
end
```

```
function A=f4(V);
%Función llamada por f3.m para calcular los centroides
%Dada una partición en las columnas de V, elijo entre todos estos su centroide y lo
%guardo en A.

GRANDE=Inf;
%k = dimensión vectores, d = número de ellos en V

[k,d]=size(V);

if(d>1)
%Declaración de A
A=ones(k,1);
%Damos valor a 'A': la media de los vectores columna que hay en 'V':
A=mean(V')';
else
A=V;
end
```

7.1.4: Modo de Reconocimiento de Conjunto Abierto: Cálculo del Umbral

function umbral=cumbral(centr2,bdatos2,calculo)

%Función que determina, dado un conjunto de vectores característicos y centroides de %estos correspondientes a las declaraciones de entrenamiento de un solo locutor %conocido almacenadas en la base de datos, el nivel de umbral máximo entre %declaraciones del mismo locutor, de forma que, si se supera este umbral en la fase de %prueba se pueda desechar el locutor desconocido como no perteneciente a la base de %datos.

%Esta función se aplicará únicamente al reconocimiento de locutores de conjunto %abierto.

%centr2: Base de datos de los centroides de los vectores característicos de las declaraciones que se almacenan en la base de datos

%bdatos2: Base de datos de los vectores característicos de las declaraciones

%calculo: Método de cálculo de distorsión

[I,J]=size(bdatos2);

%I: número de locutores de la base de datos almacenada

%J: número de palabras de código de cada declaración

[fil,col]=size(centr2(1,:));

%fil, col: Dimensiones de los centroides de cada declaración

columna=(col/I);

if(calculo==1) %Si calculo es igual a 1

for i=1:I %Para cada locutor de la base de datos

```
    for j=1:I              %calculo la distancia euclídea entre centroides con
                           %los restantes locutores
      dist(i)=disteu1(centr2(:,(((i-1)*columna)+1):(i*columna)),bdatos2(i,:),...
      centr2(:,(((j-1)*columna)+1):(j*columna)),bdatos2(j,:));
    end
  end
  else
    if(calculo==2)         %Si calculo es igual a 2
      for i=1:I            %Para cada locutor de la base de datos
        for j=1:I          %calculo la distancia euclídea entre los centroides de
                           %dicho locutor y las palabras de código de los restantes
                           %locutores
          dist(i)=disteu2(centr2(:,(((i-
1)*columna)+1):(i*columna)),bdatos2(i,:),...
          bdatos2(j,:));
        end
      end
    else                   %Si calculo es igual a 3
      for i=1:I            %Para cada locutor de la base de datos
        for j=1:I          %calculo la distancia euclídea mínima entre centroides
                           %con los restantes locutores
          dist(i)=disteu3(centr2(:,(((i-1)*columna)+1):(i*columna)),...
          centr2(:,(((j-1)*columna)+1):(j*columna)));
        end                %Fin de 'for'
      end                  %Fin de 'for'
    end                    %Fin de 'if'
end                        %Fin de 'if'

%Calculamos el umbral máximo como una expresión del vector de distorsiones

%[umbral,numero]=max(dist);
```

%[umbralmax,numeromax]=max(dist);

%[umbralmin,numeromin]=min(dist);

%umbral=(umbralmax+umbralmin)/2;

%umbral=mean(dist);

%umbral=mean(dist)+(std(dist))/3;

%umbral=mean(dist)+(3(std(dist)));*

%suma=cumsum(dist);

%umbral=suma(length(dist))/length(dist);

%orden=sort(dist);

%umbral=orden(2);

7.1.5: Distorsión entre Locutores: Cálculo de la Distancia Euclídea

function deucl=disteu1(xtrain,bdtrain,xtest,bdtest)

%Función que determina la distancia euclídea entre cada centroide que corresponde al %vector característico de entrenamiento y el correspondiente al vector característico %de prueba para cada trama.

%Este método de cálculo requiere de la existencia de sincronismo entre las %declaraciones a nivel de tramas de voz

%xtrain: vectores centroide de la declaración de entrenamiento

%xtest: vectores centroide de la declaración de prueba

%bdtrain: vectores característicos de la declaración de entrenamiento. Cada vector %contiene la referencia a su centroide en xtrain

% bdtest: vectores característicos de la declaración de prueba. Cada vector contiene la %referencia a su centroide en xtest

```
%I, J: Dimensiones de la base de datos de vectores característicos
[I,J]=size(bdtrain);

%fil, col: Dimensiones de la base de datos de vectores centroide
[fil,col]=size(xtest);

%Inicializamos la distorsión dist.
dist=0;
difind=0;

for i=1:J                    %Para cada trama
%calculamos la distancia Euclídea (distancia local %elegida) entre el vector de entrenamiento (tomado como %centroide) y el de prueba

    dist=(xtrain(bdtrain(i),:)-xtest(bdtest(i),:));
    difind=difind+(dist*dist');
    d(i)=sqrt(difind);

end

%Calculamos la distancia total acumulada (distorsión) entre ambas
%declaraciones

dacum=cumsum(d);
ultimo=length(dacum);
deucl=dacum(ultimo);
```

function deucl=disteu2(xtrain,bdtrain,bdtest)

%Función que determina la distancia euclídea entre cada centroide que corresponde al %vector característico de entrenamiento y el vector característico de prueba para cada %trama.

%Este método de cálculo requiere de la existencia de sincronismo entre las %declaraciones a nivel de tramas de voz

%xtrain: vectores centroide de la declaración de entrenamiento

%bdtrain: vectores característicos de la declaración de entrenamiento. Cada vector %contiene la referencia a su centroide en xtrain

% bdtest: vectores característicos de la declaración de prueba. Aquí no se hace %referencia a xtest, puesto que se comparan directamente los vectores característicos %de prueba no cuantizados

%I, J: Dimensiones de la base de datos de vectores característicos

[I,J]=size(bdtrain);

%fil, col: Dimensiones de la base de datos de vectores centroide

[fil,col]=size(xtrain);

%Inicializamos la distorsión dist.

difind=0;

dist=0;

for i=1:J %Para cada trama

%calculamos la distancia Euclídea (distancia local %elegida) entre el vector de entrenamiento (tomado como %centroide) y el vector característico de prueba. Aquí %comparamos vectores de referencia a centroides con %vectores de energía directamente (no cuantizados)

dist=(xtrain(bdtrain(i),:)-bdtest(i,:));

*difind=difind+(dist*dist');*

d(i)=sqrt(difind);

end

%Calculamos la distancia total acumulada (distorsión) entre ambas
%declaraciones

dacum=cumsum(d);
ultimo=length(dacum);
deucl=dacum(ultimo);

function deucl=disteu3(xtrain,xtest)
%Función que determina la mínima distancia euclídea entre centroides
%correspondientes a los vectores característicos de entrenamiento prueba sin
%considerar la correspondencia entre tramas.
%Este método de cálculo no requiere de la existencia de sincronismo entre las
%declaraciones a nivel de tramas de voz, pero resulta más costoso a nivel
%computacional. Se evalúa aquí el grado de similitud global entre declaraciones,
%seleccionando el mínimo de cada comparación.

%xtrain: vectores centroide de la declaración de entrenamiento
%xtest: vectores centroide de la declaración de prueba

%I, J: Dimensiones de la base de datos de vectores centroide de entrenamiento
[I,J]=size(xtrain);

%fil, col: Dimensiones de la base de datos de vectores centroide de prueba
[fil,col]=size(xtest);

%Inicializamos la distorsión dist.

```
dist=0;
difind=0;
for i=1:fil      %Para cada vector centroide de prueba
    for j=1:I          %Para cada vector centroide de entrenamiento
%Calculamos la distancia Euclídea (distancia local elegida) %entre el vector centroide de prueba (tomado como centroide) y %los centroides de la declaración de entrenamiento
dist(j,1:col)=(xtest(i,:)-xtrain(j,:));
    end
    dif=dist*dist';
    for k=1:I          %Para cada vector centroide de entrenamiento
%tomamos la distorsión obtenida respecto al centroide de la %declaración de prueba
      ddiag(k)=dif(k,k);
    end
                       %y seleccionamos el mínimo de cada comparación
    [dmin(i),j]=min(ddiag);
%eliminando el vector de entrenamiento que haya proporcionado %ese mínimo para que no entre en juego en la comparación %siguiente.
    xtrain(j,:)=zeros(1,J);


    difind=difind+dmin(i);
    d(i)=sqrt(difind);
end

%Calculamos la distancia total acumulada (distorsión) entre ambas
%declaraciones

dacum=cumsum(d);
ultimo=length(dacum);
deucl=dacum(ultimo);
```

7.2: Fase de Identificación

*%%%%%%%**FASE DE PRUEBA DEL IDENTIFICADOR (TEST1)**%%%%%%%*
%%%
% Programa 'script' de MATLAB que genera el proceso de prueba del sistema de %reconocimiento del locutor desconocido para conjunto cerrado (diferentes locutores, %un solo micrófono, mic1, 5.000 muestras de voz por declaración, 195 coeficientes FB) %llamando a vtest.m y a identcc.m.

%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

```
%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr: base de datos de vectores centroide de cada declaración
%bdatos: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc: base de datos que asocia a cada locutor un número

load centr
load bdatos
load bdloc

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Método de cálculo de la distorsión

calculo=1;

 [vtest,bvtest]=vtest('c:\matlab\yo11.wav',nmuest,nsolap,ncoef,k,6251,11250,calculo);
locutor=identcc(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

```
%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TEST2)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto abierto (un único locutor, dos
%micrófonos, 5.000 muestras de voz por declaración, 195 coeficientes FB) llamando a
%vtest.m y a identca.m.

%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
```

```
%centr2: base de datos de vectores centroide de cada declaración
%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc2: base de datos que asocia a cada locutor un número

load centr2
load bdatos2
load bdloc2

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Método de cálculo de la distorsión

calculo=1;

%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef,k,5001,10000,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\elena211.wav',nmuest,nsolap,ncoef,k,9501,14500,calc
ulo);
[vtest,bvtest]=vtest('c:\matlab\mang211.wav',nmuest,nsolap,ncoef,k,11001,16000,calcu
lo);
%[vtest,bvtest]=vtest('c:\matlab\maria211.wav',nmuest,nsolap,ncoef,k,6301,11300,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\lina11.wav',nmuest,nsolap,ncoef,k,5101,10100,calculo
);
locutor=identca(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

```
%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TEST3)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto abierto (un único locutor, un
%solo micrófono, mic2, 5.000 muestras de voz por declaración, 195 coeficientes FB)
%llamando a vtest.m y a identca.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr2: base de datos de vectores centroide de cada declaración
```

```
%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc2: base de datos que asocia a cada locutor un número

load centr2
load bdatos2
load bdloc2

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Método de cálculo de la distorsión

calculo=1;

%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef,k,5001,10000,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\elena211.wav',nmuest,nsolap,ncoef,k,9501,14500,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\mang211.wav',nmuest,nsolap,ncoef,k,11001,16000,cal
culo);
%[vtest,bvtest]=vtest('c:\matlab\maria211.wav',nmuest,nsolap,ncoef,k,6301,11300,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\juanjo12.wav',nmuest,nsolap,ncoef,k,11501,16500,calc
ulo);

locutor=identca(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

```
%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TEST4)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto abierto (un único locutor, dos
%micrófonos, 5.000 muestras de voz por declaración, 65 coeficientes FB) llamando a
%vtest.m y a identca.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr2: base de datos de vectores centroide de cada declaración
```

```
%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc2: base de datos que asocia a cada locutor un número

load centr2
load bdatos2
load bdloc2

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro
%Método de cálculo de la distorsión

calculo=1;

%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef2,k2,5001,10000,c
alculo);
%[vtest,bvtest]=vtest('c:\matlab\elena216.wav',nmuest,nsolap,ncoef2,k2,5001,10000,ca
lculo);
%[vtest,bvtest]=vtest('c:\matlab\mang216.wav',nmuest,nsolap,ncoef2,k2,8201,13200,ca
lculo);
[vtest,bvtest]=vtest('c:\matlab\maria216.wav',nmuest,nsolap,ncoef2,k2,5401,10400,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\juanjo12.wav',nmuest,nsolap,ncoef2,k2,11501,16500,c
alculo);
locutor=identca(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

```
%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TEST5)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto cerrado (diferentes locutores,
%un solo micrófono, mic1, 5.000 muestras de voz por declaración, 65 coeficientes FB)
%llamando a vtest.m y a identcc.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes De Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr: base de datos de vectores centroide de cada declaración
```

```
%bdatos: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc: base de datos que asocia a cada locutor un número

load centr
load bdatos
load bdloc

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Método de cálculo de la distorsión

calculo=1;

[vtest,bvtest]=vtest('c:\matlab\yo11.wav',nmuest,nsolap,ncoef2,k2,6251,11250,calculo);
[vtest,bvtest]=vtest('c:\matlab\juanjo12.wav',nmuest,nsolap,ncoef2,k2,11501,16500,cal
culo);
locutor=identcc(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

*%%%%%%%**FASE DE PRUEBA DEL IDENTIFICADOR (TEST6)**%%%%%%%*
%%
% Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto cerrado (diferentes locutores,
%un solo micrófono, mic2, 5.000 muestras de voz por declaración, 65 coeficientes FB)
%llamando a vtest.m y a identcc.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes De Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr: base de datos de vectores centroide de cada declaración

```
%bdatos: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc: base de datos que asocia a cada locutor un número

load centr
load bdatos
load bdloc

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Método de cálculo de la distorsión

calculo=1;

%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef2,k2,5001,10000,c
alculo);
%[vtest,bvtest]=vtest('c:\matlab\elena216.wav',nmuest,nsolap,ncoef2,k2,5001,10000,ca
lculo);
%[vtest,bvtest]=vtest('c:\matlab\mang216.wav',nmuest,nsolap,ncoef2,k2,8201,13200,ca
lculo);
[vtest,bvtest]=vtest('c:\matlab\maria216.wav',nmuest,nsolap,ncoef2,k2,5401,10400,calc
ulo);

locutor=identcc(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

*%%%%%%%**FASE DE PRUEBA DEL IDENTIFICADOR (TEST7)**%%%%%%%*
%%%
% Programa 'script' de MATLAB que genera el proceso de prueba del sistema de %reconocimiento del locutor desconocido para conjunto abierto (un único locutor, dos %micrófonos, 4.000 muestras de voz por declaración, 195 coeficientes FB) llamando a %vtest.m y a identca.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas para
%las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr2: base de datos de vectores centroide de cada declaración

```
%bdatos2: base de datos de correspondencia entre el alfabeto de vectores
%característicos y los vectores centroide calculados
%bdloc2: base de datos que asocia a cada locutor un número

load centr2
load bdatos2
load bdloc2

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%Método de cálculo de la distorsión

calculo=1;

[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef,k,5401,9400,calculo
);
%[vtest,bvtest]=vtest('c:\matlab\elena211.wav',nmuest,nsolap,ncoef,k,9901,13900,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\mang211.wav',nmuest,nsolap,ncoef,k,11701,15700,cal
culo);
%[vtest,bvtest]=vtest('c:\matlab\maria211.wav',nmuest,nsolap,ncoef,k,6701,10700,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\lina11.wav',nmuest,nsolap,ncoef,k,5101,10100,calculo
);
locutor=identca(centr2,bdatos2,vtest,bvtest,bdloc2,calculo)
```

```
%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TEST8)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto cerrado (todos los locutores
%de la base de datos, dos micrófonos, 5.000 muestras de voz por declaración, 195
%coeficientes FB) llamando a vtest.m y a identcc.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de índices correspondientes a frecuencias críticas
%para las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr: Base de datos de vectores centroides de cada
```

```
%declaración.
%bdatos: Base de datos de correspondencia entre el alfabeto
%de vectores característicos y los vectores centroide calculados.
%bdloc: Base de datos que asocia a cada locutor un número.

load centr
load bdatos
load bdloc

%Método de cálculo de la distorsión

calculo=1;

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%[vtest,bvtest]=vtest('c:\matlab\angel16.wav',nmuest,nsolap,ncoef,k,4801,9800,calculo
);
%[vtest,bvtest]=vtest('c:\matlab\carmen16.wav',nmuest,nsolap,ncoef,k,4901,9900,calcu
lo);
%[vtest,bvtest]=vtest('c:\matlab\monica16.wav',nmuest,nsolap,ncoef,k,5301,10300,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\lina16.wav',nmuest,nsolap,ncoef,k,3451,8450,calculo);
%[vtest,bvtest]=vtest('c:\matlab\jordi16.wav',nmuest,nsolap,ncoef,k,1351,6350,calculo)
;
%[vtest,bvtest]=vtest('c:\matlab\yo16.wav',nmuest,nsolap,ncoef,k,9501,14500,calculo);
%[vtest,bvtest]=vtest('c:\matlab\juanjo16.wav',nmuest,nsolap,ncoef,k,3501,8500,calcul
o);
%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef,k,5001,10000,calc
ulo);
```

```
%[vtest,bvtest]=vtest('c:\matlab\yomic217.wav',nmuest,nsolap,ncoef,k,6301,11300,calculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic218.wav',nmuest,nsolap,ncoef,k,8501,13500,calculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic219.wav',nmuest,nsolap,ncoef,k,8501,13500,calculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic21a.wav',nmuest,nsolap,ncoef,k,4301,9300,calculo);
%[vtest,bvtest]=vtest('c:\matlab\elena216.wav',nmuest,nsolap,ncoef,k,5001,10000,calculo);
%[vtest,bvtest]=vtest('c:\matlab\mang216.wav',nmuest,nsolap,ncoef,k,8201,13200,calculo);
[vtest,bvtest]=vtest('c:\matlab\maria216.wav',nmuest,nsolap,ncoef,k,5401,10400,calculo);

locutor=identcc(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

```
%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TEST9)%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto cerrado (locutores grabados
%con micrófono 2, 5.000 muestras de voz por declaración, 195 coeficientes FB)
%llamando a vtest.m y a identcc.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef=195;

%Vector de Índices correspondientes a frecuencias críticas
%para las que calcularemos los coeficientes de Fourier-Bessel

k=[10,20,35,50,60,70,80,90,100,115,130,145,160,175,190];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr: Base de datos de vectores centroides de cada
```

```
%declaración.
%bdatos: Base de datos de correspondencia entre el alfabeto
%de vectores característicos y los vectores centroide calculados.
%bdloc: Base de datos que asocia a cada locutor un número.

load centr
load bdatos
load bdloc

%Método de cálculo de la distorsión

calculo=3;

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef,k,5001,10000,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\yomic217.wav',nmuest,nsolap,ncoef,k,6301,11300,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\yomic218.wav',nmuest,nsolap,ncoef,k,8501,13500,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\yomic219.wav',nmuest,nsolap,ncoef,k,8501,13500,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\yomic21a.wav',nmuest,nsolap,ncoef,k,4301,9300,calcu
lo);
%[vtest,bvtest]=vtest('c:\matlab\elena216.wav',nmuest,nsolap,ncoef,k,5001,10000,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\mang216.wav',nmuest,nsolap,ncoef,k,8201,13200,calc
ulo);
```

```
[vtest,bvtest]=vtest('c:\matlab\maria216.wav',nmuest,nsolap,ncoef,k,5401,10400,calcul
o);

locutor=identcc(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

%%%%%%%FASE DE PRUEBA DEL IDENTIFICADOR (TESTa)%%%%%%%

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Programa 'script' de MATLAB que genera el proceso de prueba del sistema de
%reconocimiento del locutor desconocido para conjunto cerrado (todos los locutores
%de la base de datos, dos micrófonos, 5.000 muestras de voz por declaración, 65
%coeficientes FB) llamando a vtest.m y a identcc.m.
%Definimos las variables del programa

%Número de muestras de cada una de las tramas en las que
%segmentaremos la señal de voz

nmuest=200;

%Número de muestras que consideraremos de solapamiento entre
%trama y trama segmentada

nsolap=100;

%Número de coeficientes de Fourier-Bessel que calcularemos
%para caracterizar cada trama segmentada de la señal de voz

ncoef2=65;
```

```
%Vector de Índices correspondientes a frecuencias críticas
%para las que calcularemos los coeficientes de Fourier-Bessel

k2=[10,20,35,50,60];

%Cargamos las bases de datos generadas en la fase de
%entrenamiento.
%centr: Base de datos de vectores centroides de cada
%declaración.
%bdatos: Base de datos de correspondencia entre el alfabeto
%de vectores característicos y los vectores centroide calculados.
%bdloc: Base de datos que asocia a cada locutor un número.

load centr
load bdatos
load bdloc

%Método de cálculo de la distorsión

calculo=3;

%Añadimos cada una de las declaraciones de referencia de los
%locutores a nuestra base de datos, pasando la ruta de acceso
%a los archivos de voz como parámetro

%[vtest,bvtest]=vtest('c:\matlab\angel16.wav',nmuest,nsolap,ncoef2,k2,4801,9800,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\carmen16.wav',nmuest,nsolap,ncoef2,k2,4901,9900,cal
culo);
%[vtest,bvtest]=vtest('c:\matlab\monica16.wav',nmuest,nsolap,ncoef2,k2,5301,10300,c
alculo);
```

```
%[vtest,bvtest]=vtest('c:\matlab\lina16.wav',nmuest,nsolap,ncoef2,k2,3451,8450,calcul
o);
%[vtest,bvtest]=vtest('c:\matlab\jordi16.wav',nmuest,nsolap,ncoef2,k2,1351,6350,calcu
lo);
%[vtest,bvtest]=vtest('c:\matlab\yo16.wav',nmuest,nsolap,ncoef2,k2,9501,14500,calcul
o);
%[vtest,bvtest]=vtest('c:\matlab\juanjo16.wav',nmuest,nsolap,ncoef2,k2,3501,8500,calc
ulo);
%[vtest,bvtest]=vtest('c:\matlab\yomic216.wav',nmuest,nsolap,ncoef2,k2,5001,10000,c
alculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic217.wav',nmuest,nsolap,ncoef2,k2,6301,11300,c
alculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic218.wav',nmuest,nsolap,ncoef2,k2,8501,13500,c
alculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic219.wav',nmuest,nsolap,ncoef2,k2,8501,13500,c
alculo);
%[vtest,bvtest]=vtest('c:\matlab\yomic21a.wav',nmuest,nsolap,ncoef2,k2,4301,9300,cal
culo);
%[vtest,bvtest]=vtest('c:\matlab\elena216.wav',nmuest,nsolap,ncoef2,k2,5001,10000,ca
lculo);
%[vtest,bvtest]=vtest('c:\matlab\mang216.wav',nmuest,nsolap,ncoef2,k2,8201,13200,ca
lculo);
[vtest,bvtest]=vtest('c:\matlab\maria216.wav',nmuest,nsolap,ncoef2,k2,5401,10400,calc
ulo);

locutor=identcc(centr,bdatos,vtest,bvtest,bdloc,calculo)
```

7.2.1: Extracción de la Característica del Locutor de Prueba

function [vtest,bvtest]=vtest(ruta,nmuest,nsolap,ncoef,k,a,b,calculo)

%Función principal de la fase de prueba en el reconocimiento, tanto de conjunto %cerrado como de conjunto abierto, que realiza llamadas a leedatos.m, para convertir %las señales de voz a secuencias de muestras, fb.m, para el cálculo de los coeficientes %FB, energia.m, para el cálculo de los vectores característicos de energía y cvect.m, en %caso de que se realice la cuantización vectorial de estos vectores característicos. %Devuelve los vectores característicos y/o sus centroides para cada declaración para %su posterior comparación con la base de datos almacenada en el sistema.

```
[voz,fs]=leedatos(ruta);
[cmvoz,vozrec]=fb(voz(a:b),nmuest,nsolap,ncoef,k);
clear voz
envoz=energia(cmvoz,k);
clear cmvoz

[coef,ntram]=size(envoz);
if((calculo==1)|(calculo==3))   %calculamos los centroides

metodo=1;

%10 CENTROIDES%

N=10;

%15 CENTROIDES%
%N=15;
%20 CENTROIDES%
%N=20;
```

```
[A,S,D]=cvect(N,envoz',metodo);

[fil,col]=size(A');
[totcol]=length(S');

vtest(1:fil,1:col)=A';
bvtest(1,1:totcol)=S';

else   %la base de datos se compondrá de los vectores característicos
vtest=0;
bvtest=envoz;
end
```

7.2.3: Determinación de la Identidad del Locutor

```
function locutor=identcc(centr,bdatos,vtest,bdvtest,bdloc,calculo)
%Función que determina, dado un conjunto de vectores característicos
%correspondientes a una declaración conocida y a un locutor desconocido, la
%identidad de dicho locutor de entre un conjunto cerrado de locutores cuyos vectores
%característicos se encuentran almacenados en una base de datos.
%I:número de locutores de la base de datos almacenada

[fil,col]=size(vtest);
[I,J]=size(bdatos);
[fila,columna]=size(bdvtest);

if(calculo==1)
for i=1:I
dist(i)=disteu1(centr(:,(((i-1)*col)+1):(i*col)),bdatos(i,:),vtest,bdvtest);
end
else
if(calculo==2)
for i=1:I
dist(i)=disteu2(centr(:,(((i-1)*columna)+1):(i*columna)),bdatos(i,:),bdvtest);
end     %Fin de 'for'
else
for i=1:I
dist(i)=disteu3(centr(:,(((i-1)*col)+1):(i*col)),vtest);
end
end
end
```

```
[dmin,numero]=min(dist);
locutor=bdlocut(numero,bdloc);
```

function locutor=identca(centr2,bdatos2,vtest,bdvtest,bdloc2,calculo)

```
%Función que determina, dado un conjunto de vectores característicos
%correspondientes a una declaración conocida y a un locutor desconocido,la identidad
%de dicho locutor de entre un conjunto cerrado de locutores cuyos vectores
%característicos se encuentran almacenados en una base de datos
%I:número de locutores de la base de datos almacenada

load umbral

[fil,col]=size(vtest);
[I,J]=size(bdatos2);
[fila,columna]=size(bdvtest);

if(calculo==1)
for i=1:I
dist(i)=disteu1(centr2(:,(((i-1)*col)+1):(i*col)),bdatos2(i,:),vtest,bdvtest);
end
else
if(calculo==2)
for i=1:I
dist(i)=disteu2(centr2(:,(((i-1)*columna)+1):(i*columna)),bdatos2(i,:),bdvtest);
end
else
for i=1:I
dist(i)=disteu3(centr2(:,(((i-1)*col)+1):(i*col)),vtest);
```

```
end
end
end

[dmin,numero]=min(dist);
dminimo=dmin+umbral(numero);
%dminimo=dmin;
%dminimo=dmin+mean(umbral);
%mumbral=mean(umbral)+(std(umbral)/3);
%mumbral=mean(umbral);
orden=sort(umbral);
mumbral=(orden(2)+orden(length(umbral)-1))/2;
if (dminimo<=mumbral)
locutor=bdlocut(numero,bdloc2);
else
locutor='El locutor no pertenece a la base de datos';
end
```

function nombre=bdlocut(numero,bdloc)

%Función que recibe como parámetro de entrada el número del %locutor identificado %en la base de datos, así como la base de datos que asocia un número a cada locutor. %Devuelve el nombre del locutor registrado en la base de datos que se identifica en el %número dado.

```
nombre=bdloc(numero,:);
```

Anexo I.: Glosario

Robustez:

En un sistema robusto, el funcionamiento se degrada gradualmente (más que catastróficamente) conforme las condiciones tornan más diferentes respecto a aquellas bajo las que el sistema fue entrenado. Las diferencias en las características del canal y en el entorno acústico deberían recibir una especial atención.

Portabilidad:

La portabilidad se refiere al logro de rapidez en diseño, desarrollo y despliegue en sistemas para nuevas aplicaciones. En la actualidad, los sistemas tienden a sufrir degradaciones significativas cuando son destinados a una nueva tarea. Con el fin de que vuelvan a elevar su rendimiento, deben ser entrenados sobre ejemplos específicos de la nueva tarea, lo cual supone un consumo de dinero y tiempo.

Adaptación:

¿Cómo pueden los sistemas adaptarse continuamente a las condiciones cambiantes (nuevos locutores, micrófono, tarea, etc.) y mejorar a través del uso? Esta adaptación puede ocurrir en los sistemas a muchos niveles, modelados de las palabras? Subword, pronunciación de las palabras, modelos de lenguaje, etc.

Modelado del Lenguaje:

Los sistemas actuales usan modelos estadísticos de lenguaje para ayudar a reducir el espacio de búsqueda y resolver la ambigüedad acústica. Conforme crezca el tamaño del vocabulario y otras restricciones se relajen para crear sistemas más habitables, será más importante obtener tantas restricciones como sea posible de los modelos de lenguaje; quizás incorporando restricciones sintácticas y semánticas que no se extraen de puros modelos estadísticos.

Medidas fiables:

La mayoría de los sistemas de reconocimiento de voz asignan marcas a hipótesis con el propósito de ordenarlas según un 'ranking'. Estas marcas no proporcionan una buena indicación de si la hipótesis es correcta o no, sino únicamente si es mejor que las otras hipótesis. Conforme realizamos funciones que requieren acciones, necesitamos métodos mejores para evaluara la absoluta bondad de las hipótesis.

Palabras fuera de Vocabulario:

Los sistemas se diseñan para usarlos con un conjunto particular de palabras, pero los usuarios pueden no conocer exactamente qué palabras están en el vocabulario del sistema. Esto conduce a un cierto porcentaje de palabras fuera del vocabulario en condiciones normales. Los sistemas deben contar con un método que detecte tales palabras, o terminarán mapeando una palabra del vocabulario sobre la palabra desconocida, conduciendo a error.

Voz Espontánea:

Los sistemas que se implementan para su uso real deben tratar con una gran variedad de fenómenos de voz espontánea, tales como pausas largas, falsos comienzos, dudas, construcciones no gramaticales, y otros comportamientos comunes que no se encuentras en la voz leída. Algunos desarrollos han progresado en este asunto, pero aún queda mucho por hacer.

Prosodia:

Se puede definir como la información suprasegmental de la voz; es decir, información que no puede localizarse en un segmento específico de sonido, o información que no cambia la identidad segmental de los segmentos de voz. Por ejemplo, patrones de variación de la frecuencia fundamental, duración, amplitud o intensidad, pausas, y cadencia del habla se ha demostrado que contienen información relativa a tales elementos prosódicos, como acento léxico, pausas en frases, y forma de frase declarativa o interrogativa. La prosodia se compone de un aspecto fonológico (caracterizado por unidades abstractas discretas) y un aspecto fonético (caracterizado por correlaciones acústicas de variación continua).

Los sistemas actuales no capturan la estructura prosódica. Cómo integrar la información prosódica en la arquitectura del reconocimiento es una cuestión crítica que aún no ha sido respondida.

Dinámicas de modelado:

Los sistemas asumen una secuencia de tramas de entrada que son tratadas como si fueran independientes. Pero se sabe que la percepción de las palabras y fonemas requiere la integración de características que reflejan los movimientos de las

articulaciones, que son dinámicos en la realidad. Cómo modelar dinámicas e incorporar esta información a los sistemas de reconocimiento es un problema sin resolver.

Voz Sobre Líneas Telefónicas:

El reconocimiento de voz telefónica es difícil porque cada canal telefónico tiene su propia Relación Señal a Ruido (SNR) y respuesta en frecuencia únicas. La voz sobre la línea telefónica puede además ser corrompida por interferencias transitorias y distorsión no lineal. Las aplicaciones basadas en el teléfono deben ser capaces de adaptarse a nuevos canales sobre la base de una cantidad muy pequeña de datos específicos del canal.

Entornos de Baja SNR:

Incluso con las técnicas de compensación actuales, la fiabilidad en el reconocimiento se degrada cuando la SNR del canal disminuye por debajo de 15 dB, aun cuando los humanos pueden obtener una fiabilidad en el reconocimiento excelente con SNRs más bajas.

Interferencia de Voz Co-canal:

Las interferencias de otros locutores plantean un problema mucho más difícil para el reconocimiento robusto que las interferencias de fuentes de ruido de banda ancha. Hasta la fecha, los esfuerzos en explotar la información específica de la voz para reducir los efectos de la interferencia co-canal de otros locutores han sido un rotundo fracaso.

Anexo II.: Referencias Bibliográficas

[1] ***"A Comparison of Speaker Identification Results Using Features Based on Cepstrum and Fourier-Bessel Expansion"***. K. Gopalan, Senior Member, IEEE; Timothy R. Anderson, Member, IEEE; Edward J. Cupples, Member, IEEE. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 3, Mayo, 1999.

[2] ***"An Algorithm for Vector Quantizer Design"***. Yoseph Linde, Member, IEEE; Andrés Buzo, Member, IEEE; Robert M. Gray, Senior Member, IEEE. *IEEE Transactions on Communications,* Vol. Com-28, NO. 1, Enero, 1980.

[3] ***"Speaker Identification Based on the Use of Robust Cepstral Features Obtained from Pole-Zero Transfer Functions"***. Mihailo S. Zilovic; Ravi P. Ramachandran, Member, IEEE; Richard J. Mammone, Senior Member, IEEE. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 1, Enero, 1999.

[4] ***"Robust Text-Independent Speaker Identification over Telephone Channels"***. Hema A. Murthy, Member, IEEE; Françoise Beaufays, Member, IEEE; Larry P. Heck, Member, IEEE; Mitchel Weintraub. *IEEE Transactions on Speech and Audio Processing,* Vol. 7, NO. 5, Septiembre, 1999.

[5] ***"Matemáticas Avanzadas para la Ingeniería"***. Volumen I. Erwin Kreyszig. Nueva edición en español de la 5ª edición en inglés. Ed. Noriega-Limusa.

[6] ***"Discrete Time Signal Processing"***. Alan V. Oppenheim, Ronal W. Schafer. Ed. Prentice Hall, Englewood Cliffs, New Jersey 07632.

[7] ***Apuntes de "Tratamiento Digital de Señales en Comunicaciones"***. 5º Curso Ingeniería de Telecomunicaciones. E.T.S.I. Universidad de Sevilla.

[8] ***"Communication Systems"***. Simon Haykin. John Wiley & Sons INC, 3ª Edición, New Baskerville, 1994.

[9] ***"Análisis Numérico"***. Burden & Faires. Ed. Iberoamericana.

[10] ***"Spoken Language Input"***. Victor Zue[♦] & Ron Cole[♦♦]. [♦]MIT Laboratory for Computer Science, Cambridge, Massachusets, USA. [♦♦]Oregon Graduate Institute of Science & Technology, Portland, Oregon, USA.
http://www.cslu.cse.ogi.edu/HLTsurvey/ch1node3.html

[11] ***"Speech Recognition"***. Victor Zue[♦], Ron Cole[♦♦] & Wayne Ward[♦♦♦]. [♦]MIT Laboratory for Computer Science, Cambridge, Massachusets,

USA. ††Oregon Graduate Institute of Science & Technology, Portland, Oregon, USA. †††Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.
http://www.cslu.cse.ogi.edu/HLTsurvey/ch1node4.html

[12] ***"Speaker Recognition"***. Sadoki Furui. NTT Human Interface Laboratories, Tokyo, Japan.
http://www.cslu.cse.ogi.edu/HLTsurvey/ch1node9.html

[13] ***"Vector Quantization"***.
http://eewww.eng.ohio-state.edu/~qasemm/vectorquantization/vq.html

[14] ***"Reconocimiento Automático del Lenguaje Hablado"***.
http://www.dec.uchile.cl/~sebrodi/62k/

[15] ***"Hankel Transform"***. Eric W. Weisstein.
http://www.br.campus.vt.edu/~akrowne/crc/math/h/h056.htm

[16] ***"Binary Quantization of Feature Vectors for Robust Text-Independent Speaker Identification"***. Zhong-Xuan Yuan, Associate Member, IEEE; Bo-Ling Xu; Chong-Zhi Yu. IEEE Transactions on Speech and Audio Processing, Vol. 7, NO. 1, Enero, 1999.